Plan2Sense: Open-World Task Planning in Epistemic States via Interleaved Ontic and Sensing Actions

Xiaotian Liu[1] **Armin Toroghi**[1] **Jiazhou Liang**[1] **Ali Pesaranghader**[2] **Scott Sanner**[1,3]
[1]University of Toronto [2]LG Electronics, Toronto AI Lab [3]Vector Institute of AI
{xiaotian.liu,armin.toroghi,joe.liang}@mail.utoronto.ca
ali.pesaranghader@lge.com, ssanner@mie.utoronto.ca

Abstract

Open-world task planning requires not only generating feasible plans but also acquiring the knowledge needed to execute them. Existing task planners for embodied agents either attempt to exhaustively gather information from the environment or generate plans under prior assumptions of the world and replan as new observations arrive. Both strategies are impractical in complex environments and can potentially expose agents to undesirable or unsafe situations. To enable effective planning in open-world settings, we propose PLAN2SENSE, a framework that grounds planning in the agent's epistemic state to determine what information is required for plan feasibility and how to acquire it. We formalize decision-making as a three-valued first-order *Knowledge-State Markov Decision Process* (KS-MDP) and develop a knowledge state goal regression algorithm (KSR) to construct sensing-aware plans in the open-world setting. Our formulation supports sensing actions for both predicate disambiguation and knowledge acquisition via object binding without assuming a fully known initial state or closed world of predefined objects. Across three open-world embodied benchmarks requiring active knowledge acquisition, PLAN2SENSE achieves higher success rates and efficiency over existing baselines by reasoning decision-theoretically over epistemic states to plan when to sense.

1 Introduction

Completing tasks in open-world settings [Reiter, 2001] requires agents to plan and act under incomplete information. Such incompleteness arises both from *state uncertainty* (e.g., whether an object is in a cabinet) and *missing object knowledge* (e.g., what food a user prefers and where it can be). An open-world agent must therefore act to resolve uncertainty about the current state, both object existence and properties over objects, while also strategically acquiring task-relevant knowledge necessary for successful plan execution.

A variety of approaches for planning under uncertainty predominantly focus on *state uncertainty* and treat sensing as a mechanism for disambiguating the current world state, e.g., [Pineau et al., 2003, Majercik and Littman, 2003, Bonet, 2010]. They typically plan over a distribution or set of possible world states, which is computationally expensive and often intractable in complex domains. Moreover, knowledge acquisition is usually modeled at the propositional level, requiring the enumeration of facts over a fixed, known object set. This assumption limits their applicability in open-world settings, where relevant objects and facts are not known *a priori* and can be potentially unbounded.

The challenges of open-world planning are evident in existing planners for embodied agents. Most embodied agents decouple sensing from planning, attempting to exhaustively gather information prior to plan synthesis [Chen et al., 2022, 2019]. Other planners operate under assumed world models (e.g., recent LLM-based planners), relying on prior knowledge to generate plans and trigger replanning when unexpected action outcomes are observed [Yao et al., 2022, Chen et al., 2024]. Both strategies

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

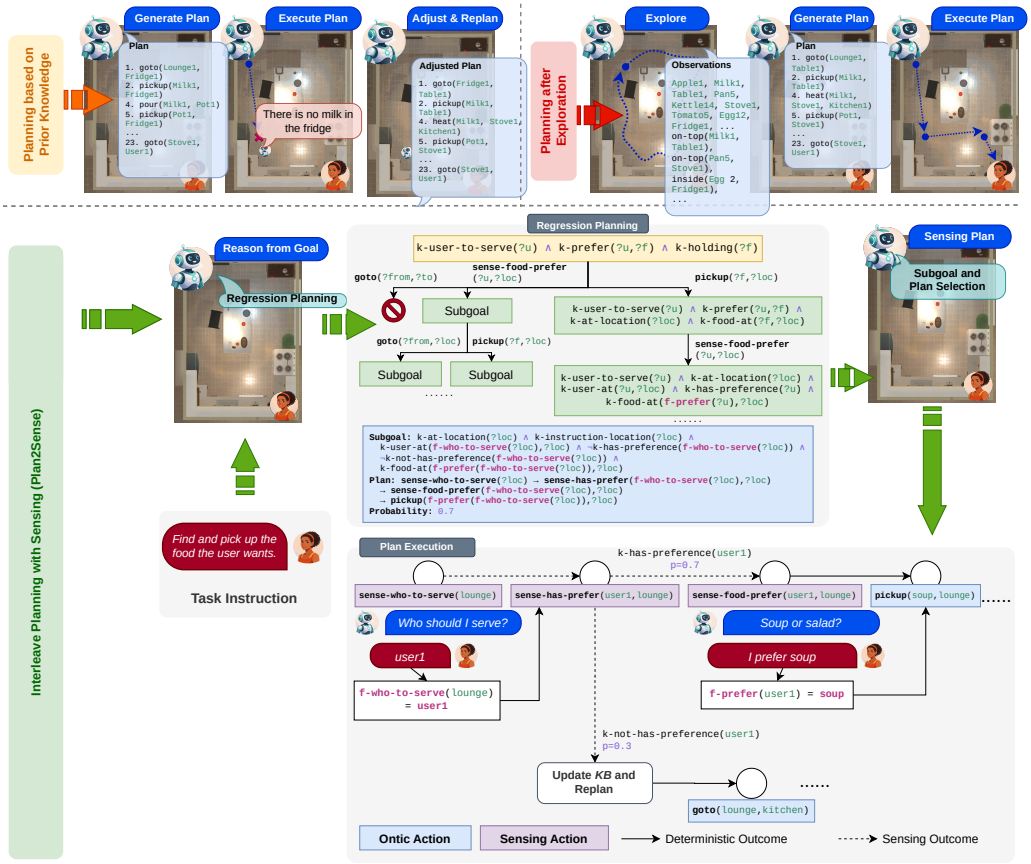


Figure 1: **Comparison of sensing and planning paradigms.** Unlike sense-then-act methods that require exhaustive information gathering, or failure-based replanning planners that react only to failure, Plan2Sense strategically interleaves planning with sensing that allows the agent to anticipate unexpected sensing outcomes.

are infeasible when physical interaction is costly, and can be undesirable in environments where naïvely applied sensing actions lead to inefficient plans, or unsafe and irreversible outcomes.

To address these challenges, we introduce PLAN2SENSE, a framework that plans directly over the agent's knowledge state. Specifically, we model decision-making in a *Knowledge-State Markov Decision Process* (KS-MDP) [Bayer-Zubek and Dietterich, 2000, 2005], which represents the agent's epistemic knowledge explicitly in the state space, capturing both what the agent knows and does not know. The KS-MDP supports both ontic actions, which modify the physical world, and sensing actions, which acquire the knowledge required to make plans feasible. Planning in the KS-MDP enables the strategic interleaving of ontic and sensing actions, allowing the agent to determine when to sense and what information to acquire.

Furthermore, we represent the KS-MDP using a compact *first-order MDP* (FO-MDP) [Boutilier et al., 2001] representation and apply *first-order decision-theoretic regression* (FO-DTR) [Boutilier et al., 2001, Sanner and Boutilier, 2009] to generate first-order (lifted) conditional plans without requiring complete specification of the initial state or a closed set of objects. This first-order representation also allows us to model sensing actions that both disambiguate predicate truth values (e.g., determining whether a stove is on by observing its status indicators) and acquire references to world objects through functions that sense (e.g., asking which person should be served next) [Scherl and Levesque, 1993, McIlraith and Son, 2002]. Figure 1 provides an illustration of our approach.

We evaluate PLAN2SENSE on three open-world benchmarks requiring active knowledge acquisition. Compared to sense-then-plan and failure-based replanning baselines, PLAN2SENSE achieves higher success rates and greater efficiency by reasoning over epistemic states to plan when to sense.

2 Background

The Situation Calculus [McCarthy, 1963, Reiter, 2001] provides a first-order logical language for axiomatizing dynamic worlds, and it supports a relational, parameterized representation suitable for open-world settings with an open set of objects. It has been extended to model Markov Decision Processes (MDPs) [Boutilier et al., 2001] and Partially Observable MDPs (POMDPs) [Sanner and Kersting, 2010] without requiring prior enumeration of objects or ground states. We provide a brief overview here and refer the reader to Sanner and Boutilier [2009] for a comprehensive treatment.

Situations, Fluents, and Actions In Situation Calculus, a *situation* is a first-order term representing a history of action executions and is constructed using the binary function symbol $\mathrm{do}(\cdot,\cdot)$, where $\mathrm{do}(a,s)$ denotes the successor situation obtained by executing action a in situation s. The world properties are described by *fluents*, which are predicate symbols whose truth values may vary across situations and whose final argument is a situation term. Under the Markovian assumption, the situation fully characterizes the world state, and thus we use *state* and *situation* interchangeably.

Successor State Axioms (SSAs) and Regression Successor state axioms (SSAs) define the logical conditions under which a fluent holds after an action is executed. Intuitively, a fluent holds in the successor situation if it was already true and is not negated by the action, or if the action makes it true [Reiter, 2001]. They are specified by the positive and negative effect axioms $\gamma_F^+(\vec{x},a,s)$ and $\gamma_F^-(\vec{x},a,s)$, which characterize when action a makes fluent F true or false:

$$\gamma_F^+(\vec{x},a,s) \Rightarrow F(\vec{x},\mathrm{do}(a,s)), \quad \gamma_F^-(\vec{x},a,s) \Rightarrow \neg F(\vec{x},\mathrm{do}(a,s))$$

Regression syntactically rewrites a fluent in the successor situation using its SSA:

$$\mathrm{Regr}\big(F(\vec{x},\mathrm{do}(a,s))\big) = \gamma_F^+(\vec{x},a,s) \ \vee \ \big(F(\vec{x},s) \wedge \neg\gamma_F^-(\vec{x},a,s)\big).$$

3 Methodology

PLAN2SENSE combines knowledge-state planning with decision-theoretic sensing in a lifted probabilistic framework to interleave sensing and acting. Our formulation builds on the situation-calculus treatment of knowledge and sensing actions [Scherl and Levesque, 2003] and the knowledge-state representation and 0-approximation [Baral, 1997], together with its compilation into PDDL by Palacios and Geffner [2009]. It extends the decision-theoretic KS-MDP perspective Bayer-Zubek and Dietterich [2005] to lifted sensing actions with functional effects [Bacchus et al., 1995, McIlraith and Scherl, 2000, Lakemeyer, 1999]. Through *knowledge-state regression* (KSR), PLAN2SENSE automatically composes sensing queries with subsequent actions, preserving nested functional dependencies without grounding their returned objects in advance. The resulting sensing-aware candidate plans are prioritized by their estimated probabilities of success, enabling the interleaving of sensing and acting during execution. We present KS-MDPs (§3.1), KSR (§3.2), and plan selection and execution (§3.3).

3.1 Knowledge State MDP

First-Order MDPs (FO-MDPs) [Boutilier et al., 2001, Sanner and Boutilier, 2009] describe states with first-order formulas over lifted object bindings, without enumerating propositional states, grounded actions, or a fixed object set. Our KS-MDP extends FO-MDPs with an explicit knowledge-state representation (§3.1.1) and an action representation that supports both ontic and sensing actions, including functional sensing actions that return object-valued bindings (§3.1.2).

