

Unifying Deep Predicate Invention with Pre-trained Foundation Models

Qianwei Wang^{1,2,*}, Bowen Li^{1,*}, Zhanpeng Luo^{1,3}, Yifan Xu², Alexander Gray⁴, Tom Silver⁵,
Sebastian Scherer¹, Katia Sycara¹, Yaqi Xie^{1,†}

Abstract—Long-horizon robotic tasks are hard due to continuous state-action spaces and sparse feedback. Symbolic world models help by decomposing tasks into discrete predicates that capture object properties and relations. Existing methods learn predicates either top-down, by prompting foundation models without data grounding, or bottom-up, from demonstrations without high-level priors. We introduce UniPred, a bilevel learning framework that unifies both. UniPred uses large language models (LLMs) to propose predicate effect distributions that supervise neural predicate learning from low-level data, while learned feedback iteratively refines the LLM hypotheses. Leveraging strong visual foundation model features, UniPred learns robust predicate classifiers in cluttered scenes. We further propose a predicate evaluation method that supports symbolic models beyond STRIPS assumptions. Across five simulated and one real-robot domains, UniPred achieves 2 ~ 4× higher success rates than top-down methods and 3 ~ 4× faster learning than bottom-up approaches, advancing scalable and flexible symbolic world modeling for robotics. For more information and project materials (video, code...) please visit <https://unipred.github.io/>

Index Terms—Predicate Discovery, Hierarchical Planning, Foundation Models for Manipulation

I. INTRODUCTION

FOR a robot that must make decisions over hours or days, it is neither practical nor necessary to predict every pixel-level detail of world transitions [3], [4], [5]. Instead, it should plan and reason at a higher level of abstraction. Consider the example in Figure 1: when learning to clean up a cluttered table, the exact color or type of each toy is irrelevant to high-level decisions. One thing truly matters is whether the toys have all been packed into the box so that the table is ready to be wiped. By capturing such essential concepts, symbolic world models [6], [7], [8] offer an efficient and generalizable framework for long-horizon tasks, allowing the robot to reason about what needs to be done before deciding how to do it.

Bilevel planning [7], [9], [10], [11], [12], [13], [14], [15], [16] provides a hierarchical framework that exploits the relational structure of symbolic world models for long-horizon, generalizable decision-making. In this paradigm,

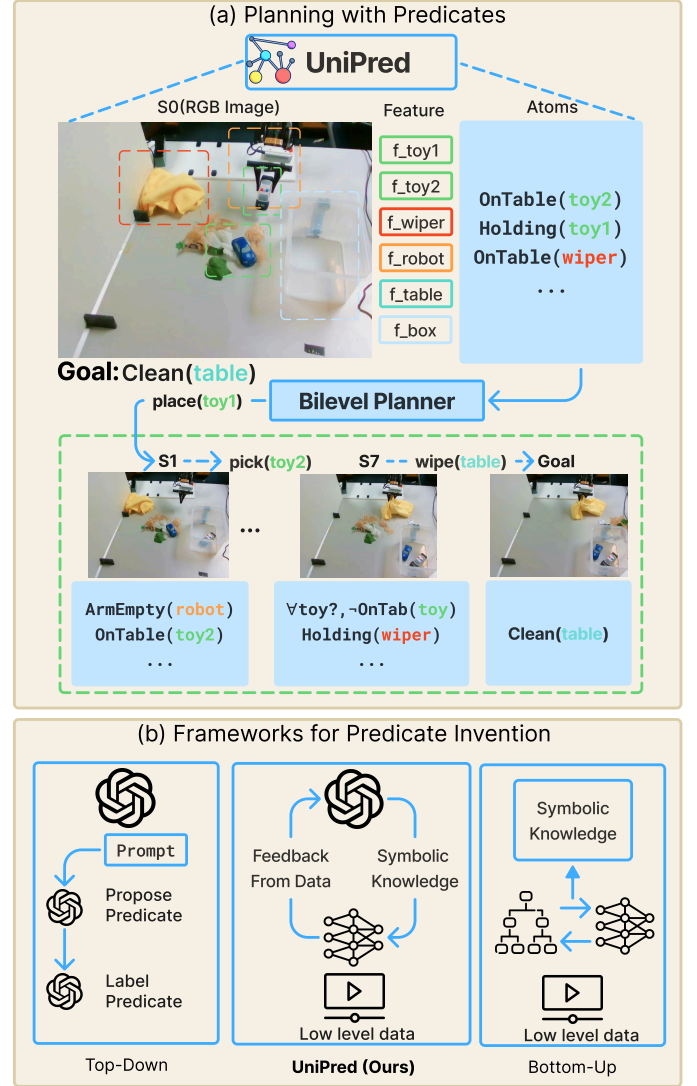


Fig. 1: (a) Planning Framework: Given an initial RGB image, UniPred extracts object-centric features using a foundation model (e.g., DinoV3 [1]) and abstracts them into **invented** ground atoms. These atoms are input into the learned Bilevel Planner to generate a hybrid plan toward the goal. In intermediate steps, UniPred infers corresponding atoms from observed images to facilitate potential replanning. (b) Predicate Invention: UniPred effectively invents predicates by unifying top-down knowledge from the foundation model [2] with bottom-up feedback derived from data.

*Equal Contribution. † Corresponding author.

¹Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. Contact: {bowenli2@andrew.cmu.edu}.

²Computer Science and Engineering Division, University of Michigan.

³Department of Computer Science, University of Pittsburgh.

⁴Centaur AI Institute.

⁵Department of Electrical and Computer Engineering, Princeton University.

The work was partly done when Qianwei Wang and Zhanpeng Luo were Robotics Institute Summer Scholars (RISS) associated with the Advanced Agent-Robotics Technology Lab, CMU.

low-level continuous states (e.g., RGB images) are abstracted into high-level discrete symbols using predicates (e.g., `Clean(?table)`). Planning then proceeds jointly in the

symbolic space—deciding what to do—and the continuous space—deciding how to do it. By leveraging AI planning principles [17], [18], such a bilevel structure achieves greater efficiency and generalization than planning solely in the low-level space. Nonetheless, most classical planning systems [19], [20] require exhaustive manual engineering of the symbolic model, which largely limits their scalability. To address this problem, recent work has explored how to learn the abstractions required for bilevel planning—such as predicates [11], [15], [16], PDDL operators [9], [13], samplers [14], and skills [10]—directly from demonstrations.

Among these abstractions, *predicates* form the core component. As shown in Figure 1 (a), at the high level, they define the preconditions and effects of planning operators, providing a relational reasoning space in which an AI planner can generate novel solutions to unseen problems [9]. At the low level, predicates specify the constraints that samplers must satisfy [21] and determine the initiation and termination conditions of motor skills [22]. Broadly, two paradigms exist for learning symbolic predicates and the associated classifiers. Top-down approaches [12], [23], [24] generate predicates in a zero-shot manner using pre-trained foundation models—such as large language models (LLMs) [2], [25] and visual foundation models (VFM) [26], [1]—leveraging their embedded common-sense priors. However, because these models are trained on broad, general-purpose knowledge, their zero-shot application to specialized robotic domains typically requires carefully engineered, domain-specific prompts [27], [28], [29], [30]. Moreover, purely foundation-model-driven methods [12], [23] often struggle in cluttered scenes because their predicate proposal and classification pipelines rely entirely on test-time prompts, making them unstable to scene variability. Bottom-up approaches [11], [16], [5], [31], in contrast, directly optimize predicates from low-level data without relying on priors from foundation models. They are also generally applicable to diverse and noisy states, such as cluttered tabletop scenes in Figure 1, but suffer from low efficiency due to the combinatorial complexity of the predicate search space [3], [16].

To unify these two families of approaches, we present a highly integrated bilevel learning framework, UniPred, for deep predicate invention. Since LLMs are often unreliable without carefully engineered prompts [28], we use them only as coarse top-down guidance, with low-level learning providing bottom-up feedback. Specifically, UniPred first constructs a candidate pool of predicates: the LLM proposes rough predicate–effect distributions, which are used to derive pseudo-labels for training neural predicate classifiers on demonstration data [16]. The classifier’s training loss then serves as feedback that informs the LLM about potential inconsistencies and steers its subsequent proposals toward more plausible hypotheses. Through this iterative querying process, UniPred efficiently generates the predicate candidate set without relying on domain-specific prompt engineering. Building on top of this bilevel learning framework, UniPred further propose to *optimize* predicate classifiers on the strong feature representations from the recent VFMs [1] in image-based domains, yielding robust predicate grounding over cluttered real images. In the second predicate selection

stage, prior approaches [12], [23], [16], [11] often fail to distinguish between basic predicates directly grounded by (neural) classifiers (e.g., `OnTable(?toy)`) and the derived ones (e.g., `∃?toy, OnTable(?toy)`), leading to unreliable selection in domains with Non-STRIPS operators [32] such as `WipeTable`. UniPred addresses this by adding only basic predicates in operator effects and treating derived predicates as derived relational constraints when computing planning-driven predicate scores, resulting in robust symbolic world model in complex, Non-STRIPS domains.

We conduct comprehensive evaluation of UniPred on 6 robot planning domains with various state representations. Thanks to the discovered relational symbolic world model, UniPred generalizes significantly better than various model-free baselines. Compared with purely top-down methods [23], [12], UniPred is free of carefully engineered prompts and flexibly learns more effective predicates that ground in various state representations. Compared with purely bottom-up methods [16], [11], UniPred achieves $3 \sim 4\times$ speed up with perceptually generalizable predicates on real-world images. We summarize our key contributions as follows:

- **Foundation Model-based Unified Bilevel Learning:** UniPred introduces an iterative framework that queries LLMs with feedback from neural predicate classifiers, thereby avoiding reliance on domain-specific prompt engineering and substantially improving learning efficiency.
- **Derived-aware Predicate Selection:** UniPred differentiates between basic and derived predicates during predicate selection, treating derived predicates as derived relational constraints for predicate evaluation, enabling effective world modeling in Non-STRIPS domains.
- **Comprehensive Evaluation:** We evaluate UniPred on 6 domains, including 5 simulated domains with diverse state representations (pose, image, and point cloud) and 1 real-world Table Clean domain with 10 tasks, demonstrating strong generalization and efficiency.

II. RELATED WORK

A. Foundation Models for Robot Planning

Recent literature increasingly positions foundation models as integral building blocks for robot planning. At the motion level, vision-language-action (VLA) models trained on extensive collections of robot trajectories map visual and linguistic inputs directly to joint-level actions. These models frequently serve as general-purpose skill primitives, operating in conjunction with classical motion planners for local control or waypoint following [33], [34], [35], [36], [37]. At the task level, large language models (LLMs) function as high-level planners that translate natural language goals into structured subtask sequences, executable code, or symbolic planning operators. Notable examples include systems that ground language in robot affordances, learn PDDL-style domains from feedback, or generate robot control code and task graphs from instructions [24], [38], [39], [40], [41]. Concurrently, visual foundation models, such as CLIP and large multimodal transformers, enable open-vocabulary scene understanding and affordance prediction. These models are employed to construct

symbolic state descriptions, semantic cost maps, or object-centric features for consumption by downstream planners [26], [1], [42], [43]. Collectively, these research streams suggest a paradigm in which foundation models supply semantic priors, reusable skills, and scene abstractions. In the context of task and motion planning (TAMP), prior works have explored the utility of foundation models for specifying subgoals [44], formulating constraints [45], and grounding semantic state abstractions [23]. In this work, we investigate how foundation models can be most effectively leveraged to discover predicates for bilevel planning [23], [11], [16].

B. Learning for Bilevel Planning

Classical TAMP methods [19], [21], [20] typically rely on extensive, hand-engineered planning models for specific domains. To mitigate this manual engineering burden, a growing body of work has adopted modern learning techniques to acquire the components required for TAMP. In this work, we build upon recent research regarding learning for bilevel planning—a prominent framework for solving TAMP problems. In the bilevel planning paradigm, the system can alternate between a high-level task planner and a low-level sampler. Given a high-level task plan (determining *what* to do), the sampler attempts to determine the continuous parameters (determining *how* to do it) that satisfy the requisite geometric constraints. Prior literature has investigated learning planning operators [7], [9], [13], samplers [14], parametrized skill policies [10], and relational predicates [11], [12], [16], [23]. Learning these structural representations offers two primary advantages in planning: (1) relational representations enable the system to better generalize to compositionally novel problems; and (2) task decomposition renders long-horizon planning significantly more efficient compared to flat policy learning [46] or planning directly in the low-level state space. Among these abstractions, discovering a rich and expressive set of predicates for low-level states remains a fundamentally challenging problem [16], [3]. In UniPred, we present a unified framework demonstrating that the prior knowledge and robust representations derived from foundation models [2], [1] can efficiently and effectively address this challenge.

C. Predicate Discovery for Planning

In planning contexts, predicates abstract low-level continuous states (e.g., object poses, point clouds, and images) into high-level discrete states represented by logical atoms (e.g., $\text{OnTable}(\text{?toy})$, $\text{InBox}(\text{?toy})$). These abstractions serve two primary functions: (1) enabling AI planners [17] to efficiently search the discrete state space to generate task plans for novel object compositions; and (2) defining the geometric constraints that samplers must satisfy to generate continuous parameters at each planning step.

Recent approaches to predicate discovery fall broadly into top-down and bottom-up categories. Top-down methods [12], [23], [24] leverage large, pre-trained foundation models to propose predicates in a zero-shot manner. While benefiting from strong commonsense priors and data efficiency, these approaches often prove unreliable in specialized domains due to

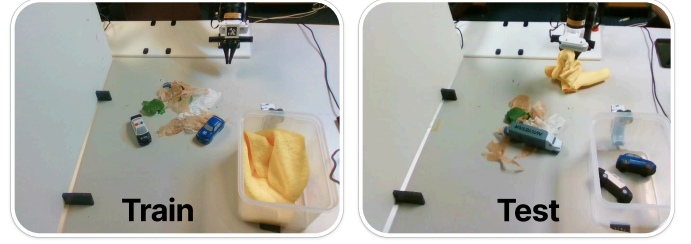


Fig. 2: An example of train and test tasks in the table-cleaning domain. During training, both of the two toys are initially on table and the wiper is in the box. During test, the system needs to plan with more toys whose configurations are randomly initialized.

the general nature of internet-scale pre-training. Consequently, they are sensitive to prompt design and may fail to generate complete predicate sets for unfamiliar tasks [23]. Moreover, reliance on frozen, prompt-based classifiers can be brittle when applied to cluttered or unseen states [23], [12].

Conversely, bottom-up approaches [3], [5], [11], [16] optimize directly within the complete predicate space using only low-level data. These methods are typically prompt-free, offer greater reliability in specialized domains, and adapt better to noisy observations. However, they face combinatorial scaling challenges and are often inefficient in high-dimensional spaces like RGB images. Grammar-based methods [11] are restricted to simple states with explicit relational structures. A recent bilevel learning approach, IVNTR [16], is able to optimize neural classifiers for predicates based on effect-centric objective. Though flexible, this approach has two key limitations: (1) Since the potential predicate space is combinatorially large, learning from scratch becomes extremely slow for complicated domains. (2) It is limited to domains where all of the operators follow STRIPS assumption.

To address these limitations and bridge the gap, we introduce UniPred, a bilevel learning framework that synergizes the prior knowledge of foundation models with domain-specific low-level feedback. By closing the loop between high-level reasoning and data-driven learning, UniPred efficiently explores the predicate space without extensive prompt engineering. Furthermore, our derived-aware selection procedure explicitly distinguishes between basic and derived predicates during operator model construction. We show that UniPred efficiently yields expressive, planner-friendly predicates, enabling robust bilevel planning across diverse and complex simulated and real-robot domains.

III. PROBLEM FORMULATION

We address the problem of learning planning *abstractions* from offline demonstration datasets. UniPred is designed to (1) generalize to test tasks involving unseen object compositions and (2) maintain computational efficiency during training. In this section, we formalize the planning domain, action space, predicates, data structure, and computational objective. Our notation follows [11], [16] and is summarized in Table I.

We define a *planning domain* as a tuple $\mathcal{D} = \langle \Lambda, \mathcal{C}, f, \Psi_G, \Psi_{\text{sta}} \rangle$ associated with a task distribution \mathcal{T} . A specific *planning task* is sampled as $T = \langle \mathcal{O}, \mathbf{x}_0, g \rangle \sim \mathcal{T}$.

Here, Λ denotes a finite set of object *types* $\lambda \in \Lambda$, and $\mathcal{O} = \{o_1, \dots, o_N\}$ is a set of N objects, each assigned a type from Λ . The task tuple includes the initial continuous state \mathbf{x}_0 and a set of ground goal predicates $g \subseteq \Psi_G$ to be satisfied upon completion. A state $\mathbf{x} \in \mathcal{X}$ assigns continuous features to every object in \mathcal{O} . For notational simplicity, we represent the state of N objects as a matrix $\mathbf{x} \in \mathbb{R}^{N \times K}$, where each row encodes K domain-specific features (e.g., pose, gripper status, or table dimensions), consistent with [11], [16]. In our experiments, we extend this formulation to richer object-centric representations, such as point clouds and RGB images, by treating their encodings as per-object features.

The action space comprises a finite set of M *parameterized controllers*, denoted $\mathcal{C} = \{C_1, \dots, C_M\}$. Each controller $C \in \mathcal{C}$ is defined by a typed signature $(\lambda_1, \dots, \lambda_v)$ over object types in Λ and a continuous parameter space Ω . A *grounded* controller instance, $\underline{C} = C(o_{i_1}, \dots, o_{i_v}, \omega)$, is obtained by binding all object arguments and selecting a parameter $\omega \in \Omega$. States and actions are linked via a known transition function $f(\mathbf{x}, \underline{C}) \mapsto \mathbf{x}'$. A *plan* for task T is a sequence of grounded controllers $\pi = [\underline{C}_1, \dots, \underline{C}_H]$. A plan is valid if recursively applying the transition model $\mathbf{x}_i = f(\mathbf{x}_{i-1}, \underline{C}_i)$ starting from \mathbf{x}_0 yields a terminal state \mathbf{x}_H that satisfies the goal set g .

We use a table clean domain as a running example (see Figure 2). This domain includes five object types: $\Lambda = \{\text{robot}, \text{toy}, \text{box}, \text{wiper}, \text{table}\}$. The objective is for the robot to pick all toy objects from the table, place them into a designated box, and subsequently wipe the table surface. The controller set for this domain includes primitives such as `PickToyFromTable(?r, ?to, ?ta)`, `PlaceToyInBox(?r, ?t, ?b)`, `PushBox(?r, ?b)`, and `WipeTable(?r, ?w, ?t)`. These controllers encompass skills derived from motion planning [11], [16] (e.g., pick and place) and imitation learning [46], [47] (e.g., `PushBox`). We provide further implementation details in the Section V.

To facilitate efficient planning, bilevel methods use a predicate set Ψ to decompose long-horizon decision problems. A *lifted predicate* is defined by a typed signature over variables, $\psi(? \lambda_1, \dots, ? \lambda_u)$, and an associated classifier $\theta_\psi : \mathcal{X} \times \mathcal{O}^u \rightarrow \{\text{True}, \text{False}\}$. A *ground predicate* $\underline{\psi}$ is formed by instantiating all variables with specific objects from \mathcal{O} ; for brevity, we define $\theta_\psi(\mathbf{x}) \triangleq \theta_\psi(\mathbf{x}, (o_1, \dots, o_u))$. This classifier is used to determine whether the ground predicate holds in state \mathbf{x} for the specific object tuple.

Following [11], [16], we categorize predicates into three subsets: (i) Ψ_{sta} , static predicates with truth values that remain fixed throughout the task, depending solely on time-invariant object attributes (e.g., `ColorRed(?toy)` or `SizeSmall(?toy)`); (ii) Ψ_{dyn} , dynamic predicates whose truth values may change during execution (e.g., `InBox(?toy, ?box)` or `OnTable(?toy, ?table)`); and (iii) Ψ_G , goal predicates. We further partition dynamic predicates into *basic* dynamic predicates, Ψ_{dyn}^b , which are evaluated directly from object-centric features (e.g., `OnTable(?toy, ?table)`), and *derived* dynamic predicates, Ψ_{dyn}^d , which are computed from basic predicates via quantifiers. For instance, the predicate $\forall ? \text{toy}. \neg \text{OnTable}(\text{?toy}, \text{?table})$ signifies that no toys

remain on the table. We assume Ψ_G and Ψ_{sta} are provided, whereas the dynamic predicate set Ψ_{dyn} may be expanded by the learning algorithm, as in [16].

During training, the robot utilizes an offline demonstration dataset of B task-plan pairs, $\mathcal{D} = \{(T_b, \pi_b)\}_{b=1}^B$, where $T^b = \langle \mathcal{O}^b, \mathbf{x}_0^b, g^b \rangle \sim \mathcal{T}^{\text{train}}$. Since f is known and deterministic, we derive the full state-action trajectory $(\mathbf{x}_0^b, \underline{C}_1^b, \mathbf{x}_1^b, \dots)$ for each instance by simulating π_b . At test time, the agent encounters tasks $T = \langle \mathcal{O}, \mathbf{x}_0, g \rangle \sim \mathcal{T}^{\text{test}}$ characterized by unseen object configurations, larger object sets, and more diverse action compositions than those in $\mathcal{T}^{\text{train}}$. As illustrated in Figure 2, training demonstrations consistently initialize with the robot hand empty, two toys on the table, and the wiper inside the box. Conversely, test tasks may initialize with the robot holding the wiper and involve three toys in varied locations, such as inside the box. These variations alter both the requisite plan length and the sequencing of controllers. Consequently, success necessitates abstractions that capture reusable structure rather than simply memorizing demonstration trajectories.

IV. METHODOLOGY

A. Bilevel Planning with Predicates

We provide a brief review of bilevel planning and refer the reader to existing literature for a comprehensive discussion [9], [10], [11], [13], [14], [15], [16], [20]. Figure 1a illustrates an instance of our setting within the table-cleaning domain.

Bilevel planning leverages relational abstractions to achieve sequential and compositional generalization. The framework relies on two primary abstractions: *predicates* (state abstractions) and *operators* (action abstractions). Furthermore, bilevel planning employs relational *samplers* to generate continuous parameters for controllers.

Definition 1 (Operator): The *operator* for a parameterized controller C is a tuple $\text{Op}^C = \langle \text{Var}, \text{Pre}, \text{Eff}^+, \text{Eff}^- \rangle$, where Var is a tuple of object placeholders matching the type signature of C , and $\text{Pre}, \text{Eff}^+, \text{Eff}^- \subseteq \Psi$ are sets of lifted predicates defined over variables in Var , representing *preconditions*, *add effects*, and *delete effects*, respectively. Here, Ψ denotes the predicate set.

As an illustrative example, the operator for `PickToyFromTable(?r, ?t)` may be defined as:

$$\begin{aligned} \text{Pre} &= \{\text{HandEmpty}(\text{?r}), \text{ToyOnTable}(\text{?t}, \text{?t})\}, \\ \text{Eff}^+ &= \{\text{Holding}(\text{?r}, \text{?t})\}, \\ \text{Eff}^- &= \{\text{HandEmpty}(\text{?r}), \text{ToyOnTable}(\text{?t}, \text{?t})\}. \end{aligned}$$

Given a task $T = \langle \mathcal{O}, \mathbf{x}_0, g \rangle$, the bilevel planning process (see Figure 1a) initiates by constructing an *abstract state* comprising all ground predicates over objects \mathcal{O} satisfied in \mathbf{x}_0 . These initial ground predicates, alongside the operator set and goal g , are input to a classical AI planner [17] to synthesize a plan *skeleton* $\bar{\pi}$. This skeleton consists of a sequence of partially grounded actions \underline{C} with unspecified continuous parameters. To refine the actions in this skeleton $\underline{C} \in \bar{\pi}$ into fully grounded actions \underline{C} equipped with continuous parameters ω , bilevel planning utilizes *samplers*.