3.1.1 Knowledge State Representation

Knowledge Fluents Following the three-valued knowledge-state representations of Baral [1997], Bayer-Zubek and Dietterich [2000, 2005] and the PDDL compilation approach of Palacios and Geffner [2009], we represent each domain fluent using epistemic predicates that distinguish between known-true, known-false, and unknown states. Let $\mathcal{F}$ be the set of underlying fluents. For each fluent $F(\vec{x}) \in \mathcal{F}$, we introduce two epistemic counterparts: $KF(\vec{x})$ and $K\neg F(\vec{x})$, denoting that $F(\vec{x})$ is known true or known false, respectively. We use $\mathcal{F}_K$ and $\mathcal{F}_{\neg K}$ to denote the sets of all known-true and known-false fluents induced by $\mathcal{F}$. Unknown fluents are represented implicitly by the absence

Example (1): Food-Serving PDDL Example

DECLARATIONS

```
(:types user food
        appliance location)
(:functions
  (f-who-to-serve ?loc - location)
    - user
  (f-prefer ?u - user) - food)
```

GOAL

Instruction: "Find and pick up the food the user wants."

```
(:goal (exists (?u - user
                ?f - food)
  (and (k-user-to-serve ?u)
       (k-prefer ?u ?f)
       (k-holding ?f))))
```

SENSING ACTIONS

```
(:action sense-who-to-serve
  :parameters (?loc - location)
  :precondition (and
    (k-at-location ?loc)
    (k-instruction-location ?loc))
  :effect
    (k-user-to-serve
      (f-who-to-serve ?loc)))

(:action sense-food-prefer
  :parameters (?u - user
               ?loc - location)
  :precondition (and
    (k-user-to-serve ?u)
    (k-at-location ?loc)
    (k-user-at ?u ?loc)
    (k-has-preference ?u))
  :effect
    (k-prefer ?u
      (f-prefer ?u)))
```

```
(:action sense-has-prefer
  :parameters (?u - user
               ?loc - location)
  :precondition (and
    (k-user-to-serve ?u)
    (k-at-location ?loc)
    (k-user-at ?u ?loc)
    (not (k-not-has-preference ?u))
    (not (k-has-preference ?u)))
  :effect (probabilistic
    0.7 (and
      (k-has-preference ?u))
    0.3 (and
      (k-not-has-preference ?u))))
```

ONTIC ACTIONS

```
(:action pickup
  :parameters (?f - food
    ?loc - location)
  :precondition (and
    (k-at-location ?loc)
    (k-food-at ?f ?loc))
  :effect (and
    (k-holding ?f)
    (not (k-food-at
      ?f ?loc))))
```

```
(:action goto
  :parameters
    (?from - location
     ?to - location)
  :precondition
    (k-at-location ?from)
  :effect (and
    (k-at-location ?to)
    (not (k-at-location
      ?from))))
```

```
(:action heat
  :parameters (?f - food
    ?a - appliance
    ?loc - location)
  :precondition (and
    (k-at-location ?loc)
    (k-holding ?f)
    (k-object-at ?a ?loc)
    (k-can-heat ?a ?f))
  :effect
    (k-is-heated ?f))
```

```
(:action serve
  :parameters (?u - user
    ?f - food
    ?loc - location)
  :precondition (and
    (k-holding ?f)
    (k-at-location ?loc))
  :effect (and
    (k-served-to ?f ?u)
    (not (k-holding ?f))))
```

of both epistemic counterparts, i.e., $\neg KF(\vec{x}) \wedge \neg K\neg F(\vec{x})$. Our formulation is knowledge-based: knowing that a fluent is true entails that the fluent itself is true, i.e., $KF(\vec{x}) \models F(\vec{x})$. However, we do not explicitly maintain the underlying world state $F(\vec{x})$; instead, PLAN2SENSE reasons over the agent's knowledge state, represented by what the agent knows, i.e., $KF(\vec{x}), K\neg F(\vec{x})$, while unknown information remains implicit.

Epistemic Consistency We enforce mutual exclusivity between known-true and known-false epistemic fluents to ensure that the agent's knowledge state is fully observable and well defined. For each underlying fluent $F(\vec{x})$, at most one of its epistemic counterparts may hold in any knowledge state: $KF(\vec{x}, s)$ or $K\neg F(\vec{x}, s)$. Intuitively, the agent cannot simultaneously know a fact to be true and know it to be false. Formally, for all $F \in \mathcal{F}$,

$$\forall \vec{x}, s\ KF(\vec{x}, s) \Rightarrow \neg K\neg F(\vec{x}, s), \qquad \forall \vec{x}, s\ K\neg F(\vec{x}, s) \Rightarrow \neg KF(\vec{x}, s).$$

We assume a single-agent setting that models only the planning agent's knowledge state K and do not model nested epistemic beliefs.

3.1.2 Action Representation

[Sanner and Boutilier, 2009] have shown that the PPDDL notation can be converted to FO-MDPs. We adopt a PPDDL-style notation [Younes and Littman, 2004] to specify KS-MDP action theories. Each action a is defined by $\langle \mathrm{pre}(a), \mathrm{Eff}(a) \rangle$, where $\mathrm{pre}(a)$ is a set of precondition literals and $\mathrm{Eff}(a) = \{(p_1, \mathrm{eff}_1), \ldots, (p_n, \mathrm{eff}_n)\}$ is a set of probabilistic effects, with each eff_i occurring with probability p_i and $\sum p_i = 1$. The epistemic fluents KF and $K\neg F$ are represented in PPDDL syntax by `k-F` and `k-not-F`, respectively, while unknown is represented by the conjoined negation of both. An example PPDDL-style domain is shown in Example 1, which we use as a running example.

Ontic Action Compilation Ontic actions are physical interventions that modify the world state; in Example 1, these are `pickup`, `goto`, `heat`, and `serve`. In the KS-MDP, an ontic action a_{ontic} has a single set of deterministic effects, $\mathrm{Eff}(a_{\mathrm{ontic}}) = (1.0, \mathrm{eff})$, with no new variables or functions introduced in the effect. We omit the cancellation effects used by Palacios and Geffner [2009], which are needed to handle conditional effects that our formalism does not support. Instead, all conditions are action preconditions that must be explicitly satisfied before execution.

Example (2): Compiling a Stochastic Sensing Action into Nature's-Choice Outcomes

The stochastic sensing action `sense-has-prefer` from Example 1 senses the value of `has-preference` for user `?u`. It is compiled into two deterministic outcome actions, one per sensed value, which Nature selects with probability 0.7 and 0.3, respectively.

TRUE OUTCOME

```
P(sense-has-prefer-true, sense-has-prefer, s) = 0.7
(:action sense-has-prefer-true
  :parameters (?u - user
               ?loc - location)
  :precondition (and
    (k-user-to-serve ?u)
    (k-at-location ?loc)
    (k-user-at ?u ?loc)
    (not (k-not-has-preference ?u))
    (not (k-has-preference ?u)))
  :effect (k-has-preference ?u))
```

FALSE OUTCOME

```
P(sense-has-prefer-false, sense-has-prefer, s) = 0.3
(:action sense-has-prefer-false
  :parameters (?u - user
               ?loc - location)
  :precondition (and
    (k-user-to-serve ?u)
    (k-at-location ?loc)
    (k-user-at ?u ?loc)
    (not (k-not-has-preference ?u))
    (not (k-has-preference ?u)))
  :effect (k-not-has-preference ?u))
```

Stochastic Sensing Actions A stochastic sensing action modifies the agent's knowledge state of fluent truth; e.g., `sense-has-prefer` in Example 1 senses whether the user has a food preference. Formally, a Boolean sensing action a_{sense} has stochastic effects $\text{Eff}(a_{\text{sense}}) = \{(p, \text{eff}^+), (1-p, \text{eff}^-)\}$: when its preconditions are satisfied, its nondeterministic effects involve known-true $KF(\vec{x})$ or known-false $K\neg F(\vec{x})$ fluents. We compile each such action schema $A_{sense}(\vec{x})$ into a set of deterministic *Nature's choice* actions [Reiter, 2001], one per outcome, $\Omega\big(A_{sense}(\vec{x})\big) = \{a^+(\vec{x}),\, a^-(\vec{x})\}$, as shown for `sense-has-prefer` in Example 2. When the agent executes $A_{sense}(\vec{x})$ in state s, Nature samples a deterministic outcome action $a(\vec{x}) \in \Omega\big(A_{sense}(\vec{x})\big)$ according to $P\big(a(\vec{x}), A_{sense}(\vec{x}), s\big)$ and executes it; thus SSAs of $A_{sense}(\vec{x})$ are defined over deterministic outcomes $\{a^+(\vec{x}),\, a^-(\vec{x})\}$.

Functional Sensing Actions Beyond determining fluent truth values, sensing can answer "who", "what", and "where" questions posed to external sources (e.g., a human, a database, or an LLM), returning an object that satisfies a relevant relation even when that object is unknown at planning time. Following prior work on functional sensing [Scherl and Levesque, 1993, 2003, Lakemeyer, 1999, McIlraith and Son, 2002, Baier and McIlraith, 2006], we represent a sensing query as an object-valued function $f_{\text{sense}}(\vec{x})$. Here, $\vec{x}$ denotes the query arguments, and $f_{\text{sense}}(\vec{x}) = y$ denotes the returned object y. For example, `f-prefer(?u:user)` denotes the food preferred by user u. Functional sensing is formalized through action-fluent SSAs with functional effects (cf. Example 4, Steps 2 & 4). During regression, a functional sensing term such as `f-prefer(?u:user)` in an SSA unifies with variables in subgoal fluents (cf. Example 4, Step 2) and is evaluated only on action execution.

To guard against null observations, we decompose each functional sensing action into a *two-stage* action sequence: (i) a *stochastic sensing action* "does an answer exist?" query, followed by (ii) a *functional sensing action* executed only when Nature's choice for the stochastic sensing action is affirmative. This action sequence acts like a $KWhether$ gating precondition for sensing [Levesque et al., 1996] adapted to the FO-MDP setting. As illustrated in Example 5 (Step 3), `sense-has-prefer` first determines whether the user has a preference, represented by `k-has-preference` (we assume veracity) followed by the functional sensing action `sense-food-prefer` in Example 5 (Step 2).