Definition 2 (Sampler): The *sampler* η^C for a planning operator Op^C with v object placeholders is a conditional distribution

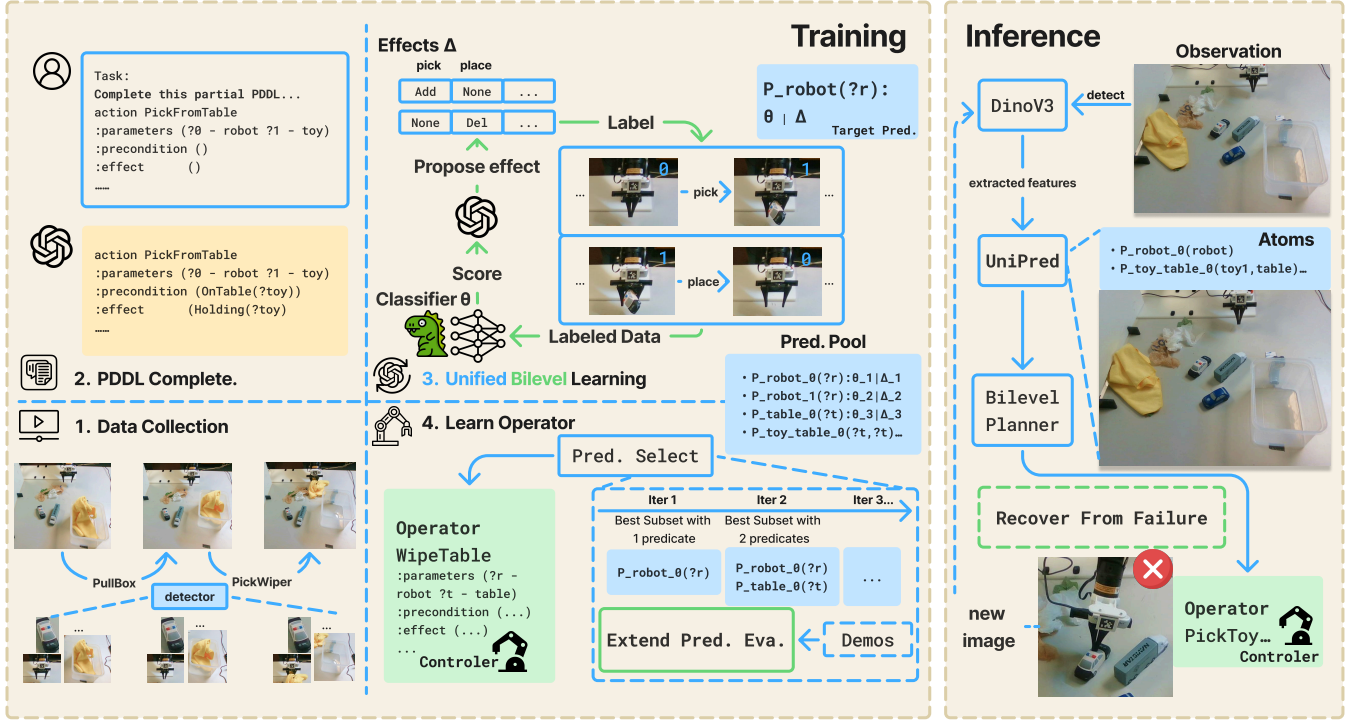


Fig. 3: Overview of the UniPred Framework: Training and Inference. The framework consists of two main phases. Training (Left) utilizes an offline demonstration dataset, where we first prompt a Large Language Model (LLM) to complete a partial PDDL domain specification. This is followed by an LLM-in-the-loop bilevel learning framework designed to construct a comprehensive set of predicate candidates, which are then refined through a final down-selection step. During Inference (Right), UniPred processes input images by extracting robust visual features from object-centric regions using a VFM [1], which are converted to the initial set of ground predicates. Based on the predicates, the learned bilevel planner then generates the initial execution plan and optionally replans when skill execution failures are observed.

$\omega \sim \eta^c(\cdot \mid \mathbf{x}, (o_1, \dots, o_v))$ that proposes continuous action parameters for $C((o_1, \dots, o_v), \cdot)$ given a state \mathbf{x} .

In contrast to deterministic operators, samplers are typically stochastic and may fail to yield valid parameters. Consequently, bilevel planning alternates between the AI planner and the samplers, using the predicate classifiers θ_ψ to guide the search and compensate for potential sampling failures.

Assuming a sufficiently expressive predicate set, prior work has demonstrated the learnability of *operators* [7] and *samplers* [10], [14] from demonstration data \mathcal{D} . In this context, predicates fulfill two critical functions. First, the *relational* nature of predicates, learned operators, and samplers enables strong generalization to held-out test tasks ($\mathcal{T}^{\text{test}}$) involving varying object sets and configurations. Second, informative *dynamic* predicates structure the abstract state space, allowing the planner to efficiently search the space of operator sequences, as depicted in Figure 1a. Conversely, an impoverished predicate set—containing, for instance, only Ψ_G and Ψ_{sta} —can render bilevel planning computationally intractable [11]. We next introduce the UniPred framework, which addresses this challenge by automatically inventing dynamic predicates to facilitate efficient bilevel planning.

B. UniPred Overview

Figure 3 provides an overview of the UniPred framework. The stages of the pipeline are detailed below.

a) Stage one: Data collection and preparation: The first stage focuses on data collection and preprocessing before the predicate learning. Our approach relies on object-centric states (in image domains, we use an off-the-shelf object detector [48] to obtain object-centric image patch as the states) and labeled state-transition pairs $(\mathbf{x}, \mathcal{C}, \mathbf{x}')$.

b) Stage two and three: Learning a pool of basic dynamic predicates: The core innovation in these stages lies in integrating low-level data feedback with the reasoning capabilities of large foundation models to efficiently explore dynamic predicates. Specifically, UniPred utilizes foundation models to propose candidate symbolic relations, which are instantiated as “seed” predicates. We then employ effect-based supervision [16] to train classifiers for these predicates, iteratively using low-level learning feedback to guide the foundation model. This closed-loop interaction between low-level supervision and high-level proposals significantly enhances exploration efficiency. Furthermore, this bilevel learning system enables UniPred to optimize predicates derived from recent VFMs [1], yielding perceptually generalizable predicates with only approximately 20 demonstration trajectories in the table-cleaning domain. This procedure is detailed in Section IV-C.

c) Stage four: Derived-aware predicate selection and operator learning: Once the candidate predicate pool is constructed, identifying the subset necessary for efficient planning is crucial. Consistent with prior work on predicate selection [11], [16], [12], we perform a hill-climbing search to optimize a planning-driven objective. However, we observe

TABLE I: Important notations used in this work.

Symbol	Meaning	Space
Λ	Type set in the domain	Set
λ	A type in the domain	Symbol
$? \lambda$	Typed variable	Symbol
T	Planning task	Tuple
\mathcal{T}	Task distribution	Distribution
K	Feature dimension per object	Scalar
N	Number of objects in a task	Scalar
\mathbf{x}_i	The i th state	Matrix
\mathcal{O}	Object set in a task	Set
\mathcal{C}	Controller or action set	Set
\mathcal{C}	Parameterized controller	Symbol
M	Number of controllers	Scalar
Ω	Continuous parameter space	Set
ω	Continuous action parameter	Vector
$\underline{\mathcal{C}}$	Grounded controller instance	Symbol
f	Transition function	Function
ψ	Lifted predicate	Symbol
$\underline{\psi}$	Ground predicate	Symbol
θ_{ψ}	Classifier for predicate ψ	Function
Ψ	Complete predicate set	Set
Ψ_{sta}	Static predicate set	Set
Ψ_{dyn}	Dynamic predicate set	Set
Ψ_{dyn}^b	Basic dynamic predicate set	Set
Ψ_{dyn}^d	Derived dynamic predicate set	Set
Ψ_G	Goal predicate set	Set
π	Refined plan	List
$\bar{\pi}$	Plan skeleton	List
\mathcal{D}	Offline demonstration data set	Set
B	Number of task plan pairs	Scalar
$\text{Op}^{\mathcal{C}}$	Operator for controller \mathcal{C}	Set
Var	Variable set	List
Pre	Precondition set	Set
Eff^+	Add effect set	Set
Eff^-	Delete effect set	Set
$\eta^{\mathcal{C}}$	Sampler for controller \mathcal{C}	Function
$\hat{\Psi}_{dyn}$	Candidate dynamic predicate set	Set
$J(\cdot)$	Score function based on planning outcome	Function
Δ^{ψ}	Effect vector for predicate ψ	Vector
$\mathbf{t}^{\psi, \mathcal{C}}$	Predicted predicate transition	Vector
\mathbf{L}_t	Loss information at iteration t	Vector
\mathbf{R}_t	Global value vector at iteration t	Vector
τ	Total learning time	Scalar
τ_{predinv}	Predicate invention time	Scalar
τ_{predsel}	Predicate selection time	Scalar
τ_{skill}	Skill and sampler learning time	Scalar

that derived predicates often violate the STRIPS assumption despite their utility in planning. To address this, unlike previous approaches, UniPred distinguishes between basic dynamic predicates Ψ_{dyn}^b and derived dynamic predicates Ψ_{dyn}^d during evaluation. We demonstrate that UniPred learns expressive symbolic world models capable of capturing global concepts in complex environments, such as the cluttered table-cleaning scenes depicted in Figures 1a and 3. Section IV-D formally describes our derived-aware predicate selection procedure.

Finally, using the complete set of predicates, UniPred learns operators $\text{Op}^{\mathcal{C}}$ and samplers $\eta^{\mathcal{C}}$ following the standard paradigm [7], [10], [14], as detailed in Section IV-E.

d) *Inference and closed-loop planning*: Upon completion of training, UniPred is deployed for test-time planning. We illustrate this process using the image-based table-cleaning

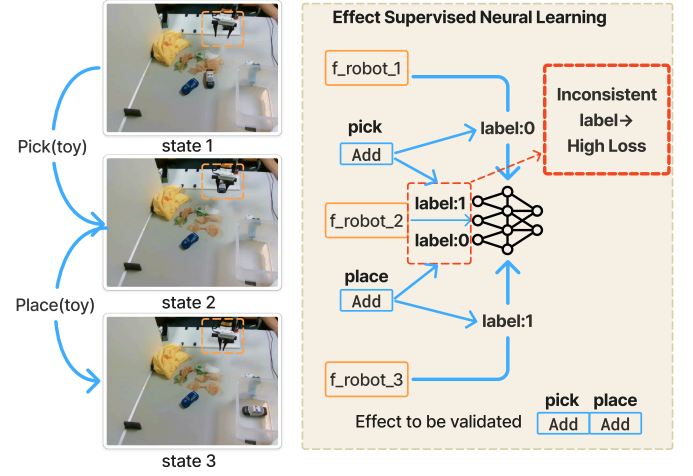


Fig. 4: Effect supervised training process for the classifier of predicate $p_{robot}(\text{?robot})$. The left panel shows three consecutive states during a pick and place sequence, and the right panel illustrates how an incorrect effect specification leads to inconsistent labels across these states, which increases the training and validation loss.

domain (Figure 3). A raw RGB observation is processed by a detector and a visual foundation model (DINOv3) to produce an object-centric state \mathbf{x} , comprising one feature vector per object. This representation facilitates the invention of robust predicates from limited training trajectories. The object-centric state is processed by the learned classifiers to generate a set of ground atoms describing the current scene. These atoms, together with the learned operators and the goal set Ψ_G , are fed into a classical planner to generate a plan skeleton $\bar{\pi}$. UniPred then refines the first abstract action in $\bar{\pi}$ into a grounded controller using the learned sampler or closed-loop skill and executes it on the robot.

Execution proceeds in a closed loop. After each controller execution, a new observation is obtained, and the object-centric state and atoms are updated to verify the applicability of the subsequent abstract action. If an action fails—for example, if a `PickToyFromTable` controller fails to grasp the object as shown in Figure 3—the updated atoms reflect this state change. UniPred then triggers the planner to replan from the current configuration. This cycle continues until the goal predicates in Ψ_G are satisfied or the planning budget is exhausted. In this manner, the learned predicates, operators, samplers, and closed-loop skills synergize to enable robust bilevel planning and recovery in complex domains.

C. Foundation Model-based Predicate Candidate Learning

In this section, we detail how UniPred effectively leverages foundation models to learn neural classifiers for candidate dynamic predicates. A primary challenge in predicate invention is the system’s lack of ground truth supervision labels for ground atoms. To address this, UniPred builds upon the effect-based classifier learning framework proposed in [16], which we briefly review below.

a) Effect-based Supervision on Predicate Classifiers:

Let us assume the effect distribution of a predicate ψ within operator effects is known—specifically, whether ψ is added

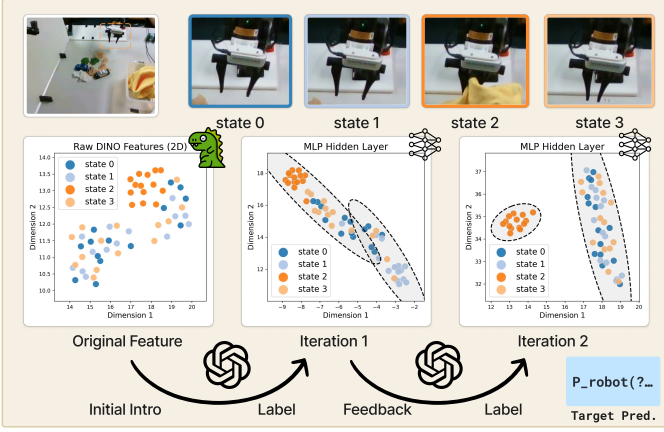


Fig. 5: Overview of unified bilevel learning. The top row depicts four image states from the table-cleaning task: the robot hand is empty in states 0, 1, and 3, while it holds a towel in state 2. The bottom row visualizes the evolution of DINO image features through the MLP classifier. Left: The original 2D projection, where the four states are not clearly separable. Middle (Iteration 1): After the LLM proposes symbolic knowledge and generates training labels, the hidden features begin to exhibit structure, though state 2 remains partially entangled. Right (Iteration 2): The task loss is fed back to the LLM, prompting an update to the predicate specification and labels; this yields a refined hidden representation where state 2 is cleanly separated from the empty-hand states.

or deleted by each controller class $C_i \in \mathcal{C}$. We encode this information in a lifted effect vector $\Delta^\psi = [\delta_1^\psi, \dots, \delta_M^\psi] \in \{-1, 0, 1\}^M$, where each entry indicates whether ψ constitutes an add effect, a delete effect, or remains unaffected by controller C_i ¹. Given a transition pair $(\mathbf{x}, C_i, \mathbf{x}')$, the lifted effect vector is instantiated as a ground effect vector $\mathbf{t}^{\psi, \mathcal{C}_i}$. This vector specifies for each ground atom ψ over the task objects \mathcal{O} whether its value should transition or remain constant following the application of C_i .

As illustrated in Figure 4, this effect information provides critical supervision for training the classifier θ_ψ . For a lifted predicate ψ absent from the effect set of an operator $\text{Op}_{C_i}^{\psi}$, the truth values of all its ground atoms must remain invariant across transitions induced by any ground controller C_i . Conversely, if ψ appears in the effect set of $\text{Op}_{C_i}^{\psi}$, the pre- and post-transition truth values of its ground atoms must reflect the add or delete effects specified by Δ^ψ . Aggregating these constraints over all transitions and controllers $C \in \mathcal{C}$ yields a differentiable loss $\mathcal{L}^\mathcal{D}(\theta_\psi)$. This loss is minimized via standard deep learning optimizers to train the neural classifier θ_ψ directly from object-centric features, eliminating the need for ground truth labels.

This bottom-up, effect-based learning scheme offers a significant advantage over purely top-down approaches [12], [23] that rely on zero-shot, prompting-based foundation models for predicate classification. Since foundation models are pre-trained on broad, internet-scale data, their zero-shot classification performance often degrades in specialized or custom domains. In contrast, the neural predicates θ_ψ in [16] and

UniPred are optimized directly against low-level state transitions. This results in predicates that are better aligned with complex environmental states, such as those encountered in cluttered table-cleaning scenarios.

However, identifying the correct effects Δ^ψ in prior work [16] requires a tree expansion algorithm guided by neural learning feedback. The search space for this method expands exponentially with the number of controllers and objects [3]. To overcome this efficiency bottleneck, we introduce the LLM-in-the-loop bilevel learning algorithm employed by UniPred.

b) *Unified FM-Bilevel Learning Framework*: Drawing on the complementary strengths of top-down [12], [23] and bottom-up approaches [16], as well as recent advances utilizing foundation models as optimizers to refine their own outputs [49], [50], we propose a unified predicate learning framework shown in Figure 3 and Figure 5. In UniPred, foundation models serve three distinct roles: (1) initializing the effect distribution by completing the partial domain PDDL; (2) operating in-the-loop with effect-based supervision to generate a consistent pool of dynamic predicates; and (3) In image-based domains, providing image-based feature encodings for the learnable predicate classifier θ_ψ .

First, we represent the set of controllers \mathcal{C} and the limited set of given predicates $\{\Psi_{\text{sta}}, \Psi_G\}$ as a structured PDDL [51] domain definition. We use this partial domain, supplemented by a small number of demonstrations from \mathcal{D} , to prompt an LLM for completion. From the completed PDDL, we extract newly proposed predicates and their symbolic effect distributions. For each candidate, we instantiate a neural classifier θ_ψ , train it using the aforementioned effect-based supervision, and record the resulting effect distribution and learning feedback in the exploration history. These “seed” predicates encode a prior over the domain structure, enabling the LLM to explore potentially consistent effects more efficiently than the *ab initio* search required in [16].

Second, given that initial predicate proposals are typically incomplete, UniPred employs the LLM as a proposer and the classifier learning process as an evaluator. For each initially proposed predicate ψ , we query the language model to suggest possible effect patterns based on the current learning history. UniPred instantiates each proposal as a neural classifier θ_ψ , trained via the effect-based loss $\mathcal{L}^\mathcal{D}(\theta_\psi)$. The resulting effect vector and learning feedback are then appended to the history to inform the subsequent iteration of LLM proposals.

Formally, for a single transition pair $(\mathbf{x}, C, \mathbf{x}')$, we apply θ_ψ to all possible groundings of ψ over the object set \mathcal{O} to obtain:

$$\hat{\mathbf{v}}, \hat{\mathbf{v}}' = \text{Ground}(\mathbf{x}, \theta_\psi), \text{Ground}(\mathbf{x}', \theta_\psi) \in [0, 1]^P,$$

where P denotes the number of ground atoms of ψ . Let $\mathbf{t}^{\psi, \mathcal{C}} \in \{-1, 0, 1\}^P$ represent the ground effect vector derived from the lifted effect vector Δ^ψ for controller class C . We define masks to identify unaffected and affected ground atoms,

$$\mathbf{m}^0 = \mathbb{I}(\mathbf{t}^{\psi, \mathcal{C}} = 0), \quad \mathbf{m}^1 = \mathbb{I}(|\mathbf{t}^{\psi, \mathcal{C}}| = 1),$$

and let $S^1 = \{p : m_p^1 = 1\}$ be the set of indices corresponding to affected atoms. The per-transition loss is defined as:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}', \theta_\psi) = \mathcal{L}_{\text{zero}} + \mathcal{L}_{\text{one}},$$

¹Following [16], we assume: (1) each predicate appears at most once in an operator’s effect set, and (2) ground operators modify only ground predicates sharing the same or a subset of parameters.

where

$$\mathcal{L}_{\text{zero}} = \text{JS}(\hat{\mathbf{v}} \odot \mathbf{m}^0 \parallel \hat{\mathbf{v}}' \odot \mathbf{m}^0)$$

penalizes deviations in atoms that should remain constant, and

$$\mathcal{L}_{\text{one}} = \frac{1}{|S^1|} \sum_{p \in S^1} \text{BCE}([\hat{v}_p, \hat{v}'_p], [\frac{1-t_p}{2}, \frac{1+t_p}{2}])$$

penalizes violations of the add ($t_p = 1$) or delete ($t_p = -1$) effects on affected atoms. Here, $\text{JS}(\cdot \parallel \cdot)$ denotes the Jensen-Shannon divergence and $\text{BCE}(\cdot, \cdot)$ denotes binary cross-entropy. The global loss over the dataset is given by:

$$\mathcal{L}^{\mathcal{D}}(\theta_\psi) = \sum_{\mathbf{c} \in \mathcal{C}} \mathbb{E}_{(\mathbf{x}, \mathbf{c}, \mathbf{x}') \sim \mathcal{D}_{\mathbf{c}}} \mathcal{L}(\mathbf{x}, \mathbf{x}', \theta_\psi),$$

where $\mathcal{D}_{\mathbf{c}}$ represents the distribution of transitions for the controller class \mathbf{c} in the domain.

Subsequent to training, we evaluate θ_ψ on a held-out validation subset to obtain a validation loss $\ell_{\text{val}}(\psi)$. We transform $\ell_{\text{val}}(\psi)$ into a scalar score $s(\psi) \in [0, 100]$ via monotonic rescaling (e.g., linear normalization across the candidate group) and provide this score as feedback to the language model. This closed-loop process allows the foundation model to refine its proposals based on low-level data feedback, prioritizing predicates whose classifiers are effectively optimizable. The loop terminates when a target number of accepted predicates is collected or a maximum iteration count is reached; both parameters are treated as hyperparameters. All seed predicates undergo this LLM-in-the-loop bilevel learning process. Proposals with $\ell_{\text{val}}(\psi)$ below a specified threshold are subsequently aggregated into the set of basic dynamic predicate candidates $\hat{\Psi}_{\text{dyn}}^{\text{b}}$.

Finally, in contrast to previous approaches where predicate classifiers θ_ψ are either fixed (e.g., grammar functions [11] or frozen VFMs [23]) or optimized from scratch [16], UniPred proposed to optimize predicates atop deep representations from a pre-trained visual foundation model (DINOv3) in image-based domains. Formally, given an input image, an object detector [52] and a visual encoder produce a state $\mathbf{x} \in \mathbb{R}^{N \times K}$ comprising one feature vector per object in \mathcal{O} . Because these strong feature representations inherently encode rich geometric and semantic information, the learned predicate classifiers generalize robustly with sparse training trajectories.

To summarize, UniPred employs a language model to provide a symbolic prior over predicates and effects, utilizes an LLM-in-the-loop bilevel supervision scheme to align neural predicates with low-level dynamics, and leverages a visual foundation model to extract robust object-centric features when applicable (image domain). These integrated components create a unified loop that renders predicate learning significantly more cost- and data-efficient than purely bottom-up methods [16], [11], while ensuring grounding in low-level data for customized domains.

D. Derived-aware Predicate Selection

With the basic dynamic predicate pool $\hat{\Psi}_{\text{dyn}}^{\text{b}}$ obtained from the unified bilevel learning system, UniPred identifies the subset of predicates most valuable for planning. Following previous work [11], [16], we consider augmenting the basic

Algorithm 1: Derived-aware Predicate Selection

Input: Basic predicate pool $\hat{\Psi}_{\text{dyn}}^{\text{b}}$, Training tasks \mathcal{D} , Goal/Static predicates $\Psi_{\text{G}}, \Psi_{\text{sta}}$

Output: Selected dynamic predicates Ψ_{dyn}

```

1 Initialize selected set  $\Psi_{\text{dyn}} \leftarrow \emptyset$ 
2 Initialize candidate pool  $\tilde{\Psi}_{\text{dyn}} \leftarrow \hat{\Psi}_{\text{dyn}}^{\text{b}}$ 
3 Initialize best score  $J^* \leftarrow \infty$ 
4 while True do
5    $\psi_{\text{best}} \leftarrow \text{None}$ ,  $J_{\text{min}} \leftarrow \infty$ 
6   for each candidate  $\psi \in \tilde{\Psi}_{\text{dyn}}$  do
7     Form candidate set
8      $\tilde{\Psi} \leftarrow \Psi_{\text{dyn}} \cup \{\psi\} \cup \Psi_{\text{G}} \cup \Psi_{\text{sta}}$ 
9     Procedure EVALUATE( $\tilde{\Psi}, \mathcal{D}$ ):
10      Ground states in  $\mathcal{D}$  to obtain atom dataset
11      Learn operators  $\tilde{\mathcal{O}}$  using  $\tilde{\Psi}$ 
12      // A* planner with  $\tilde{\mathcal{O}}$ 
13      while planning for each task in  $\mathcal{D}$  do
14        Expand current state using  $\tilde{\mathcal{O}}$ 
15        if derived predicates in  $\tilde{\Psi}$  then
16          Compute derived atoms
17        Reconstruct plan skeleton  $\pi$ 
18      if  $J < J_{\text{min}}$  then
19         $J_{\text{min}} \leftarrow J$ ,  $\psi_{\text{best}} \leftarrow \psi$ 
20    if  $J_{\text{min}} \geq J^*$  or  $J_{\text{min}} < \epsilon$  then
21      break // Converged or threshold met
22     $\Psi_{\text{dyn}} \leftarrow \Psi_{\text{dyn}} \cup \{\psi_{\text{best}}\}$ 
23     $\tilde{\Psi}_{\text{dyn}} \leftarrow \tilde{\Psi}_{\text{dyn}} \setminus \{\psi_{\text{best}}\}$ 
24     $J^* \leftarrow J_{\text{min}}$ 
25    Add derived predicates to candidate pool if
26       $\psi_{\text{best}} \in \hat{\Psi}_{\text{dyn}}^{\text{b}}$  then
27        Generate derived forms  $\Psi_{\text{dyn}, \text{best}}^{\text{d}}$  from  $\psi_{\text{best}}$ 
28         $\tilde{\Psi}_{\text{dyn}} \leftarrow \tilde{\Psi}_{\text{dyn}} \cup \Psi_{\text{dyn}, \text{best}}^{\text{d}}$  // Expand search
29 return  $\Psi_{\text{dyn}}$ 

```

predicates with negations and quantifiers, which are denoted as derived dynamic predicate candidates $\hat{\Psi}_{\text{dyn}}^{\text{d}}$. Crucially, UniPred distinguishes between $\hat{\Psi}_{\text{dyn}}^{\text{b}}$ and $\hat{\Psi}_{\text{dyn}}^{\text{d}}$, yielding symbolic models that are more expressive than those produced by prior predicate invention approaches.