3.2 Knowledge State Regression Planning

We perform *first-order regression* to identify the set of feasible plans, their sufficient preconditions, and the corresponding probabilities of reaching the goal within the horizon h. Each plan $\tau_i(\vec{x}^{(i)}) = \langle A_1(\vec{x}_1), \ldots, A_k(\vec{x}_k)\rangle$, with $k \leq h$ has a corresponding precondition formula $\Psi_{\tau_i(\vec{x}^{(i)})}(s)$ defined using the case notation, where $\vec{x} = \langle \vec{x}^{(1)}, \ldots, \vec{x}^{(m)}\rangle$:

$$\Psi_{\tau_i(\vec{x})}(s) = \text{case}\big[\, \psi^{(1)}_{\tau_i(\vec{x}^{(1)})}(s) : p_1;\ \ldots;\ \psi^{(m)}_{\tau_i(\vec{x}^{(m)})}(s) : p_m \,\big].$$

This case expression encodes state formula $\psi^{(i)}_{\tau_i(\vec{x}^{(i)})}(s)$ (referred to as plan preconditions) and their corresponding probabilities p_i of reaching a state that satisfies the goal condition $\mathcal{G}$ after executing plan τ_i. To reduce notational clutter, we use τ_i and $\Psi_{\tau_i}(s)$ instead of $\tau_i(\vec{x})$ and $\Psi_{\tau_i(\vec{x})}(s)$. Each nonzero case $\psi^{(j)}_{\tau_i}(s) : p_j$ yields a conditional plan (ϕ_i, τ_i, p_j), whose condition ϕ_i consists of the predicates of $\psi^{(j)}_{\tau_i}$ *without* functional terms; the remaining predicates are checked once sensing resolves their functional terms. The resulting solution set is $\Pi = \{(\phi_1(\vec{x}), \tau_1(\vec{x}), p_1),\ \ldots,\ (\phi_n(\vec{x}), \tau_n(\vec{x}), p_n)\}$.

Example (3): Goal Initialization and Case Notation

GOAL

Instruction: "Find and pick up the food the user wants."

```
(:goal (exists (?u - user
                ?f - food)
  (and (k-user-to-serve ?u)
       (k-prefer ?u ?f)
       (k-holding ?f))))
```

GOAL FORMULA AND $\Psi_\emptyset$

$\mathcal{G}(s) \equiv \exists$?u, ?f.

k-user-to-serve(?u, s)
$\wedge$ k-prefer(?u, ?f, s)
$\wedge$ k-holding(?f, s)

Regression starts from the terminal case statement, which assigns value 1 to goal states and 0 otherwise:

$$\Psi_\emptyset(s) = \begin{array}{|c|c|}\hline \mathcal{G}(s) & 1.0 \\ \hline \neg\mathcal{G}(s) & 0 \\ \hline \end{array}$$

CASE OPERATORS USED BY KSR

Scaling by an outcome probability ($\otimes$):

$$\begin{array}{|c|c|}\hline \top & 0.7 \\ \hline \end{array} \otimes \begin{array}{|c|c|}\hline \phi & 1.0 \\ \hline \neg\phi & 0 \\ \hline \end{array} = \begin{array}{|c|c|}\hline \phi & 0.7 \\ \hline \neg\phi & 0 \\ \hline \end{array}$$

Combining outcomes by cross-sum ($\oplus$):

$$\begin{array}{|c|c|}\hline \phi_1 & 0.7 \\ \hline \neg\phi_1 & 0 \\ \hline \end{array} \oplus \begin{array}{|c|c|}\hline \phi_2 & 0.3 \\ \hline \neg\phi_2 & 0 \\ \hline \end{array} = \begin{array}{|c|c|}\hline \phi_1 \wedge \phi_2 & 1.0 \\ \hline \phi_1 \wedge \neg\phi_2 & 0.7 \\ \hline \neg\phi_1 \wedge \phi_2 & 0.3 \\ \hline \neg\phi_1 \wedge \neg\phi_2 & 0 \\ \hline \end{array}$$

Goal Initialization and Knowledge State Goal Regression Given the goal condition $\mathcal{G}$, we initialize regression with a terminal case statement assigning probability 1 to states satisfying the goal formula, 0 otherwise. Example 3 shows the goal initialization for our running example, together with the case operators $\otimes$ and $\oplus$ used below. We define the goal-oriented *Knowledge State Regression* (KSR) operator as an adaptation of First-Order Decision-Theoretic Regression (FO-DTR) [Boutilier et al., 2001]. KSR produces a case representation of regressed plan preconditions Ψ defining *both* epistemic preconditions of a plan and probability of goal-reachability when executing action $A(\vec{x})$:

$$\Psi_{A(\vec{x})\circ\tau}(s) = \mathrm{KSR}[\Psi_\tau(s), A(\vec{x})] = \bigoplus_{a\in\Omega(A(\vec{x}))} \big[P(a, A(\vec{x}), s) \otimes \mathrm{Regr}\big(\Psi_\tau(\mathrm{do}(a, s))\big)\big] .$$

During regression, action effects are unified with matching subgoal literals, the resulting substitution is propagated through the remaining subgoal, and branches with no consistent unifier are discarded.

Plan Regression For a plan $\tau = \langle A_1(\vec{x}_1), \ldots, A_k(\vec{x}_k)\rangle$, we recursively apply KSR:

$$\Psi_\tau(s) = \mathrm{KSR}\big[\ \ldots \mathrm{KSR}[\Psi_\emptyset(s), A_k(\vec{x}_k)] \ldots, A_1(\vec{x}_1)\ \big].$$

The resulting case expression enumerates sufficient state conditions and their cumulative probabilities of reaching the goal after executing the full plan τ_i. Our restriction of PDDL to unquantified conjunctive preconditions and unconditional effects as well as existential goals guarantees all regressed plan conditions ϕ_i will be existentially quantified. Example 4 traces this recursion for the goal of Example 3 (with the fully worked KSR calculation provided in Appendix A). Example 5 executes the resulting plan (in reverse order of the regression). This representation enables selecting plans in Π that maximize the probability of success given the agent's current knowledge state.

3.3 Plan Generation, Execution, and Replanning

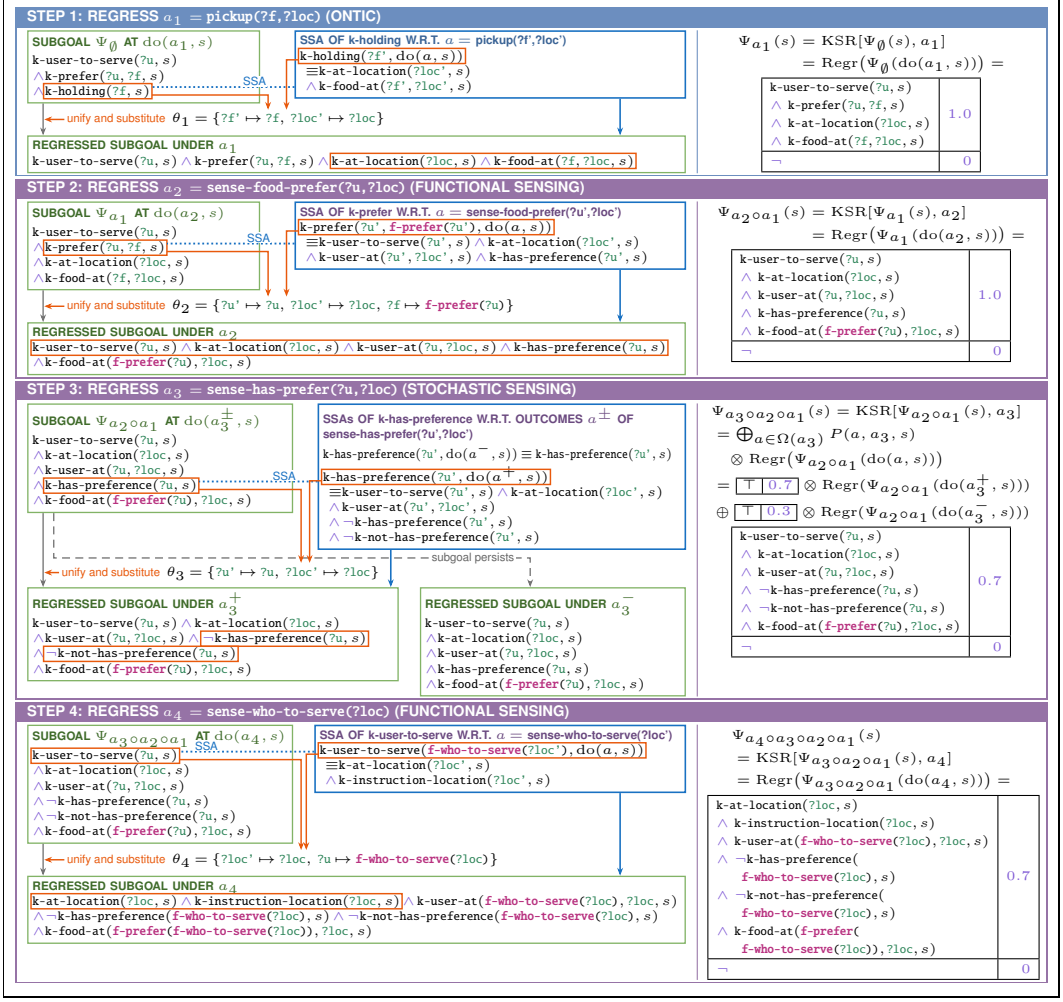
After Knowledge State Regression, PLAN2SENSE applies Algorithm 1 in Appendix B to obtain a set of lifted candidate plans $\Pi = \{(\phi_i, \tau_i, p_i)\}_{i=1}^n$, where ϕ_i is the regressed plan precondition, τ_i is the lifted plan, and p_i is its probability of reaching the goal. Candidate plans are prioritized by p_i, and the highest-priority plan precondition ϕ_i satisfying the current knowledge base, $KB \models \phi_i$, selects plan τ_i for execution (Example 5). Notably, unresolved object-valued query terms such as `f-prefer(?u:user)` were previously removed from ϕ_i and will not be substituted and checked until later plan steps when their functional sensing queries are executed; failure to satisfy delayed plan precondition evaluation (e.g., the user asked for milk, but no milk is present) will trigger replanning.

After each action, the observed outcome effects are incorporated into KB, and the regressed condition of the remaining plan is checked against the updated KB. *Replanning is triggered when this check fails*, either due to (i) substituting the evaluation of a functional sensing term after sensing violates a previously uncheckable precondition, or (ii) a stochastic sensing action produces a Nature's choice outcome different from the branch assumed by the current plan; in either case, PLAN2SENSE reselects from Π a candidate plan whose regressed condition is satisfied by the updated KB. Example 5 shows a full illustration of plan execution. Appendices C, D, and E describe the full execution procedure.

Because the replanning and recovery mechanism reselects among regressed candidate plans rather than computing a complete contingent policy, it is not guaranteed to be optimal. Optimality could instead be obtained by computing an optimal *policy* with symbolic dynamic programming [Boutilier et al., 2001, Sanner and Boutilier, 2009] (at potentially greater computational cost for policy generation) or by monitoring plan optimality during execution [Fritz and McIlraith, 2007].

Example (4): Knowledge State Regression, Step by Step

Starting from $\Psi_\emptyset(s)$ (Example 3), we regress through the actions of Example 1 in reverse execution order. Each tree shows the subgoal being regressed (top left) and the action's SSA for the targeted fluent (top right). Substituting θ in the subgoal and expanding the fluent with the SSA body together give the regressed subgoal (bottom). The right column gives the resulting case statement, where $\neg$ denotes the complement of the nonzero case. Detailed derivations of SSAs and KSR are shown in Appendix A.