At a high level, previous methods select the final predicate set Ψ_{dyn} from the union of basic and derived predicates $\hat{\Psi}_{\text{dyn}}^{\text{b}} \cup \hat{\Psi}_{\text{dyn}}^{\text{d}}$ by optimizing a planning-driven surrogate objective $J(\Psi_{\text{dyn}}, \mathcal{D})$ [11]. Specifically, $J(\Psi_{\text{dyn}}, \mathcal{D})$ evaluates whether the planning operators [7] derived from Ψ_{dyn} can: (1) reproduce the demonstrated task plans in \mathcal{D} , and (2) minimize node expansions during task planning. However, a key limitation of prior work [11], [16] is that this evaluation relies on STRIPS assumptions, where predicates must appear consistently in the

effects of operators. A representative failure case is illustrated in our table cleaning domain (see Figure 1), where the derived predicate $\forall ?\text{toy}. \neg P_{\text{toy_table_0}}(? \text{toy}, ? \text{table})$ is a critical precondition for the operator $\text{WipeTable}(? \text{table})$, yet it does not appear as a consistent effect in any operator. Consequently, the operator learning stage fails during the calculation of the planning-driven surrogate objective, resulting in an unexpressive set of predicates.

A key insight in UniPred is that basic dynamic predicates $\hat{\Psi}_{\text{dyn}}^b$ naturally appear consistently in operator effects, whereas derived predicates $\hat{\Psi}_{\text{dyn}}^d$ may not. This discrepancy arises because effect-based supervision [16] implicitly ensures that for any predicate absent from an operator’s effect set, its ground atoms remain constant during transitions. Since derived candidates $\hat{\Psi}_{\text{dyn}}^d$ are not learned with such supervision, they may exhibit inconsistent appearances in effect sets. Motivated by this observation, we introduce a derived-aware predicate selection pipeline in UniPred. The complete predicate selection procedure is detailed in Algorithm 1.

Similar to previous work [11], [16], UniPred employs a hill-climbing procedure to optimize $J(\Psi_{\text{dyn}}, \mathcal{D})$. The search is initialized with an empty set of selected dynamic predicates Ψ_{dyn} , a candidate pool $\tilde{\Psi}_{\text{dyn}}$ initially set to the basic predicates $\hat{\Psi}_{\text{dyn}}^b$, and a sufficiently large best score J^* . Starting from the current set Ψ_{dyn} , each iteration seeks a single new predicate $\psi_{\text{best}} \in \tilde{\Psi}_{\text{dyn}}$ that minimizes the objective J . To achieve this, for each candidate addition yielding a temporary set $\tilde{\Psi}$, we invoke a predicate evaluation routine to compute $J(\tilde{\Psi}, \mathcal{D})$. We then add the optimal predicate ψ_{best} to Ψ_{dyn} and remove it from the candidate set $\tilde{\Psi}_{\text{dyn}}$. When a basic predicate $\psi_{\text{best}} \in \hat{\Psi}_{\text{dyn}}^b$ is selected, we generate its derived forms $\Psi_{\text{dyn}, \text{best}}^d$ and add them to the candidate pool $\tilde{\Psi}_{\text{dyn}}$. The search terminates when the score improves beyond a target threshold or when no remaining predicate further reduces J . The predicate evaluation routine consists of three steps: (1) Given the current dynamic predicates Ψ_{dyn} and the new predicate ψ , we form $\tilde{\Psi} = \{\Psi_G, \Psi_{\text{sta}}, \Psi_{\text{dyn}}, \psi\}$ and ground all demonstration states \mathbf{x} to obtain a ground atom dataset. (2) Using the learned lifted effect vectors Δ^ψ for $\psi \in \tilde{\Psi}_{\text{dyn}}$, we derive the add (Eff^+) and delete (Eff^-) effects for each operator $\tilde{\text{Op}}^c$ and compute precondition sets $\tilde{\text{Pre}}$ using the standard intersection rule [7]. (3) The resulting operator set $\tilde{\text{Op}}$ and $\tilde{\Psi}$ are used to run an A* planner [53] that generates plan skeletons for each training task. These skeletons are compared with the demonstration skeletons $\bar{\pi}$ to compute $J(\tilde{\Psi}_{\text{dyn}}, \mathcal{D})$.

Our key strategy during the third step (A* search) is to use operators to transition basic ground atoms, and subsequently calculate derived ground atoms from the basic state, rather than predicting them via operators. At each symbolic state, we first apply an operator to obtain the next set of ground atoms for the basic predicates Ψ_{dyn}^b . We then perform a *derived closure* step that deterministically computes all derived atoms implied by the current basic atoms and the definitions in Ψ_{dyn}^d . Consider the example from the table cleaning domain with the derived predicate $\forall ?\text{toy}. \neg P_{\text{toy_table_0}}(? \text{toy}, ? \text{table})$. Suppose that applying an operator deletes all basic atoms of $P_{\text{toy_table_0}}(? \text{toy}, ? \text{table})$. During derived closure, the sys-

tem evaluates the universal condition and adds the corresponding ground atom $\forall ?\text{toy}. \neg P_{\text{toy_table_0}}(? \text{toy}, ? \text{table})$ to the state. Thus, even though the quantifier never appears directly in any operator effect set, this closure step allows the planner to evaluate its truth value based on the basic atoms $P_{\text{toy_table_0}}$ and plan accordingly.

Upon termination of the hill-climbing search, the resulting dynamic predicate set $\Psi_{\text{dyn}} = \Psi_{\text{dyn}}^b \cup \Psi_{\text{dyn}}^d$ is passed to the operator, sampler, and skills learning stages.

E. Operator, Sampler, and Skill Learning

We primarily build upon prior work on operator and sampler learning for bilevel planning [16], [10], [11], [14], extending it to our setting which integrates both parametrized controllers and learned closed-loop skills.

Given the final predicate set $\Psi = \{\Psi_G, \Psi_{\text{sta}}, \Psi_{\text{dyn}}\}$ and the controller set \mathcal{C} , the predicate learning phase yields the corresponding lifted effect vectors Δ^ψ for all $\psi \in \Psi$. For each controller class $\mathcal{C} \in \mathcal{C}$, we construct its operator $\text{Op}^c = \langle \text{Var}, \text{Pre}, \text{Eff}^+, \text{Eff}^- \rangle$ by extracting Eff^+ and Eff^- from the components of Δ^ψ , and deriving Pre using the standard intersection rule over ground atoms that hold prior to successful executions of \mathcal{C} [7], [11].

For parametrized controllers, we learn samplers η^c from demonstration tuples $(\mathbf{x}, \mathcal{C}, (\mathbf{o}_1, \dots, \mathbf{o}_v), \omega)$ via supervised learning. This process models the conditional distribution $\omega \sim \eta^c(\cdot \mid \mathbf{x}, (\mathbf{o}_1, \dots, \mathbf{o}_v))$, following the formulation in [10].

Furthermore, our controller set \mathcal{C} incorporates closed-loop skills learned directly from the same demonstrations (e.g., ACT-based policies [46]). We treat these skills as controllers that share the same symbolic operators Op^c but do not require separate samplers, as the low-level policy inherently proposes continuous actions until the desired effects are achieved. Consequently, parametrized controllers with samplers and learned closed-loop skills are unified under a common operator-based abstraction for bilevel planning.

V. EXPERIMENT

A. Implementation Details

Computational Hardware: Experiments for the *Satellites*, *Blocks*, *Packing*, *Table Clean Sim*, *Table Clean Real* domains were conducted on a workstation equipped with a single NVIDIA RTX 4090 GPU and an Intel Ultra 265 CPU. Conversely, the *Tools* domain relies on point cloud observations; due to the higher computational demands of this observation space, we utilized a server featuring a single NVIDIA A100 GPU (80 GB VRAM) and an AMD EPYC 7543 32-Core CPU. **Real Robot Experiment Hardware:** As illustrated in Figure 6, our real-world experimental setup comprises an AgileX Robotics Piper 6-DoF manipulator and two Intel RealSense D435i cameras. A static camera positioned frontally provides a global top-down view of the workspace, while a wrist-mounted camera captures egocentric visual data. We collected demonstrations via the teleoperation interface depicted in the figure, using a smaller master arm to control the slave manipulator. During data collection, we recorded the robot state and synchronized RGB streams from both cameras; these data

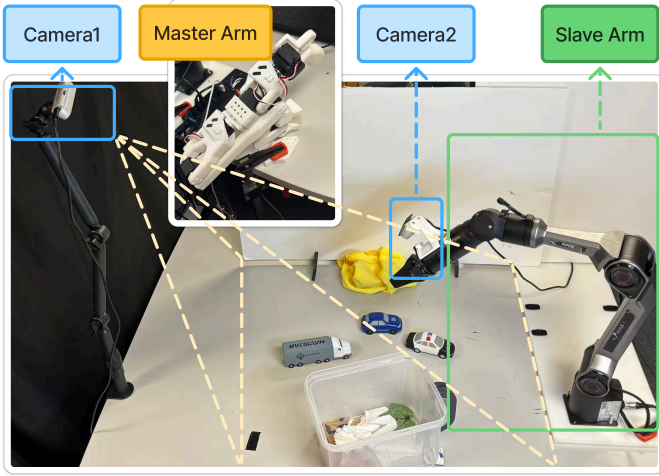


Fig. 6: Overview of our real world experimental platform, showing the master arm used for teleoperation, the slave arm that executes the learned policies, and the two RGB cameras (Camera 1 and Camera 2) that capture observations of the table scene.

serve as inputs for both predicate invention and the training of low-level controller skills.

UniPred Training Details: We employ GPT-4o as the default model for UniPred, and—unless otherwise specified—all LLM- or VLM-based components throughout the entire system also rely exclusively on GPT-4o. For the baselines reported in Table II, we additionally evaluate two model variants: `gemini-2.5-flash-lite` for the Gemini baseline and `qwen-plus` for the Qwen baseline. To ensure a fair comparison, all models operate at a fixed temperature of 0.2. Each LLM receives the same structured prompt containing an incomplete PDDL domain definition, operator demonstration sequences, interaction history, and a concise task description (e.g., completing PDDL fragments or proposing effects). Importantly, these prompts include no domain-specific hints beyond explicitly observable data. All baseline prompts strictly follow the fairness constraints used for UniPred.

For predicate classifiers, we adopt architectures tailored to the observation modality of each domain. In domains where states are represented by continuous pose features, we train a 128-layer MLP encoder. Conversely, for point-cloud observations, we employ a PointNet-based feature extractor. In image-based domains, visual embeddings are extracted using DINOv3 prior to classification. Across all domains, predicate classifiers are trained for a fixed number of epochs (typically 100). A predicate is designated as consistent when its classification loss converges below a specific threshold (typically 0.005). Further implementation details, hyperparameters, and training scripts are provided in our publicly code repository.

Baselines: We evaluate our approach against several established baselines to demonstrate its efficacy:

In simulated, non-image domains:

- 1) **IVNTR** [16]: A foundational bilevel learning approach for predicate invention, utilizing neural loss to guide symbolic effect searches.
- 2) **GNN** [54] and **Transformer** [55]: Relational neural policy learning methods trained via standard behavior

cloning pipelines and evaluated with a shooting strategy.

- 3) **Grammar** [11]: Uses pre-defined grammar rules for predicate invention. However, this baseline fails to produce effective predicates across all three tested domains.
- 4) **Random**: Generates plans randomly, serving as a basic control method to benchmark algorithmic performance.

In real image domains:

- 1) We group these three variants of ViLa together, as both rely on a vision-language model to plan directly from images without any learned symbolic abstractions.
 - **ViLa-zero-shot** [56]: Uses the original ViLa prompting scheme without incorporating demonstrations from our real image domains. The VLM performs direct image-conditioned planning purely through in-context examples from the original ViLa.
 - **ViLa-fewshot** [23]: Extends ViLa-zero-shot by augmenting the prompt with a small number of demonstrations drawn from our real image domains. The VLM continues to plan directly from images, but now benefits from in-domain examples more aligned with our tasks and visual distributions.
 - **ViLa-HPE (Heavily Prompt-Engineered)**: A prompt-optimized variant in which we exhaustively hand-engineer detailed, task-specific constraints with the goal of maximizing ViLa’s success rate. Unlike other baselines, this setting provides the VLM with explicit operational constraints such as “*the gripper must be empty before picking any object*”, and other domain-specific descriptions that are never learned from data. This version relies on substantial manual prompt engineering that is not feasible for generalization or scalable deployment.
- 2) A group of VLM-based bilevel planning baselines that differ in how predicates are proposed and labeled.
 - **Pix2Pred** [23]: The standard, fully top-down version. A VLM is prompted with domain types Δ , a task description, and example images to propose symbolic predicates. The same VLM labels these predicates on all states to produce atoms. Operators are then learned via effect extraction and intersection, without any neural predicate classifiers.
 - **Pix2Pred w/ GT predicates**: Disables predicate invention in **Pix2Pred** by manually supplying the oracle predicate set. The VLM is still used for predicate labeling during inference time.
 - **Pix2Pred w/ GT predicates and partial GT label**: Based on **Pix2Pred w/ GT predicates**, but replaces VLM labeling with ground truth for a subset of predicates in test time. The remaining predicates are still labeled by the VLM.
- 3) **ACT** [46]: An end to end visuomotor imitation learning baseline that maps image observations directly to low-level actions. ACT learns a separate policy for each skill from demonstration trajectories and executes without symbolic abstractions.

Simulated Domains: We evaluate the proposed methods across five diverse simulated robot planning domains, shown

in Figure 8, each characterized by distinct state representations and varying complexities:

- *Satellites*: Adapted from prior work [13], this domain involves multiple satellites collaboratively capturing sensor data from designated targets. The states are represented by SE2 poses and associated object attributes. Training scenarios ($\mathcal{T}^{\text{train}}$) consist of 2 satellites and 2 targets, while test scenarios ($\mathcal{T}^{\text{test}}$) involve 3 satellites and 3 targets, introducing additional complexity.
- *Blocks*: Inspired by [16], this domain requires a robot to manipulate 3D blocks to construct specified goal towers. Unlike the classic Blocks World setup, the goals involve constructing two-level towers, effectively packing pairs of blocks. The training set ($\mathcal{T}^{\text{train}}$) features scenarios with 4–5 blocks, whereas the test set ($\mathcal{T}^{\text{test}}$) increases complexity to 6–7 blocks.
- *Tools*: This challenging domain features high-dimensional state spaces represented as object-centric point clouds. The environment includes three types of tools (wrench, screwdriver, hammer) and their corresponding fastening items (bolt, screw, nail), along with various contraptions. The primary objective is to secure each fastening item onto a designated board using the correct tool. A distinctive requirement for screwdrivers and screws is geometric compatibility, determined by analyzing shape via point cloud data; for wrenches and hammers, item-tool matching is based solely on object type. Notably, this domain defines a larger set of operators compared to others, resulting in a significantly larger predicate search space. Training tasks involve securing two items with two contraptions, while test tasks scale complexity by requiring two to three items and three contraptions.
- *Packing*: This domain models constrained packing of items into segmented boxes. The environment contains several boxes and items of different sizes. Each box includes a divider that splits the interior into two regions, and each region can hold at most one item. The robot must place all items into boxes by selecting suitable boxes and adjusting divider positions so that items of different sizes fit without overlap or violation of capacity constraints. States are represented by top down images together with object pose. The robot does not have direct access to region capacities; instead, it must infer from the image whether a given region can accommodate a candidate item, so feasibility of placements depends on visual reasoning. Training tasks use fewer boxes and items, while test tasks increase their number, leading to more coupled packing decisions and longer plans.
- *Table Clean Sim*: This domain instantiates the table-cleaning task used as the running example throughout the paper. Each scene contains a robot, a table, a box, several toys scattered on the table, a towel, and dirt that can only be removed by wiping. The goal is to place all toys inside the box and remove all dirt from the table. Tasks are subject to several constraints: wiping while toys remain on the table is prohibited, as it would sweep the toys away; wiping is disallowed when the box is placed

on the near side of the table to avoid collision; and to store the towel in the box, the box must be placed on the near side to facilitate grasping. States are represented by vector features for each object, including 2D poses and attributes such as surface cleanliness and gripper status. Training tasks ($\mathcal{T}^{\text{train}}$) contain exactly two toys and initialize from a standard configuration where the robot hand is empty and toys lie on the table. Test tasks ($\mathcal{T}^{\text{test}}$) increase complexity by including three toys and initial states absent from demonstrations, such as scenes where the robot already grasps an object or toys are pre-positioned in the box.

Real Image Domains: We further evaluate the approach on the real-robot domains where states are represented by object-centric visual features extracted from RGB images:

- *Table Clean Real*: This domain shares the same underlying task structure as *Table Clean (Sim)*: a robot must place all toys into a box and wipe away non-graspable dirt on the table using a towel, subject to the same wiping and reachability constraints described above. The primary distinction lies in the state representation. Each scene is observed through RGB images from the cameras described in Section V-A, utilizing a visual foundation model to extract object-centric embeddings for each detected object.

Simulated Experiment Setup: For each domain, we implement an oracle bilevel planner (Oracle) solely to collect training demonstrations. We conduct evaluations across five simulated domains—Satellites, Blocks, Tools, Packing, and TableCleanSim—reporting results averaged over five random seeds. For Satellites, Blocks, and Tools, we collect 500 demonstration trajectories per seed. Conversely, for Packing and TableCleanSim, we limit collection to 50 trajectories per seed. During testing, each seed within each domain comprises 50 in-distribution tasks sampled from $T \sim \mathcal{T}^{\text{train}}$ and 50 generalization tasks sampled from $T \sim \mathcal{T}^{\text{test}}$. We report the success rate for all methods subject to a uniform maximum planning time budget.

In addition to planning success, we quantify the computational cost required to construct the abstraction. We define the total learning time as $\tau = \tau_{\text{predinv}} + \tau_{\text{predsel}} + \tau_{\text{skill}}$, where τ_{predinv} denotes the duration for candidate dynamic predicate invention, τ_{predsel} represents the time for scoring and selecting predicates, and τ_{skill} is the time allocated for training controllers. For our method, we measure these components individually and report their sum as τ . For baselines employing different training pipelines, we utilize their reported total training time as τ .

Real-robot Experiment Setup: For the Table Clean Real domain, we collect 20 human demonstrations via teleoperation, using a master arm to control the slave arm. Camera 1 records RGB video, which is manually segmented into actions. We select a representative image for each state and perform object detection to obtain object-centric crops. We employ GroundingDINO [57] as the primary detector; for objects that prove challenging to detect, we utilize a custom detector integrating DINO and SAM2 [52]. Pick-and-place

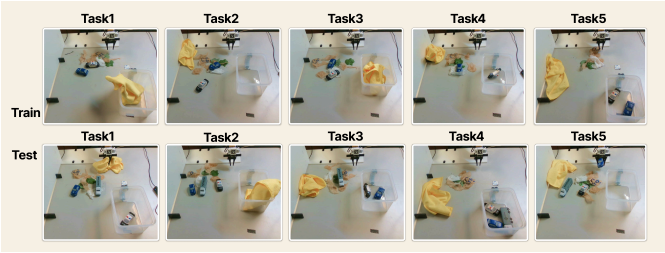


Fig. 7: Illustration of the ten tasks in the Table Clean Real domain. The top row shows the five training tasks, and the bottom row shows the five held-out test tasks.

controllers, such as `PickToyFromTable`, are implemented as analytic skills: `SAM2` is used to extract object point clouds and compute grasp and place poses. Conversely, controllers such as `PushBox` and `PullBox` are implemented using `ACT`, trained on the same 20 teleoperation trajectories. This training utilizes master and slave arm states alongside RGB images from camera1 and camera2.

We evaluate performance across two dimensions. First, we conduct offline task evaluation using images. We define 10 tasks (5 training and 5 testing), as shown in Figure 7. For each task, we collect 10 random seeds and execute each seed three times, comparing the average task success rates of the different methods on both training and test sets. Since the original `Pix2Pred` cannot propose the `HandEmpty` predicate and struggles to label relational predicates such as `Near(robot, box)`, we include three `Pix2Pred` variants in this evaluation: the original method, a variant without invention provided with the oracle predicate set, and a variant without invention that additionally receives ground truth labels for `Near(robot, box)`. Second, we evaluate real-robot execution on 10 tasks (5 training and 5 testing), utilizing 3 seeds per task. For each seed, we allow up to 3 rollouts and report the overall execution success rate. In this setting, the Oracle baseline corresponds to a human operator selecting the subsequent controller based on the current state. We exclude the `IVNTR` baseline from the Table Clean Real experiments, as it is unable to discover the derived predicates required for this domain.

B. Empirical Results

Simulated domains. The quantitative comparison on the five simulated domains is summarized in Table II and illustrated qualitatively in Figure 8. Across all domains, our approach `UniPred` closely tracks the oracle planner on the training tasks and generalizes well to the held out test configurations, while remaining computationally efficient.

Compared to the strong bottom up structure learning baseline `IVNTR`, `UniPred` achieves very similar success rates on the *Satellites*, *Blocks*, and *Tools* domains. For *Packing* and *Table Clean Sim*, `IVNTR` is not directly applicable because it cannot perform derived-aware predicate selection, which is required to obtain a sound symbolic model in these domains, so we do not report `IVNTR` success rates there; we return to this design choice in our ablation on derived-aware selection in Section V-C. However, the learning time of `UniPred` is

consistently much lower: in *Tools*, it discovers predicates with **more than a factor of three speedup** compared with `IVNTR`, and we observe the same qualitative pattern on the remaining domains. This shows that our method can match the accuracy of a strong bottom up relational learner on the domains where it applies, while greatly reducing the computational cost of predicate invention in training.

The behavior cloning baselines `GNN` and `Transformer` can often fit the demonstrations reasonably well on the training tasks in some domains, but their test performance deteriorates sharply, especially in settings where the test instances contain longer horizons or more objects than the demonstrations. The grammar based predicate invention baseline is strongly limited in our domains, where the state spaces can’t be easily abstracted by pre-defined grammar functions.

Table II also reports variants of `UniPred` that replace the default language model with `Qwen` or `Gemini`. Across all domains where we evaluate them, these variants keep both train and test success high, with only modest drops in generalization performance and somewhat increased learning time. This suggests that `UniPred` is robust to the exact choice of language model back end, but that stronger priors from the model still translate into better sample efficiency and slightly higher test success.

Overall, across all five simulated domains, `UniPred` reaches accuracy that is comparable to or better than the best bottom up learning baseline while being substantially more efficient, and at the same time it significantly outperforms the top down behavior cloning policies based on `GNN` and `Transformer` in both test success and robustness to distribution shift.

Task planning evaluation from images. Quantitative task planning results for the real-robot domain are summarized in Table III. In this setting, the method only needs to generate a feasible task plan by observing the init RGB image.

In the Table Clean Real domain, `UniPred` achieves success rates of 94.0% on training tasks and 92.0% on test tasks. These results indicate that predicates discovered from image-based demonstrations are sufficiently accurate to support reliable long-horizon planning, even in the presence of visual noise and actuation uncertainty.

The original `Pix2Pred` configuration cannot be reliably instantiated with our real-world data, as our challenging setting precludes the use of carefully engineered, domain-specific prompts. During training, the vision-language model (VLM) fails to propose key predicates, such as `HandEmpty`, and frequently generates inconsistent labels in cluttered scenes. Consequently, the learned transition model fails to converge, and the resulting planner achieves near-zero success. To better analyze the limitations of this approach, we evaluate two diagnostic variants that bypass the predicate invention stage.

In the first variant, “`Pix2Pred w/ GT preds`”, we provide the oracle predicate set and restrict the VLM’s role to labeling atoms from images at test time. As shown in Table III, while this configuration improves upon the original, it yields success rates of only 47.3% (train) and 33.3% (test)—significantly lower than `UniPred`. Error analysis suggests that a primary failure mode is the difficulty of assigning truth values to predicates that are challenging to describe precisely in natural

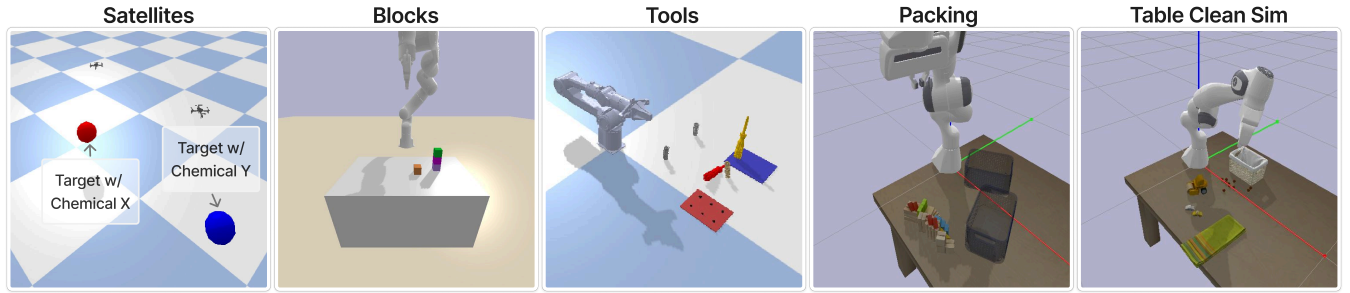


Fig. 8: Visualization of the five simulated planning domains studied in this work.