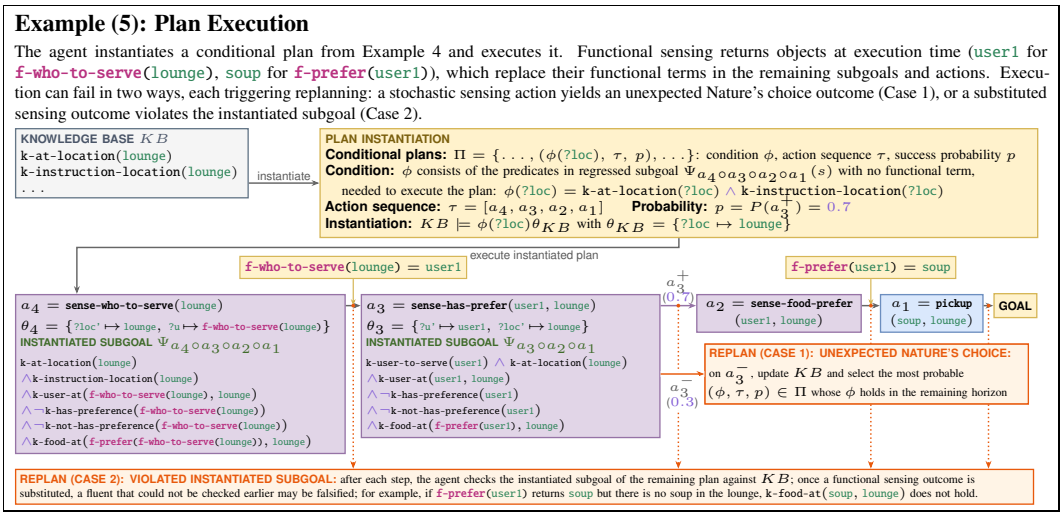
4 Experiments

Our experimental evaluation is designed to rigorously assess the effectiveness of PLAN2SENSE in open-world task planning that requires the agent to actively acquire knowledge from multiple sources. We design our experiments to address the following three research questions:

- **RQ1:** In open-world tasks, how does PLAN2SENSE compare to existing planning paradigms in terms of task success and planning efficiency?
- **RQ2:** How does the performance of PLAN2SENSE scale with increasing environment complexity relative to competing approaches?
- **RQ3:** To what extent does PLAN2SENSE reduce the execution of irreversible actions that lead to undesirable or unsafe states compared to alternative methods?

4.1 Benchmarks

The proposed research questions require benchmarks with two key properties: (1) *Multi-source knowledge acquisition*, where physical exploration alone is insufficient and agents must acquire task-relevant information from external sources; and (2) *Irreversibility*, where certain actions can lead to irreversible failure states, imposing explicit safety constraints. Building on existing benchmarks [Shridhar et al., 2020, Chevalier-Boisvert et al., 2023, Thrun, 1999], we develop three partially



Method	Epistemic ALFWorld				Doors-and-Keys QA				Embodied Heaven–Hell			
	SR ↑	Steps ↓	Tokens ↓	Time ↓	SR ↑	Steps ↓	Tokens ↓	Time ↓	SR ↑	Steps ↓	Tokens ↓	Time ↓
Failure-Based Replanning Methods												
Zero-Shot LLM (Naive)	$13\%_{\downarrow 76}$	$37.2_{\uparrow 13.7}$	$34{,}562_{\uparrow 32{,}944}$	$70.5_{\uparrow 51.0}$	$22\%_{\downarrow 62}$	$78.3_{\uparrow 20.7}$	$98{,}963_{\uparrow 97{,}638}$	$77.8_{\uparrow 65.6}$	$45\%_{\downarrow 46}$	$82.3_{\uparrow 19.1}$	$37{,}691_{\uparrow 36{,}475}$	$79.1_{\uparrow 68.4}$
Zero-Shot LLM (CoT)	$18\%_{\downarrow 71}$	$29.5_{\uparrow 6.0}$	$45{,}623_{\uparrow 44{,}005}$	$68.9_{\uparrow 49.4}$	$23\%_{\downarrow 61}$	$88.1_{\uparrow 30.5}$	$125{,}664_{\uparrow 124{,}339}$	$79.2_{\uparrow 67.0}$	$48\%_{\downarrow 43}$	$78.5_{\uparrow 15.3}$	$51{,}360_{\uparrow 50{,}144}$	$70.4_{\uparrow 59.7}$
ReAct	$62\%_{\downarrow 27}$	$22.3_{\downarrow 1.2}$	$614{,}235_{\uparrow 612{,}617}$	$83.6_{\uparrow 64.1}$	$53\%_{\downarrow 31}$	$73.3_{\uparrow 15.7}$	$215{,}848_{\uparrow 214{,}523}$	$89.2_{\uparrow 77.0}$	$67\%_{\downarrow 24}$	$77.6_{\uparrow 14.4}$	$193{,}654_{\uparrow 192{,}438}$	$85.5_{\uparrow 74.8}$
LLM-Refined PDDL	$8\%_{\downarrow 81}$	$44.8_{\uparrow 21.3}$	$514{,}238_{\uparrow 512{,}620}$	$85.2_{\uparrow 65.7}$	$6\%_{\downarrow 78}$	$92.5_{\uparrow 34.9}$	$189{,}331_{\uparrow 188{,}006}$	$103.2_{\uparrow 91.0}$	$12\%_{\downarrow 79}$	$90.6_{\uparrow 27.4}$	$228{,}961_{\uparrow 227{,}745}$	$89.6_{\uparrow 78.9}$
***Sense-then-Plan Methods* (w/ GT Model)**												
Random Explor.	$20\%_{\downarrow 69}$	$42.6_{\uparrow 19.1}$	$0_{\downarrow 1{,}618}$	$12.6_{\downarrow 6.9}$	$36\%_{\downarrow 48}$	$80.6_{\uparrow 23.0}$	$0_{\downarrow 1{,}325}$	$6.8_{\downarrow 5.4}$	$41\%_{\downarrow 50}$	$88.8_{\uparrow 25.6}$	$0_{\downarrow 1{,}216}$	$8.4_{\downarrow 2.3}$
Guided Explor.	$26\%_{\downarrow 63}$	$39.0_{\uparrow 15.5}$	$1{,}675_{\uparrow 57}$	$18.2_{\downarrow 1.3}$	$34\%_{\downarrow 50}$	$76.3_{\uparrow 18.7}$	$3{,}863_{\uparrow 2{,}538}$	$10.3_{\downarrow 1.9}$	$37\%_{\downarrow 54}$	$85.4_{\uparrow 22.2}$	$1{,}643_{\uparrow 427}$	$12.4_{\uparrow 1.7}$
PLAN2SENSE (OURS)	**89%**	**23.5**	**1,618**	**19.5**	**84%**	**57.6**	**1,325**	**12.2**	**91%**	**63.2**	**1,216**	**10.7**

Table 1: Overall performance of evaluated baselines and the proposed PLAN2SENSE across three benchmarks. Subscripts indicate the numeric difference relative to PLAN2SENSE.

observable, open-world domains: *Epistemic ALFWorld*, *Doors-and-Keys QA*, and *Embodied Heaven–Hell*. Each domain is extended to require agents to acquire task-relevant information from external sources and reason about irreversible actions to achieve goals safely. We evaluate PLAN2SENSE on these three benchmarks to assess its ability to integrate epistemic and ontic reasoning.

4.2 Baselines

We compare PLAN2SENSE against two broad classes of baselines. First, we consider two sense-then-plan exploration strategies: (1) *Random Exploration*, in which the agent executes random action sequences to collect state information, and (2) *LLM-Guided Exploration*, in which an LLM guides exploration to acquire informative observations. In both cases, the accumulated knowledge is subsequently compiled into a PDDL model for symbolic planning. Second, we compare against LLM-based planning methods, including zero-shot naive prompting, Chain-of-Thought (CoT) prompting, and iterative refinement methods such as ReAct. We also include a PDDL-based replanning baseline, *LLM-Refined PDDL*, which revises plans based on feedback from failed executions. This baseline builds on prior closed-world PDDL domain-generation methods [Song et al., 2023, Liu et al., 2024], but is adapted to the open-world setting with sensing. Example prompts are provided in Appendix D.

4.3 Evaluation Setup and Metrics

We evaluate methods across four dimensions: task success, efficiency, reliability, and scalability. Task success is measured by the Success Rate (*SR*), and efficiency is assessed by the Number of Steps per Task (*# Step*). Reliability is quantified by the Percentage of Tasks Terminated due to unrecoverable failure states (*Irrever. Actions*) and incomplete acquisition of multi-source environment knowledge (*Insuff. Explor.*). Computational costs are also evaluated using the LLM Token Count (*# Tok*) and the Average Clock Execution Time, measuring end-to-end runtime performance. We use `GPT-5.2` as the LLM backbone across methods.

5 Results and Discussion

RQ1: Overall Performance We first evaluate PLAN2SENSE across three benchmarks against a diverse set of baselines. As shown in Table 1, PLAN2SENSE consistently achieves substantially higher success rates (SR) than both failure-based replanning and sense-then-plan baselines, while using significantly fewer LLM tokens, and it is most notable when compared to failure-based replanning approaches. PLAN2SENSE also yields shorter episodes, as measured by the number of executed steps. Importantly, baselines with comparable episode lengths (e.g., ReAct) exhibit markedly lower success rates. As shown in Table 2, their shorter trajectories primarily result from early termination caused by irreversible actions, rather than successful execution of complete plans toward the goal. Baselines fail mainly because they do not reason explicitly about missing knowledge: LLM/ReAct agents often act too early and trigger irreversible failures. LLM-refined PDDL struggles with complex sensing dependencies, while sense-then-plan over-explores, causing inefficient or insufficient exploration.

Method	Epistemic ALFWorld		Doors-and-Keys QA		Embodied Heaven–Hell	
	Insuff. Explor.	Irrever. Actions	Insuff. Explor.	Irrever. Actions	Insuff. Explor.	Irrever. Actions
Failure-Based Replanning Methods						
Zero-Shot LLM (Naive)	36%	64%	45%	55%	63%	37%
Zero-Shot LLM (CoT)	39%	61%	58%	42%	60%	40%
ReAct	45%	55%	64%	36%	75%	25%
LLM-Refined PDDL	90%	10%	88%	12%	90%	10%
***Sense-then-Plan Methods* (w/ GT Model)**						
Random Explor.	82%	18%	77%	23%	82%	18%
LLM-Guided Explor.	79%	21%	85%	15%	77%	23%
PLAN2SENSE (Ours)	95%	5%	98%	2%	95%	5%

Table 2: Percentage of failures due to insufficient exploration versus irreversible actions across evaluated methods and benchmarks.

RQ2: Scalability In Figure 2, we analyze the scalability of PLAN2SENSE across three dimensions in a task design: the number of terminal states, the overall grid size (i.e., explorable space), and the total number of objects per task in DOORKEYS QA. PLAN2SENSE remains robust as the number of objects and terminal states increases, whereas ReAct and LLM-guided exploration exhibit a pronounced decline in success rate under higher object counts and more terminal contingencies. This robustness stems from two key design features of PLAN2SENSE. First, regression focuses sensing on goal-relevant information rather than the full environment. Second, explicitly modelling knowledge acquisition via sensing actions further prunes the effective search space by narrowing attention to relevant objects and terminal contingencies. We observe a performance decline for PLAN2SENSE as grid size increases (cf. the middle panel of Figure 2), due to the embodied nature of the environment and the increased need for physical exploration in larger spaces. However, this degradation is substantially less severe than that of the evaluated baselines.

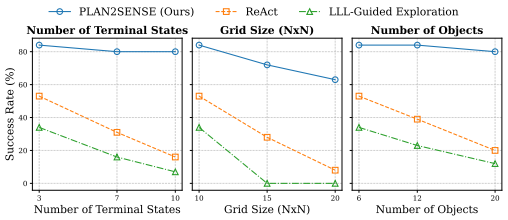


Figure 2: Success rate of evaluated methods across three major task dimensions: number of terminal states, grid size, and number of objects.

RQ3: Ability to Avoid Irreversible Actions As shown in Table 2, failure-based replanning methods, including zero-shot LLMs and ReAct, exhibit the highest rate of failures due to irreversible actions. By relying on implicit assumptions rather than explicit epistemic reasoning, these methods frequently ignore the necessary sensing steps (e.g., verifying task-relevant properties or checking safety-critical regions), which leads to premature transitions into irreversible failure states. Sense-then-plan baselines, which do not rely on replanning, also struggle to avoid irreversible failures. Their decoupling of knowledge acquisition from planning results in exhaustive and unguided exploration, often exhausting the episode budget before a feasible plan is identified. In contrast, PLAN2SENSE rarely fails due to irreversible actions: by explicitly reasoning over knowledge requirements and deferring high-risk actions until the necessary information has been sensed, it consistently avoids unsafe and unrecoverable outcomes.

Ablation Appendix F shows that PLAN2SENSE remains robust to inaccurate sensing-probability estimates and executes the correct sensing action far more reliably than baselines without explicit epistemic-state modeling.

6 Related Work

Knowledge-State and Epistemic Planning While planning under incomplete information has been extensively studied through conformant and contingent planning [Bonet, 2010, Majercik and Littman, 2003], a complementary line of work represents uncertainty explicitly in the agent's knowledge state. The 0-approximation semantics [Baral, 1997, Tu et al., 2007] represents each fluent as known true, known false, or unknown, supporting both sensing and acting. Palacios and Geffner [2009] compile conformant planning into classical planning using explicit knowledge literals. Knowledge-level planners such as PKS [Petrick and Bacchus, 2004] use know-whether and know-value information to construct conditional plans, while Baier et al. [2014] compile problems with epistemic and ontic goals into non-epistemic planning. From a decision-theoretic perspective, Bayer-Zubek and Dietterich [2000, 2005] introduce a three-valued epistemic MDP that jointly models uncertainty over sensing outcomes and the costs of alternative query specifications. Related work in multi-agent epistemic planning reasons over nested beliefs and can be compiled into classical planning for scalable search [Muise et al., 2015, 2022], with applications such as proactive robotic assistance [Shvo et al., 2022]. PLAN2SENSE combines many of these perspectives by adopting the PDDL-style knowledge-state representation of Palacios and Geffner [2009] with implicit unknowns, thereby inheriting the soundness and incompleteness of the 0-approximation, lifting the epistemic MDP view to PPDDL with epistemic transition probabilities, and applying a first-order MDP representation with first-order decision-theoretic regression [Boutilier et al., 2001, Sanner and Boutilier, 2009] to derive lifted, open-world, sensing-aware conditional plans ranked by goal-achievement probability.

Sensing and Regression Scherl and Levesque [1993, 2003] provide the theoretical foundations for epistemic states and functional sensing actions. Building on this foundation, McIlraith and Scherl [2000], Reiter [2001] develop regression for knowledge-producing actions. Early Golog-based work on planning with sensing largely focused on Boolean observations: McIlraith and Scherl [2000] use regression as the basis for conditional planning of complex tests, while Lakemeyer [1999] constructs conditional plans from Golog programs with Boolean sensing. McIlraith and Son [2002] extend Golog with functional sensing, and Baier and McIlraith [2006, 2022] use regression to compile programs with sensing into primitive actions for operator-based planners. Sensing also plays an important role in epistemic planning, which studies how agents plan to change their own and others' knowledge and beliefs. PLAN2SENSE represents a variant on some of these works that uses functional sensing, epistemic state, and (most distinctly) a stochastic first-order MDP Boutilier et al. [2001] epistemic representation of sensing outcomes via transition probabilities to produce lifted conditional plans prioritized by goal-achievement probability that interleave ontic and sensing actions.

Open-World Embodied Planning Embodied planning approaches have addressed information acquisition through exploration, active perception, and online replanning [Tateo et al., 2022, Chen et al., 2022, Yao et al., 2022, Ren et al., 2023, Valmeekam et al., 2023, Hazra et al., 2024, Shinn et al., 2024, Huang et al., 2022], alongside active object recognition and exploration methods [Bohg et al., 2017, Caselles-Dupré et al., 2021, Zhu et al., 2023]. Recent work has also explored task-directed active knowledge acquisition and goal-regression-guided sensing in open-world embodied environments [Liu et al., 2025, 2026b,a]. PLAN2SENSE integrates functional sensing actions directly into the agent's plan, enabling the interleaving of ontic and sensing actions for task-directed knowledge acquisition and execution for open-world embodied tasks.

7 Conclusion and Limitations

We introduced PLAN2SENSE, a novel framework for open-world task planning that interleaves ontic actions with on-demand knowledge acquisition through sensing actions by explicitly reasoning over the agent's epistemic state. Empirically, we demonstrated that PLAN2SENSE substantially improves task success rate and efficiency across three open-world task planning benchmarks, while consistently avoiding irreversible and unsafe actions. These advantages position PLAN2SENSE as an important framework for scalable, safe, and effective planning in complex open-world environments. Key limitations of this work include its focus on discrete action and belief representations as well as a lack of nested beliefs [Muise et al., 2015, 2022] and an explicit theory of mind [Shvo et al., 2022] that represent important directions for improved capability in future extensions of PLAN2SENSE.

Acknowledgments and Disclosure of Funding

This work was supported by LG Electronics, Toronto AI Lab Grant Ref No. 2024-0565 and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. RS-2024-00457882, National AI Research Lab Project). We thank Hyejeong Jeon, Chul Lee, and Jaehong Kim for their support of this project.

References

Fahiem Bacchus, Joseph Y Halpern, and Hector Levesque. Reasoning about noisy sensors in the situation calculus. *Contract*, 49620(91-C):0080, 1995.

Jorge A Baier and Sheila A McIlraith. On planning with programs that sense. *KR*, 6:492–502, 2006.

Jorge A Baier and Sheila A McIlraith. Knowledge-based programs as building blocks for planning. *Artificial Intelligence*, 303:103634, 2022.

Jorge A Baier, Brent Mombourquette, and Sheila A McIlraith. Diagnostic problem solving via planning with ontic and epistemic goals. In *KR*, 2014.

Chitta Baral. Approximate reasoning about actions in presence of sensing and incomplete information. In *Proc. of International Logic Programming Symposium (ILPS 97)*, 1997.

Valentina Bayer-Zubek and Thomas Dietterich. A POMDP approximation algorithm that anticipates the need to observe. In *Pacific Rim International Conference on Artificial Intelligence*, pages 521–532. Springer, 2000.

Valentina Bayer-Zubek and Thomas G Dietterich. Integrating learning from examples into the search for diagnostic policies. *Journal of Artificial Intelligence Research*, 24:263–303, 2005.

Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

Blai Bonet. Conformant plans and beyond: Principles and complexity. *Artificial Intelligence*, 174 (3-4):245–269, 2010.

Craig Boutilier, Raymond Reiter, and Bob Price. Symbolic dynamic programming for first-order mdps. In *IJCAI*, volume 1, pages 690–700, 2001.

Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. Scod: Active object detection for embodied agents using sensory commutativity of action sequences. *arXiv preprint arXiv:2107.02069*, 2021.

Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. *arXiv preprint arXiv:2209.09874*, 2022.

Guanqi Chen, Lei Yang, Ruixing Jia, Zhe Hu, Yizhou Chen, Wei Zhang, Wenping Wang, and Jia Pan. Language-augmented symbolic planner for open-world task planning. *arXiv preprint arXiv:2407.09792*, 2024.

Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, 36:73383–73394, 2023.

Christian Fritz and Sheila A McIlraith. Monitoring plan optimality during execution. In *ICAPS*, pages 144–151, 2007.

Rishi Hazra, Pedro Zuidberg Dos Martires, and Luc De Raedt. Saycanpay: Heuristic planning with large language models using learnable domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20123–20133, 2024.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR, 2022.

Gerhard Lakemeyer. On sensing and off-line interpreting in golog. In *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*, pages 173–189. Springer, 1999.

Hector J Levesque, Fangzhen Lin, and Ray Reiter. How to execute a conditional? In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 1996.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency, 2023. *URL https://arxiv. org/abs/2304.11477*, 2024.

Xiaotian Liu, Ali Pesaranghader, Hanze Li, Punyaphat Sukcharoenchaikul, Jaehong Kim, Tanmana Sadhu, Hyejeong Jeon, and Scott Sanner. Open-world planning via lifted regression with llm-inferred affordances for embodied agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20881–20897, 2025.

Xiaotian Liu, Ali Pesaranghader, Jaehong Kim, Tanmana Sadhu, Hyejeong Jeon, and Scott Sanner. Activevoo: Value of observation guided active knowledge acquisition for open-world embodied lifted regression planning. *Advances in Neural Information Processing Systems*, 38:106621–106647, 2026a.

Xiaotian Liu, Armin Toroghi, Jiazhou Liang, David Courtis, Ruiwen Li, Ali Pesaranghader, Jaehong Kim, Tanmana Sadhu, Hyejeong Jeon, and Scott Sanner. Natural language pddl (nl-pddl) for open-world goal-oriented commonsense regression planning in embodied ai. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Stephen M Majercik and Michael L Littman. Contingent planning under uncertainty via stochastic satisfiability. *Artificial Intelligence*, 147(1-2):119–162, 2003.

John McCarthy. Situations, actions, and causal laws. Technical report, 1963.

Sheila A McIlraith and Richard Scherl. What sensing tells us: Towards a formal theory of testing for dynamical systems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-00)*, pages 483–490, 2000.

Sheila A. McIlraith and Tran Cao Son. Adapting golog for composition of semantic web services. In *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR-2002)*, pages 482–493, Toulouse, France, April 22-25 2002.

Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian R Pearce, and Liz Sonenberg. Efficient multi-agent epistemic planning: Teaching planners about nested belief. *Artificial Intelligence*, 302:103605, 2022.

Hector Palacios and Hector Geffner. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research*, 35:623–675, 2009.

Ronald PA Petrick and Fahiem Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In *ICAPS*, pages 2–11, 2004.

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Ijcai*, volume 3, pages 1025–1032, 2003.

Raymond Reiter. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT press, 2001.

Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*, 2023.

S. Sanner and K. Kersting. Symbolic dynamic programming for first-order POMDPs. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, Atlanta, Georgia, July 19-23 2010. AAAI Press.

Scott Sanner and Craig Boutilier. Practical solution techniques for first-order mdps. *Artificial Intelligence*, 173(5-6):748–788, 2009.

Richard B Scherl and Hector J Levesque. The frame problem and knowledge-producing actions. In *AAAI*, volume 93, pages 689–695, 1993.

Richard B Scherl and Hector J Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2):1–39, 2003.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Maayan Shvo, Ruthrash Hari, Ziggy O'Reilly, Sophia Abolore, Sze-Yuh Nina Wang, and Sheila A McIlraith. Proactive robotic assistance via theory of mind. In *2022 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 9148–9155. IEEE, 2022.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023.