Method	Satellites (SE2)			Blocks (Vec3)			Tools (PCD)			Packing (Image)			Table Clean Sim (SE2)		
	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	τ	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	τ	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	τ	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	τ	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	τ
Oracle	100.0	100.0	/	100.0	100.0	/	100.0	100.0	/	100.0	100.0	/	100.0	100.0	/
UniPred (Ours)	99.6	95.2	1222.0	99.2	81.6	4104.4	100.0	100.0	11779.2	100.0	100.0	3657.4	96.0	93.4	764.6
IVNTR	99.2	94.0	5129.8	99.2	73.2	12939.4	100.0	100.0	45167.2	0.0	0.0	/	0.0	0.0	/
GNN	92.4	11.6	1303.0	89.2	38.8	1163.2	0.0	0.0	657.8	0.0	0.0	1286.0	0.0	0.0	360.2
Transformer	75.2	4.4	1778.6	34.4	16.8	1756.0	0.0	0.0	724.0	0.0	0.0	1364.8	0.0	0.0	951.0
Grammar	0.0	0.0	/	0.0	0.0	/	/	/	/	/	/	/	0.0	0.0	/
Random	0.0	0.0	/	10.0	1.2	/	0.0	0.0	/	12.4	1.6	/	3.6	1.2	/
UniPred(Qwen)	98.0	82.0	1493.0	100.0	75.6	5824.0	100.0	100.0	15801.0	100.0	100.0	4608.2	95.2	91.2	1274.6
UniPred(Gemini)	99.2	94.0	2686.4	100.0	78.4	6894.6	100.0	100.0	14495.2	100.0	100.0	3937.0	96.4	90.4	883.0

TABLE II: Success rate and runtime comparison on simulated domains. $\mathcal{T}^{\text{train}}$ and $\mathcal{T}^{\text{test}}$ denote training and test distribution, respectively. τ is the total learning time (seconds). Pix2Pred [23] is not applicable here since these domains do not use RGB images only as state representations.

language, such as `Near(robot, box)` and similar spatial relations.

In the second variant, “Pix2Pred w/ GT preds. and partial GT labels”, we provide ground-truth labels for these complex relational predicates, querying the VLM only for the remaining ones. This modification further improves performance to 87.3% (train) and 73.3% (test); however, the results remain inferior to UniPred. Qualitatively, we observe that in cluttered scenes, delegating predicate labeling directly to the VLM often yields inconsistent atom sets. For instance, in certain states where the robot clearly holds a toy, the model incorrectly labels `HandEmpty` as true, or fails to detect contact between the towel and the table surface (see Figure 9). These local labeling errors propagate through the planning process, causing the resulting plans to omit necessary actions or select redundant or even invalid operations.

The direct vision-language planning baselines also struggle in this domain. “ViLa zero-shot” and “ViLa few-shot” achieve, at best, single-digit success rates during task evaluation and frequently fail to produce valid plans for held-out scenes. In practice, they tend to generate action sequences that violate wiping constraints or neglect the towel entirely. Notably, the few-shot variant does not improve upon the zero-shot baseline and can even degrade test performance. We find that “ViLa few-shot” often overfits to the provided demonstrations. For instance, if a demonstration trajectory begins with a `PullBox` action, the planner repeatedly selects `PullBox` as the initial step, even in states where moving the box is unnecessary. This behavior indicates a strong bias toward

mimicking examples rather than reasoning about the current scene dynamics. We additionally evaluate a heavily hand-engineered variant, “ViLa-HPE”, in which we provide the VLM with exhaustive, task-specific prompting designed to mitigate the failure modes observed in the zero-shot and few-shot settings. Specifically, the prompt explicitly enumerates key operational constraints. This version reflects an upper-bound scenario for those vision-language planning baselines. Despite this extensive engineering, “ViLa-HPE” achieves success rates of only 20.7% (train) and 11.3% (test), substantially below both UniPred and the strongest Pix2Pred variant.

Real Robot Execution. We further evaluate execution performance on the physical system by running the planners on 10 Table Clean Real tasks three random seeds each with the same task setting as task planning evaluation shown in Figure 7, following the protocol described in Section V-A. For each method, we aggregate outcomes into fractional success scores that quantify the extent of table cleaning across tasks and seeds. We report the overall averages in Table IV.

In this setting, the “Oracle” baseline consists of a human operator who inspects the current state and selects the subsequent controller, utilizing the same perception stack and low-level controllers as the autonomous methods. When evaluated on the same set of test scenes as the Oracle, UniPred attains an average fractional success of 0.777, significantly outperforming the strongest vision-language model baseline, “Pix2Pred w/ GT preds. and partial GT labels”, which achieves 0.521. Since UniPred shares the detector and controller library with the Oracle, it is subject to identical

Method	Table Clean Real	
	$\mathcal{T}_{\text{train}}$	$\mathcal{T}_{\text{test}}$
UniPred (Ours)	94.0	92.0
Pix2Pred w/ GT preds. and partial GT labels	87.3	73.3
Pix2Pred w/ GT preds.	47.3	33.3
ViLa-zero-shot	3.3	4.0
ViLa-fewshot	6.7	0.0
ViLa-HPE (Heavily Prompt-Engineered)	20.7	11.3
Pix2Pred	/	/

TABLE III: Task planning evaluation on the Table Clean Real domain. Train / Test accuracy are success rates on training and held-out tasks.

perception and actuation limitations. Furthermore, its residual errors mirror those observed during offline task evaluation: in states diverging significantly from the training distribution, atom classifiers occasionally mislabel key predicates, causing the symbolic planner to generate incorrect plans. The “ViLa zero-shot”, “ViLa few-shot”, and “ACT” baselines yield zero fractional success under this protocol, failing to complete any long-horizon table-cleaning tasks from image observations. Collectively, these real-robot results demonstrate that the unified predicate invention pipeline, which performs well in simulation, remains robust when state representations are derived from real-world images and plans are executed on physical hardware. A critical advantage in this setting is closed-loop execution: after each controller terminates, the current state is re-grounded into atoms. This allows UniPred to detect deviations from the nominal plan and replan accordingly (as illustrated in Figure 3).

C. Ablation Studies and Analysis

We now investigate the individual contributions of the primary components of UniPred through a series of ablation studies conducted in simulated domains.

Effect of unified bilevel learning. The ablation study presented in Table V spans three simulated domains: *Satellites*, *Blocks*, and *Tools*. For each variant, we report training and testing success rates averaged across these domains, alongside the average total learning time.

UniPred represents the complete three-stage pipeline detailed in IV, integrating PDDL completion, unified bilevel learning, and loss feedback. “UniPred w/o PDDL Completion” omits the second stage, initiating unified bilevel learning from a cold start. Here, predicate sets are sampled directly from the language model without the benefit of a warm start derived from PDDL structure. “UniPred w/o Unified Bilevel Learning” eliminates the third stage, performing only a single iteration of PDDL completion; the language model generates the domain once, and the resulting predicate set is utilized for planning without further refinement. “UniPred w/o Loss Feedback” retains the unified bilevel learning loop but excludes the conversion of task loss into a scalar score for the language model. Consequently, new predicate sets are sampled based solely on the language prior and simple heuristics, rather than explicit performance feedback.

The full UniPred configuration achieves the optimal balance between accuracy and efficiency, attaining 99.6% average success on the training distribution and 92.3% on test tasks, with an average learning time of 5701.9 seconds. Omitting PDDL completion renders unified bilevel learning significantly more computationally expensive and less stable. Conversely, removing unified bilevel learning reduces computational cost (averaging 2871.1 seconds) by invoking the language model only once. However, the resulting predicates are not tuned to specific tasks, yielding substantially lower success rates. Finally, “w/o Loss Feedback” achieves success rates comparable to the full UniPred (99.6% training and 92.5% testing) but requires a longer learning duration. This suggests that loss-based feedback directs the search over predicate sets more effectively, reducing the number of required iterations.

We further display Figure 10 to demonstrate these results qualitatively. When tasking UniPred and IVNTR with learning the same predicate, both UniPred variants leverage the language model and its domain priors to propose candidate predicates that satisfy most structural constraints at the onset of training. In contrast, IVNTR must explore a significantly larger number of candidates to identify similarly effective predicates. Comparing the two UniPred variants, the version incorporating loss feedback refines these initial candidates more effectively. By utilizing loss-translated scores to discard redundant or brittle predicates and propose more compact, generalizable alternatives, it achieves the efficiency gains and quantitative improvements in test success observed in Table V.

Effect of different LLM interfaces. We subsequently evaluate alternative strategies for utilizing the LLM within the *Blocks* domain, as summarized in Table VI. Our proposed approach, which employs the LLM solely to propose predicate templates that are subsequently trained and evaluated via the bilevel pipeline, achieves a test success rate of 81.6%. Conversely, restricting the LLM to generate a static PDDL domain description—denoted as “LLM complete PDDL”—reduces test success to 63.6%. Although the generated PDDL is typically syntactically valid, it frequently omits critical preconditions or effects and fails to adapt adequately to the statistical distribution of the demonstration data.

The most direct strategy, “LLM propose code (Code as Policy [39])”, prompts the LLM to output executable planning code that operates on low-level states and selects primitive actions. This variant yields suboptimal performance, achieving only 1.2% test success, despite marginally higher training success. These results mirror the failure modes observed in neural policy baselines and substantiate the hypothesis that asking an LLM to directly generate complete programs from demonstrations for long-horizon robotic tasks is significantly less effective than leveraging it to guide a structured predicate search grounded in low-level data.

Role of semantic information in the LLM prompt. To investigate the necessity of providing semantic information to the LLM, we compare UniPred against a variant that omits operator names and known predicate names from the prompt (w/o Semantic), retaining only type signatures and a minimal task description. For reference, we also report the performance of IVNTR under the same experimental protocol.

Planner	Seed/Task	Table Clean Real										Mean	Avg.
		T1	T2	T3	T4	T5	T6	T7	T8	T9	T10		
Oracle (Human)	S0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	0.900	0.872
	S1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	0.950	
	S2	0.5	0.33	1.0	1.0	1.0	1.0	0.5	1.0	1.0	0.33	0.766	
UniPred(ours)	S0	1.0	1.0	0.33	1.0	1.0	1.0	0.5	1.0	1.0	0.5	0.833	0.777
	S1	0.33	1.0	1.0	1.0	1.0	0.5	0.33	0.33	1.0	1.0	0.749	
	S2	0.5	0.5	1.0	1.0	1.0	1.0	0.0	0.5	1.0	1.0	0.750	
Pix2Pred w/ GT preds. and partial GT labels	S0	0.33	0.50	0.50	0.33	1.00	1.00	1.00	1.00	0.00	0.00	0.566	0.521
	S1	0.00	0.33	0.00	0.33	0.50	0.50	0.50	0.50	1.00	0.00	0.366	
	S2	0.33	1.00	0.50	0.33	1.00	0.33	1.00	1.00	0.50	0.33	0.632	
ViLa-zero-shot&Vila-few-shot&ACT	S0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000
	S1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
	S2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

TABLE IV: Real robot execution results on the Table Clean Real. Entries are fractional task success scores for each planner across ten test tasks (T1–T10) and three random seeds (S0–S2). The “Mean” column averages across tasks for each seed and the “Avg.” column averages over all seeds. The Oracle (Human) baseline corresponds to a human who inspects the current state and selects the next controller in the sequence while using the same perception stack and low level controllers as all other methods.