Davide Tateo, Davide Antonio Cucci, Matteo Matteucci, and Andrea Bonarini. Object structural points representation for graph-based semantic monocular localization and mapping. *arXiv preprint arXiv:2206.10263*, 2022.

Sebastian Thrun. Monte carlo POMDPs. *Advances in neural information processing systems*, 12, 1999.

Phan Huy Tu, Tran Cao Son, and Chitta Baral. Reasoning and planning with sensing actions, incomplete information, and static causal laws using answer set programming. *Theory and Practice of Logic Programming*, 7(4):377–450, 2007.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005, 2023.

Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Håkan LS Younes and Michael L Littman. Ppddl1. 0: An extension to pddl for expressing planning domains with probabilistic effects. *Techn. Rep. CMU-CS-04-162*, 2:99, 2004.

Hao Zhu, Raghav Kapoor, So Yeon Min, Winson Han, Jiatai Li, Kaiwen Geng, Graham Neubig, Yonatan Bisk, Aniruddha Kembhavi, and Luca Weihs. Excalibur: Encouraging and evaluating embodied exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14931–14942, 2023.

A Knowledge State Regression Example Expanded

This appendix expands Example 4 using the same action names, PDDL-style variables, and regressed subgoals as Examples 1–4. We first state the simplification rules and the effect axioms and successor state axioms (SSAs) used in Steps 1–4. We then work through the regression, case multiplication, and cross-sum calculation for the stochastic sensing action in Step 3. The compact Ψ statements record the retained effect-supporting contributions. The full SSAs include persistence, and the regression and case calculations apply the no-no-op simplification to the complete subgoals.

Simplification

No no-op assumption. We assume that the domain does not contain `no-op` actions: every executed action must change the knowledge state of the agent. To restrict candidate generation to informative regressions, we additionally discard any regressed disjunct that is logically equivalent to the subgoal being regressed. Specifically, if

$$\mathrm{Regr}\big(G(\vec{x}, \mathrm{do}(a, s))\big) \equiv \Big(\bigvee_i \Psi_i^{\mathrm{DNF}}(\vec{x}, a, s)\Big) \vee G(\vec{x}, s),$$

where the Ψ_i^{DNF} are the remaining disjuncts, then the simplified regression is

$$\mathrm{Regr}\big(G(\vec{x}, \mathrm{do}(a, s))\big) = \bigvee_i \Psi_i^{\mathrm{DNF}}(\vec{x}, a, s).$$

Thus, a candidate plan is not extended through a branch that merely reproduces its existing subgoal. This rule applies to a disjunct equivalent to the complete subgoal, not to individual persistence conjuncts needed to preserve other required facts.

A.1 Effect Axioms and SSAs for Steps 1–4

For each step, we write the positive and negative effect conditions with respect to the specified action, displaying its name and parameters explicitly. The positive effect condition establishes the fluent in the successor situation, while the negative effect condition makes it false. The effect conditions include the action preconditions from Examples 1–2. Free variables in the effect axioms and SSAs are universally quantified. Each SSA combines the effects with persistence.

Step 1: `pickup`. Let $a_1 = \texttt{pickup}(\texttt{?f}, \texttt{?loc})$. The targeted fluent is `k-holding`. Its positive and negative effect conditions with respect to this action are

$$\begin{aligned}&\gamma^{+}_{\texttt{k-holding}}(\texttt{?f}, \texttt{pickup}(\texttt{?f}, \texttt{?loc}), s)\\ &\quad = \texttt{k-at-location}(\texttt{?loc}, s) \wedge \texttt{k-food-at}(\texttt{?f}, \texttt{?loc}, s),\\ &\gamma^{-}_{\texttt{k-holding}}(\texttt{?f}, \texttt{pickup}(\texttt{?f}, \texttt{?loc}), s) = \bot.\end{aligned}$$

Successor State Axiom. Since `pickup` does not delete `k-holding`, its SSA is

$$\begin{aligned}&\texttt{k-holding}(\texttt{?f}, \mathrm{do}(\texttt{pickup}(\texttt{?f}, \texttt{?loc}), s))\\ &\quad \equiv \Big(\texttt{k-at-location}(\texttt{?loc}, s) \wedge \texttt{k-food-at}(\texttt{?f}, \texttt{?loc}, s)\Big) \vee \big(\texttt{k-holding}(\texttt{?f}, s) \wedge \neg\bot\big).\end{aligned}$$

With no-op simplification,

$$\begin{aligned}&\equiv \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-prefer}(\texttt{?u}, \texttt{?f}, s)\\ &\quad \wedge \texttt{k-at-location}(\texttt{?loc}, s) \wedge \texttt{k-food-at}(\texttt{?f}, \texttt{?loc}, s).\end{aligned}$$

Step 2: `sense-food-prefer`. Let $a_2 = \texttt{sense-food-prefer}(\texttt{?u}, \texttt{?loc})$. Its positive and negative effect conditions for `k-prefer` are

$$\begin{aligned}&\gamma^{+}_{\texttt{k-prefer}}(\texttt{?u}, \texttt{f-prefer}(\texttt{?u}), \texttt{sense-food-prefer}(\texttt{?u}, \texttt{?loc}), s)\\ &\quad = \texttt{k-user-to-serve}(\texttt{?u}, s)\\ &\qquad \wedge \texttt{k-at-location}(\texttt{?loc}, s) \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s),\\ &\gamma^{-}_{\texttt{k-prefer}}(\texttt{?u}, \texttt{f-prefer}(\texttt{?u}), \texttt{sense-food-prefer}(\texttt{?u}, \texttt{?loc}), s) = \bot.\end{aligned}$$

Successor State Axiom. Since the negative effect condition is $\bot$, the SSA is

$$\begin{aligned}&\texttt{k-prefer}(\texttt{?u}, \texttt{f-prefer}(\texttt{?u}), \mathrm{do}(\texttt{sense-food-prefer}(\texttt{?u}, \texttt{?loc}), s))\\ &\quad \equiv \Big(\texttt{k-user-to-serve}(\texttt{?u}, s)\\ &\qquad \wedge \texttt{k-at-location}(\texttt{?loc}, s) \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s)\Big)\\ &\qquad \vee \big(\texttt{k-prefer}(\texttt{?u}, \texttt{f-prefer}(\texttt{?u}), s) \wedge \neg\bot\big).\end{aligned}$$

With no-op simplification,

$$\begin{aligned}&\equiv \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s)\\ &\quad \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s)\\ &\quad \wedge \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s).\end{aligned}$$

Step 3: `sense-has-prefer`**.** Let

$$\begin{aligned} a_3 &= \texttt{sense-has-prefer}(\texttt{?u}, \texttt{?loc}), \\ a_3^+ &= \texttt{sense-has-prefer-true}(\texttt{?u}, \texttt{?loc}), \\ a_3^- &= \texttt{sense-has-prefer-false}(\texttt{?u}, \texttt{?loc}), \\ \Omega(a_3) &= \{a_3^+, a_3^-\}. \end{aligned}$$

For each deterministic outcome, we give the effect conditions, SSA, and retained effect-supporting condition for its targeted fluent.

True outcome: `k-has-preference`**.** The positive and negative effect conditions are

$$\begin{aligned} &\gamma^+_{\texttt{k-has-preference}}(\texttt{?u}, \texttt{sense-has-prefer-true}(\texttt{?u}, \texttt{?loc}), s) \\ &\quad = \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ &\qquad \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ &\qquad \wedge \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s), \\ &\gamma^-_{\texttt{k-has-preference}}(\texttt{?u}, \texttt{sense-has-prefer-true}(\texttt{?u}, \texttt{?loc}), s) = \bot. \end{aligned}$$

Successor State Axiom. Since the negative effect condition is $\bot$, the SSA is

$$\begin{aligned} &\texttt{k-has-preference}(\texttt{?u}, \mathrm{do}(\texttt{sense-has-prefer-true}(\texttt{?u}, \texttt{?loc}), s)) \\ &\quad \equiv \Big(\texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ &\qquad \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ &\qquad \wedge \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s)\Big) \\ &\qquad \vee \big(\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\bot\big). \end{aligned}$$

With no-op simplification,

$$\begin{aligned} &\equiv \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ &\quad \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ &\quad \wedge \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s). \end{aligned}$$

False outcome: `k-not-has-preference`**.** The positive and negative effect conditions are

$$\begin{aligned} &\gamma^+_{\texttt{k-not-has-preference}}(\texttt{?u}, \texttt{sense-has-prefer-false}(\texttt{?u}, \texttt{?loc}), s) \\ &\quad = \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ &\qquad \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ &\qquad \wedge \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s), \\ &\gamma^-_{\texttt{k-not-has-preference}}(\texttt{?u}, \texttt{sense-has-prefer-false}(\texttt{?u}, \texttt{?loc}), s) = \bot. \end{aligned}$$

Successor State Axiom. Since the negative effect condition is $\bot$, the SSA is

$$\begin{aligned} &\texttt{k-not-has-preference}(\texttt{?u}, \mathrm{do}(\texttt{sense-has-prefer-false}(\texttt{?u}, \texttt{?loc}), s)) \\ &\quad \equiv \Big(\texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ &\qquad \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ &\qquad \wedge \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s)\Big) \\ &\qquad \vee \big(\texttt{k-not-has-preference}(\texttt{?u}, s) \wedge \neg\bot\big). \end{aligned}$$

With no-op simplification,

$$\begin{aligned} &\equiv \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ &\quad \wedge \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ &\quad \wedge \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s). \end{aligned}$$

Step 4: `sense-who-to-serve`**.** Let $a_4 = \texttt{sense-who-to-serve}(\texttt{?loc})$. The relevant effect conditions with respect to this action are

$$\begin{aligned} &\gamma^+_{\texttt{k-user-to-serve}}(\texttt{f-who-to-serve}(\texttt{?loc}), \texttt{sense-who-to-serve}(\texttt{?loc}), s) \\ &\quad = \texttt{k-at-location}(\texttt{?loc}, s) \wedge \texttt{k-instruction-location}(\texttt{?loc}, s), \\ &\gamma^-_{\texttt{k-user-to-serve}}(\texttt{f-who-to-serve}(\texttt{?loc}), \texttt{sense-who-to-serve}(\texttt{?loc}), s) = \bot. \end{aligned}$$

Successor State Axiom.

$$\begin{aligned} &\texttt{k-user-to-serve}(\texttt{f-who-to-serve}(\texttt{?loc}), \mathrm{do}(\texttt{sense-who-to-serve}(\texttt{?loc}), s)) \\ &\quad \equiv \Big(\texttt{k-at-location}(\texttt{?loc}, s) \\ &\qquad \wedge \texttt{k-instruction-location}(\texttt{?loc}, s)\Big) \vee \big(\texttt{k-user-to-serve}(\texttt{f-who-to-serve}(\texttt{?loc}), s) \wedge \neg\bot\big). \end{aligned}$$

With no-op simplification,

$$\begin{aligned}\equiv\ & \texttt{k-at-location}(\texttt{?loc}, s) \wedge \texttt{k-instruction-location}(\texttt{?loc}, s) \\ & \wedge \texttt{k-user-at}(\texttt{f-who-to-serve}(\texttt{?loc}), \texttt{?loc}, s) \\ & \wedge \neg\texttt{k-has-preference}(\texttt{f-who-to-serve}(\texttt{?loc}), s) \\ & \wedge \neg\texttt{k-not-has-preference}(\texttt{f-who-to-serve}(\texttt{?loc}), s) \\ & \wedge \texttt{k-food-at}(\texttt{f-prefer}(\texttt{f-who-to-serve}(\texttt{?loc})), \texttt{?loc}, s).\end{aligned}$$

A.2 Worked KSR Calculation for Step 3

Regressing the complete subgoal through both outcomes. Step 3 regresses the case statement obtained after Step 2. Here, a standalone $\neg$ denotes the complement of the union of the nonzero conditions:

$$\Psi_{a_2 \circ a_1}(s)$$

$$= \begin{array}{|l|r|} \hline \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) & \\ \wedge\ \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s) & : 1.0 \\ \wedge\ \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) & \\ \hline \neg & : 0 \\ \hline \end{array}.$$

The fluent `k-has-preference` is regressed through each outcome separately; the other fluents in the subgoal persist.

True outcome: SSA replacement and regression. Using the retained true-outcome result from Section A.1,

$$\begin{aligned}& \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ & \wedge\ \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ & \wedge\ \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s),\end{aligned}$$

substitution in the complete subgoal gives

$$\begin{array}{|l|r|} \hline \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) & \\ \wedge\ \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) & \\ \wedge\ \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s) & : 1.0 \\ \wedge\ \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) & \\ \hline \neg & : 0 \\ \hline \end{array}.$$

False outcome. The false outcome leaves the original subgoal unchanged:

$$\begin{array}{|l|r|} \hline \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) & \\ \wedge\ \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s) & : 1.0 \\ \wedge\ \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) & \\ \hline \neg & : 0 \\ \hline \end{array}.$$

Weighting the outcome-specific regressions. Multiplying each outcome case by its probability gives $0.7 \times 1.0 = 0.7$ for the true outcome and $0.3 \times 1.0 = 0.3$ for the false outcome; the zero rows remain zero.

True outcome.

$$\begin{array}{|l|r|} \hline \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) & \\ \wedge\ \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) & \\ \wedge\ \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s) & : 0.7 \\ \wedge\ \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) & \\ \hline \neg & : 0 \\ \hline \end{array}.$$

False outcome.

$$\begin{array}{|l|r|} \hline \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) & \\ \wedge\ \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s) & : 0.3 \\ \wedge\ \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) & \\ \hline \neg & : 0 \\ \hline \end{array}.$$

Combining the outcomes by cross-sum. The cross-sum conjoins each pair of partition conditions and adds their values: $0.7 + 0.3 = 1.0$, $0.7 + 0 = 0.7$, $0 + 0.3 = 0.3$, and $0 + 0 = 0$.

$$
\begin{array}{|l|}
\hline
\begin{pmatrix} \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \left(\begin{array}{l} \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ \wedge\, \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s) \end{array}\right) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \end{pmatrix} : 1.0 \\
\wedge \left(\begin{array}{l} \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \end{array}\right) \\
\hline
\begin{pmatrix} \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \left(\begin{array}{l} \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ \wedge\, \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s) \end{array}\right) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \end{pmatrix} : 0.7 \\
\wedge \neg \left(\begin{array}{l} \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \end{array}\right) \\
\hline
\neg \begin{pmatrix} \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \left(\begin{array}{l} \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ \wedge\, \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s) \end{array}\right) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \end{pmatrix} : 0.3 \\
\wedge \left(\begin{array}{l} \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \end{array}\right) \\
\hline
\neg \begin{pmatrix} \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \left(\begin{array}{l} \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ \wedge\, \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s) \end{array}\right) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \end{pmatrix} : 0 \\
\wedge \neg \left(\begin{array}{l} \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \end{array}\right) \\
\hline
\end{array}.
$$

The first row is inconsistent because it requires both `k-has-preference` and its negation. The 0.3 row reproduces the original subgoal.

Defining the two outcome conditions. Let

$$
\begin{aligned}
A(s) \equiv\ & \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\
& \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\
& \wedge\, \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s) \\
& \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s), \\
B(s) \equiv\ & \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\
& \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s) \\
& \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s).
\end{aligned}
$$

These are the nonzero conditions of the displayed true- and false-outcome cases, respectively. We simplify their four cross-sum partitions below.

1) $A(s) \wedge B(s)$, with value $0.7 + 0.3 = 1.0$. Combining the conjunctions and removing duplicate fluents gives

$$
\begin{aligned}
& A(s) \wedge B(s) \\
& \quad \equiv \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\
& \qquad \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\
& \qquad \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) \\
& \qquad \wedge\, \neg\texttt{k-not-has-preference}(\texttt{?u}, s) \\
& \qquad \wedge\, \underbrace{\left(\neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \texttt{k-has-preference}(\texttt{?u}, s)\right)}_{\bot} \\
& \quad \equiv \bot.
\end{aligned}
$$

This partition is inconsistent and is removed.

2) $A(s) \wedge \neg B(s)$**, with value** $0.7 + 0 = 0.7$**.** Applying De Morgan's law and distributing $A(s)$ gives

$$\begin{aligned} & A(s) \wedge \neg B(s) \\ \equiv\ & \big(A(s) \wedge \neg \texttt{k-user-to-serve}(\texttt{?u}, s)\big) \\ & \vee \big(A(s) \wedge \neg \texttt{k-at-location}(\texttt{?loc}, s)\big) \\ & \vee \big(A(s) \wedge \neg \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s)\big) \\ & \vee \underbrace{\big(A(s) \wedge \neg \texttt{k-has-preference}(\texttt{?u}, s)\big)}_{A(s)} \\ & \vee \big(A(s) \wedge \neg \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s)\big). \end{aligned}$$

Since $A(s)$ already contains $\neg\texttt{k-has-preference}(\texttt{?u}, s)$, the marked term equals $A(s)$. The absorption law

$$X \vee (X \wedge Y) \equiv X$$

then absorbs every other disjunct, giving

$$A(s) \wedge \neg B(s) \equiv A(s).$$

3) $\neg A(s) \wedge B(s)$**, with value** $0 + 0.3 = 0.3$**.** Applying De Morgan's law and distributing $B(s)$ gives

$$\begin{aligned} & \neg A(s) \wedge B(s) \\ \equiv\ & \big(B(s) \wedge \neg \texttt{k-user-to-serve}(\texttt{?u}, s)\big) \\ & \vee \big(B(s) \wedge \neg \texttt{k-at-location}(\texttt{?loc}, s)\big) \\ & \vee \big(B(s) \wedge \neg \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s)\big) \\ & \vee \underbrace{\big(B(s) \wedge \texttt{k-has-preference}(\texttt{?u}, s)\big)}_{B(s)} \\ & \vee \big(B(s) \wedge \texttt{k-not-has-preference}(\texttt{?u}, s)\big) \\ & \vee \big(B(s) \wedge \neg \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s)\big). \end{aligned}$$

Since $B(s)$ already contains $\texttt{k-has-preference}(\texttt{?u}, s)$, the marked term equals $B(s)$, which absorbs every other disjunct:

$$\neg A(s) \wedge B(s) \equiv B(s).$$

4) $\neg A(s) \wedge \neg B(s)$**, with value** $0 + 0 = 0$**.** By De Morgan's law,

$$\neg A(s) \wedge \neg B(s) \equiv \neg\big(A(s) \vee B(s)\big).$$

This is the complement of the two remaining nonzero conditions, so it is represented by the zero-valued $\neg$ row.

Combining the simplified partitions. The inconsistent first partition is removed, and the remaining partitions give

$A(s)$	: 0.7
$B(s)$	: 0.3
$\neg$	: 0

.

5) No no-op simplification. With no-op simplification, $B(s)$ is discarded because it reproduces the original subgoal, giving the compact statement displayed in Step 3 of Example 4:

$$\Psi_{a_3 \circ a_2 \circ a_1}(s)$$

$$= \begin{array}{|l|r|} \hline \texttt{k-user-to-serve}(\texttt{?u}, s) \wedge \texttt{k-at-location}(\texttt{?loc}, s) \\ \wedge\, \texttt{k-user-at}(\texttt{?u}, \texttt{?loc}, s) \\ \wedge\, \neg\texttt{k-has-preference}(\texttt{?u}, s) \wedge \neg\texttt{k-not-has-preference}(\texttt{?u}, s) \\ \wedge\, \texttt{k-food-at}(\texttt{f-prefer}(\texttt{?u}), \texttt{?loc}, s) & : 0.7 \\ \hline \neg & : 0 \\ \hline \end{array}.$$

B Regression Algorithm

Algorithm 1 describes a systematic procedure for generating the complete set of feasible subgoals, plans, and their probability of reaching a goal state. Starting from the goal formula $\Psi_\emptyset$, the algorithm performs knowledge state regression over a finite horizon h. At each step, it applies the KSR operator and simplifies the resulting case expressions to derive a set of regressed preconditions. The algorithm returns the complete set Π of conditional plans (ϕ, τ, p).

STANDARDIZATION: The standardization operator STANDARDIZE(p) replaces all variables in p with fresh variables $v' \notin \mathcal{V}$ such that:

$$\forall v \in \text{vars}(p), v \notin \text{vars}(\text{STANDARDIZE}(p)).$$

Standardization prevents variable name conflicts when reusing action schemas in a plan.

Algorithm 1 Knowledge State Regression Planner

Input: Goal initiation $\Psi_\emptyset$, Horizon h
$Frontier \leftarrow \{(\Psi_\emptyset, \epsilon, 0)\}$; $\Pi \leftarrow \emptyset$
while $Frontier \neq \emptyset$ **do**
  Extract (Ψ_τ, τ, d) from $Frontier$; $\Pi \leftarrow \Pi \cup \{(\phi, \tau, p) \mid \psi{:}p \in \Psi_\tau,\ p > 0\}$
  if $d = h$ **then**
    continue
  end if
  for each schema $A(\vec{x}) \in \mathcal{A}$ **do**
    $A(\vec{x}) \leftarrow$ STANDARDIZE$(A(\vec{x}))$
    $\Psi_{A\circ\tau} \leftarrow$ SIMPLIFY$(\mathrm{KSR}[\Psi_\tau, A(\vec{x})])$
    $Frontier \leftarrow Frontier \cup \{(\Psi_{A\circ\tau}, A \circ \tau, d{+}1)\}$
  end for
end while
return Π

SIMPLIFY: We simplify the set of regressed formulas by removing any formula that is epistemically inconsistent, i.e., contains more than one mutually exclusive knowledge fluent for the same predicate instance, i.e, known true, known false, and unknown. We also remove any subgoal formula that is entailed by another regressed subgoals, since it is already covered by a shorter plan.

C The PLAN2SENSE Agent

Here are other details about PLAN2SENSE agent's design.

Goal Translation Given a natural-language instruction, the agent first produces a structured goal formula. We use an LLM-based few-shot semantic parser that maps free-form commands to formal language based on methods proposed Xie et al. [2023] with few-shot examples.

Targeted Exploration Because the agent's initial knowledge is incomplete, regressed preconditions frequently contain literals whose truth values are unknown (e.g., whether a container is open, whether an object exists, or where a target item is located). Rather than exploring exhaustively, PLAN2SENSE treats these unknown literals as an *information gap*: the planner explicitly identifies which facts must be resolved to (i) satisfy action preconditions and (ii) guarantee safety constraints (e.g., avoiding irreversible failures). The agent constructs a compact query that includes the current local observations, the active precondition, and the specific unknown predicates to resolve. This query is provided to an LLM policy that selects a logically motivated sensing exploration action. Note that we assume not all sensing actions can be modelled, for example, partial observable room requires the agent to explore the room. However, only knowledge represented in the precondition is relevant; any action that would not lead to this knowledge is not considered.

Knowledge Base Update We keep a structured KB using pydatalog. Each sensing action produces an observation that updates the agent's knowledge base by converting newly observed literals into *known true* or *known false* entries and preserving unobserved literals as *unknown*. After each action, the agent queries the KB with subgoals and will construct a conditional plan if at least one subgoal is satisfiable.

D Example Prompt

The following example prompt specifies the actions available to an embodied agent.

```
"You are an agent in a home environment.",
"Your goal is to complete the natural-language task. You must interact only through the allowed commands.",

"Use physical actions to move/manipulate objects, and use ask_* actions to query for more information.",

"Action guide (choose one Allowed action line exactly):",

"Note: heat/cool/clean directly apply those effects using a device; you do not need extra use/open/close
steps for the effect itself.",

"Note: You can ask for clarification question about the objects to ensure you are targeting the right object

with ask_properties and ask_affordances from the user. Using the wrong object will lead to task failure.",

"- go to <receptacle>: move to that receptacle (changes what you can interact with).",

"- open <receptacle>: open a container so you can access objects inside.",

"- close <receptacle>: close a container.",

"- take <object> from <receptacle>: pick up an object from a receptacle.",

"- move <object> to <receptacle>: place an object into/on a receptacle.",

"- use <object>: to turn on or off an object (task-dependent).",

"- heat <object> with <receptacle>: heat an object using a device (task-dependent).",

"- cool <object> with <receptacle>: cool an object using a device (task-dependent).",

"- clean <object> with <receptacle>: clean an object using a device (task-dependent).",

"- ask_properties <object>: ask whether this specific object is the target.",

"- ask_affordances <object>: ask what object can be used to change properties of this object.",
```

E Plan Selection and Execution

E.1 Plan Selection

After performing Knowledge State Regression, we obtain a set of regressed plans $\Pi = \{(\phi_n(\vec{x}), \tau_n(\vec{x}), p_n)\}_{n=1}^{N}$, where $\phi_n(\vec{x})$ is the condition of a nonzero partition of a regressed subgoal (its predicates without functional terms), $\tau_n(\vec{x})$ is the corresponding lifted linear action sequence parameterized by variables $\vec{x}$, and p_n is its estimated success probability. Let the agent's knowledge be represented by a knowledge base KB. We define a procedure

$$(\tau^*, P^*) = \textsc{Select}(\Pi, KB),$$

which selects an executable lifted linear plan with the highest estimated probability of success, instantiates its variables using a supporting substitution, and returns the instantiated plan τ^* and its estimated success probability P^*. The selection procedure is shown in Algorithm 2.

E.1.1 Linear Candidate Selection

Executable Linear Plans. A lifted linear plan is executable if there exists an instantiation of its variables such that its condition is satisfied by the agent's knowledge base KB. Formally, we define the set of KB-supported candidate plans as

$$\Pi_{\text{exec}}(KB) = \left\{ (\phi_n(\vec{x}), \tau_n(\vec{x}), p_n) \in \Pi \;\middle|\; \exists \theta \text{ s.t. } KB \models \phi_n(\vec{x})\theta \right\}. \tag{1}$$

For each executable lifted plan $(\phi_n(\vec{x}), \tau_n(\vec{x}), p_n) \in \Pi_{\text{exec}}(KB)$, let $\Theta_{\tau_n}(KB) = \{\theta \mid KB \models \phi_n(\vec{x})\theta\}$ denote the set of substitutions that make $\tau_n(\vec{x})$ executable under KB. Together, $\Pi_{\text{exec}}(KB)$ and $\Theta_{\tau_n}(KB)$ define the space of executable lifted linear plans and their valid instantiations from which SELECT chooses.

Selection Procedure. SELECT prioritizes candidate plans by their estimated success probabilities. It ranks plans in $\Pi_{\text{exec}}(KB)$ in descending order of p_n and selects the highest-ranked plan. If multiple plans have the same highest probability, ties are broken arbitrarily. A supporting substitution $\theta^* \in \Theta_{\tau_{n^*}}(KB)$ is then sampled uniformly at random to instantiate the selected plan. Thus, the returned plan and its estimated success probability are

$$\tau^* = \tau_{n^*}(\vec{x})\theta^*, \qquad P^* = p_{n^*} = \max_{(\phi_n, \tau_n, p_n) \in \Pi_{\text{exec}}(KB)} p_n.$$

Algorithm 2 SELECT: Highest-Probability Plan Selection

Input: Candidate plans Π, knowledge base KB
Output: Selected plan τ^* and success probability P^*
$\Pi_{\text{exec}}(KB) \leftarrow \{(\phi_n(\vec{x}), \tau_n(\vec{x}), p_n) \in \Pi \mid \exists\theta \text{ s.t. } KB \models \phi_n(\vec{x})\theta\}$
if $\Pi_{\text{exec}}(KB) = \emptyset$ **then**
    return $(\perp, 0)$
end if
Rank $\Pi_{\text{exec}}(KB)$ in descending order of p_n
Let $(\phi_{n^*}(\vec{x}), \tau_{n^*}(\vec{x}), p_{n^*})$ be the first candidate in the ranking
$\Theta_{\tau_{n^*}}(KB) \leftarrow \{\theta \mid KB \models \phi_{n^*}(\vec{x})\theta\}$
Sample $\theta^* \in \Theta_{\tau_{n^*}}(KB)$ uniformly at random
$\tau^* \leftarrow \tau_{n^*}(\vec{x})\theta^*$
$P^* \leftarrow p_{n^*}$
return (τ^*, P^*)

If $\Pi_{\text{exec}}(KB) = \emptyset$, the procedure returns $(\perp, 0)$, indicating that no candidate plan is executable under the current knowledge base.

E.2 Plan Execution

Given the selected plan and supporting substitution, the lifted sequence is instantiated as

$$\tau^* = \langle A_1(\vec{x}_1)\theta^*, \ldots, A_m(\vec{x}_m)\theta^* \rangle.$$

Object-valued query terms are preserved until their corresponding sensing actions are executed. The returned values are then propagated into the remaining subgoal and action arguments.

Starting from $KB_0 = KB$, the agent executes the selected sequence and updates its knowledge base after each action. Deterministic ontic actions update the knowledge base by progression, while sensing actions incorporate their observed outcomes and any newly acquired object bindings.

Before executing the next action, the agent checks the conditions required by the remaining plan against the updated knowledge base. If an observed outcome invalidates the remaining plan, the agent invokes Knowledge State Regression again to generate and reprioritize candidate plans, and applies SELECT to choose the executable plan with the highest estimated success probability. Execution terminates successfully when the knowledge base satisfies the goal condition.

F Ablation

Veracity of External Query Sources. While we assume the veracity of external query sources, PLAN2SENSE can naturally recover from incorrect sensing outcomes when observations conflict with the agent's knowledge base. In such cases, the agent updates its knowledge base and reinstantiates plans based on the new observations. To illustrate robustness to inaccurate sensing estimates, we include two ablations in which the planner uses (i) uniform probabilities and (ii) reversed probabilities, while the environment follows the true distribution.

Domain	True	Uniform	Reverse
HEAVEN–HELL (10×10)	0.91	0.81	0.67
DOORS-AND-KEYS (10×10)	0.84	0.72	0.59
ALFWORLD-QA	0.89	0.85	0.77

Table 3: Success rates under different sensing-probability estimates. TRUE uses the correct sensing distribution, UNIFORM assumes uniform probabilities, and REVERSE assigns higher probability to less likely outcomes while the environment still follows the true distribution.

REACT and LLM-REFINED PDDL Ablation. To evaluate the benefit of explicit knowledge-state modeling, we compare PLAN2SENSE with REACT and LLM-REFINED PDDL on their ability to acquire necessary information before acting. Specifically, we measure the percentage of trials in which the correct sensing action is executed when sensing is required.

Method	Heaven–Hell	DoorKey	ALFWorld-QA
REACT	23%	41%	18%
LLM-REFINED PDDL	8%	3%	5%
PLAN2SENSE	**95%**	**93%**	**97%**

Table 4: Percentage of trials in which each method executes the correct sensing action when sensing is required. PLAN2SENSE substantially outperforms REACT and LLM-REFINED PDDL, demonstrating the benefit of explicit epistemic-state modeling and sensing-aware planning.

These results show that REACT and LLM-REFINED PDDL often fail to execute the correct sensing action when needed, consistent with our findings in the main results. In contrast, PLAN2SENSE reliably acquires the necessary knowledge before committing to irreversible decisions, demonstrating the benefit of explicit epistemic-state modeling and sensing-aware plans.

G Code of ethics

The research conducted in this paper conforms with the full NeurIPS Code of Ethics.

H Broader Impact

This research involves no human subjects, no human data collection, no sensitive personal data, and no identified dual-use risks. The proposed framework may have positive impacts by enabling more reliable and safer embodied agents through explicit reasoning about uncertainty, task-relevant sensing, and irreversible actions. The paper also discusses the scope and limitations of the approach to prevent overclaiming and discourage misuse beyond the settings in which it has been evaluated.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: Yes. The contributions stated in the abstract and introduction are justified through theoretical and technical explanations in the methodology, experiments, and results

   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 limitation is stated in the "Conclusion and Limitation" section.

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 provides assumptions and derivations in both the methodology section and the appendix.

   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: Yes. The paper includes detailed descriptions of the methodology and experiments required to reproduce the results.

   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: [No]

   Justification: The code is currently unavailable due to intellectual property restrictions that require internal approval. The authors will release the code once this approval process is complete.

   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: The paper focuses on planning and the formalization of knowledge-state planning, rather than training-based learning. The benchmark settings, evaluation protocol, and method details necessary to understand the results are described in the paper.

   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: The paper does not currently report error bars because the evaluation is conducted over fixed benchmark instances with planning-based methods rather than stochastic training runs over multiple random seeds. We instead report aggregate performance across all instances. We have disclosed all the details in the main paper.

   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: answerYes

Justification: The paper reports the computational resources used for the experiments, including the LLM token and runtime information.

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 research conducted in this paper conforms with the Full 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: The paper inlcude a broader impact statement in the Appnedix.

    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 high-risk models such as pre-trained language models, image generators, or scraped datasets.

    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 properly credits the creators and original owners of the datasets, benchmarks, models, and other assets used in the experiments with detailed citations. The corresponding licenses and terms of use respected where applicable.

    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 benchmark extensions used for evaluation. These assets are documented in the paper and appendix.

    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 nor research with human subjects

    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 nor research with human subjects.

    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: [N/A]

    Justification: LLMs are only used for grammar and spelling error checks.

    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.