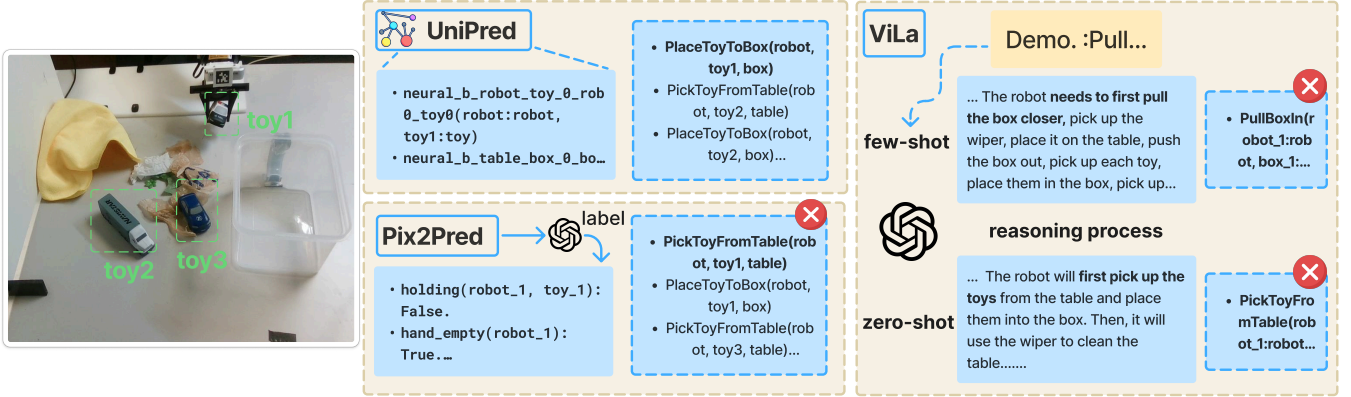


Fig. 9: Qualitative comparison on a real table-cleaning task. UniPred produces accurate symbolic predicates and action sequences, whereas Pix2Pred and ViLa generate incorrect or inconsistent next actions despite receiving the same visual observations.

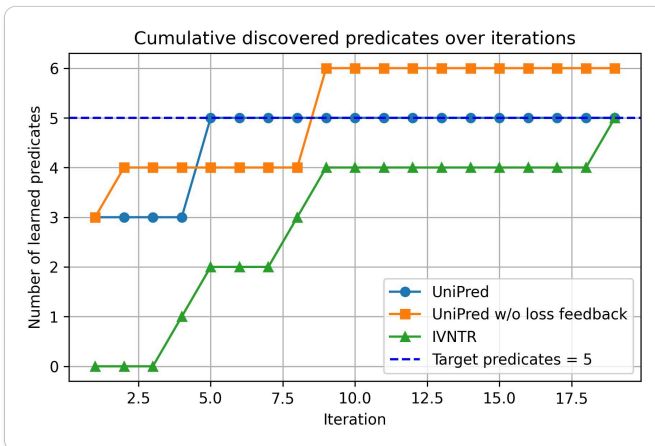


Fig. 10: Cumulative number of valid predicates discovered over iterations for UniPred, UniPred w/o loss feedback, and IVNTR, compared against the target set of five predicates (dashed line).

As detailed in Table VII, the full UniPred pipeline achieves 99.6% average training success, 92.3% average test success, and a mean learning time of 5.7×10^3 seconds. Removing

Method	$\overline{\mathcal{T}}_{\text{train}}$	$\overline{\mathcal{T}}_{\text{test}}$	\bar{t}
UniPred (Ours)	99.6	92.3	5701.9
UniPred w/o PDDL Completion	99.7	86.7	15455.1
UniPred w/o Unified Bilevel Learning	30.0	22.4	2871.1
UniPred w/o Loss Feedback	99.6	92.5	6612.5

TABLE V: Ablation result for the effect of unified bilevel learning in UniPred.

Method	Blocks (Vec3)	
	$\overline{\mathcal{T}}_{\text{train}}$	$\overline{\mathcal{T}}_{\text{test}}$
UniPred (Ours)	99.2	81.6
LLM complete PDDL	79.6	63.6
LLM propose code (Code as Policy [39])	12.4	1.2

TABLE VI: Comparison between different ways of LLM proposing for planning.

semantic information (w/o Semantic) maintains high accuracy—with 100.0% training and 91.9% test success—but incurs a significant computational penalty, increasing the learn-

Method	$\mathcal{T}_{\text{train}}$	$\mathcal{T}_{\text{test}}$	\bar{t}
UniPred (Ours)	99.6	92.3	5701.9
UniPred w/o Semantic	100.0	91.9	18612.9
IVNTR	99.5	89.1	21081.8

TABLE VII: Comparison between UniPred, UniPred w/o semantic information (removing operator names and names of known predicates from the prompt), and IVNTR, averaged over the three simulated domains.

Method	Table Clean Sim (SE2)	
	$\mathcal{T}_{\text{train}}$	$\mathcal{T}_{\text{test}}$
UniPred (Ours)	96.0	93.4
IVNTR	0.0	0.0
UniPred w/o derived. select	0.0	0.0
IVNTR w/ derived. select	95.2	91.6

TABLE VIII: Ablation study for the derived-aware predicate selection module.

ing time to 1.86×10^4 seconds. IVNTR attains comparable training success (99.5%) but lower test success (89.1%) and exhibits further computational overhead, with an average learning time of 2.11×10^4 seconds.

These findings suggest that while rich semantic context is not strictly necessary for correctness, it substantially enhances efficiency by steering the LLM toward predicate proposals that align with the underlying task structure. Even in the absence of semantic identifiers, UniPred remains more efficient than IVNTR, indicating that the LLM-in-the-loop bilevel learning pipeline provides superior priors and reasoning capabilities compared to the tree expansion approach employed by IVNTR. Thus, semantic information primarily functions as an additional prior that further reduces the exploration cost within the predicate search space.

Derived-aware predicate selection. Table VIII presents an ablation study in the *Table Clean Sim* domain, where task completion requires reasoning over quantified conditions, such as “all toys have been placed.” Since these conditions can only be expressed via derived predicates, successful planning necessitates the effective selection of such predicates during the search. Both the full UniPred and IVNTR variants equipped with the derived-aware selection module achieve high success. Conversely, variants lacking this module—“UniPred w/o derived-aware selection” and IVNTR—fail completely, achieving 0.0% success on both splits.

In the absence of the derived-aware module, the hill-climbing procedure treats all predicates indistinguishably, failing to differentiate between low-level relational predicates and derived predicates that encode quantified conditions. Consequently, the search fails to discover the necessary derived predicate, causing the score to plateau after minimal improvement, as illustrated by the IVNTR curves in Figure 11. In contrast, enabling the derived-aware selection module allows UniPred to explicitly track and evaluate predicates that capture quantified effects. Upon the addition of a salient derived predicate to the pool, the selection score exhibits a sharp

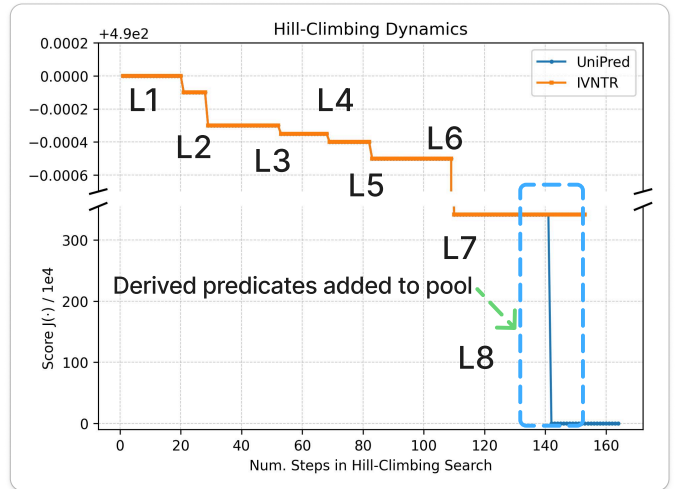


Fig. 11: Hill-climbing dynamics of the predicate selection score for IVNTR and UniPred. The curve shows how the score changes over hill-climbing steps: most updates cause only small score adjustments that are visually subtle, while a few iterations produce noticeable drops when useful derived predicates are added to the pool.

decline, indicating that the planner has successfully encoded the domain objective.

VI. DISCUSSIONS

Limitations and Future Work: Despite its effectiveness in long-horizon planning, our UniPred framework has several limitations that suggest promising avenues for future research. (1) Similar to prior work such as IVNTR [16], our predicate classifiers are supervised with effect vectors. This imposes the implicit constraint that each low-level controller must correspond to a unique, deterministic STRIPS operator in the symbolic domain (with finite object-centric effects). A crucial future direction is investigating how symbolic predicates can be discovered in contexts involving non-STRIPS controllers (with continuous, non-deterministic action effects) [58], [59]. (2) Our predicate invention framework currently requires a fixed, provided set of low-level controllers, which are typically learned via imitation. Future work should explore extending the framework to jointly discover new operational skills [60], [61], [62] and the corresponding high-level predicates and symbolic abstractions that govern them. (3) Since UniPred learns its abstractions directly from human demonstrations and indirectly through the representations of foundation models, the discovered predicates are inherently constrained by human prior knowledge. Discovering emergent, non-intuitive, or “surprisingly smart” abstractions [60] that move beyond human preconceptions remains an intriguing and important direction. (4) Following established works in bilevel planning [7], [16], [11], UniPred operates under the assumption of a fully-observable and deterministic environment. A necessary next step is studying how to robustly learn and utilize these symbolic predicates within more challenging contexts, such as Partially Observable Markov Decision Processes (POMDPs) [63] and environments characterized by stochastic physics.

Conclusion: In this work, we presented UniPred, a unified bilevel learning framework for automatically discovering sym-

bolic predicates grounded by expressive neural classifiers. The core innovation of our system is threefold: First, UniPred introduces an LLM-in-the-loop bilevel learning system where high-level knowledge priors proposed by foundation models guide the top-down efficient exploration of the symbolic predicate space, while bottom-up learning feedback from low-level interaction data effectively rectifies high-level model inconsistencies. Second, we proposed a derived-aware predicate selection pipeline that explicitly learns useful, planning-driven predicates for Non-STRIPS domains, resulting in a more expressive and broadly applicable symbolic world model for complex, long-horizon robot planning. Finally, for image-based domains, UniPred further enhances efficient perceptual generalizability by directly optimizing predicate classifiers atop the powerful visual features extracted from a recent VFM [1], requiring only few real-world demonstrations. Across five simulated domains and one real-world domain, we demonstrated the efficiency and efficacy of UniPred across various state representations. Our results show that compared to purely top-down methods, UniPred achieves a $2 \sim 4\times$ higher success rate due to the effective task decomposition enabled by our discovered and optimized predicates. Furthermore, compared to purely bottom-up methods, UniPred demonstrates a $3 \sim 4\times$ faster learning speed by leveraging knowledge priors from pre-trained LLMs. We conclude that UniPred represents a pivotal step towards robust, foundation model-based abstraction learning for tackling general, long-horizon robot planning problems.

VII. ACKNOWLEDGEMENT

This work has been funded in part by the Army Research Laboratory (ARL) award W911NF-23-2-0007 and W911QX-24-F-0049, and the Office of Naval Research (ONR) award N00014-23-1-2840. We also acknowledge the support of the Air Force Research Laboratory (AFRL), the Defense Advanced Research Projects Agency (DARPA), under agreement number FA8750-23-2-1015, and the Defence Science and Technology Agency (DSTA) under contract #DST000EC124000205. This work used Bridge s-2 at PSC through allocation cis220039p from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program which is supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #213296.

REFERENCES

- [1] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, *et al.*, “Dinov3,” *arXiv preprint arXiv:2508.10104*, 2025.
- [2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, “Gpt-4 Technical Report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [3] J. Mao, T. Lozano-Pérez, J. Tenenbaum, and L. Kaelbling, “What Planning Problems Can A Relational Neural Network Solve?” in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2024.
- [4] B. Li, Z. Li, Q. Du, J. Luo, W. Wang, Y. Xie, S. Stepputtis, C. Wang, K. P. Sycara, P. K. Ravikumar, *et al.*, “LogiCity: Advancing Neuro-Symbolic AI with Abstract Urban Simulation,” in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2024, pp. 69 840–69 864.
- [5] M. Asai and C. Muise, “Learning Neural-Dymbolic Descriptive Planning Models via Cube-Space Priors: the Voyage Home (to STRIPS),” in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 2021, pp. 2676–2682.
- [6] G. Konidaris, L. Pack Kaelbling, and T. Lozano-Pérez, “From skills to symbols: Learning symbolic representations for abstract high-level planning,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 215–289, 2018.
- [7] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, “Learning Neuro-Symbolic Relational Transition Models for Bilevel Planning,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 4166–4173.
- [8] R. Chitnis, T. Silver, J. B. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, “GLIB: Efficient Exploration for Relational Model-Based Reinforcement Learning via Goal-Literal Babbling,” in *Proceedings of The AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, 2021, pp. 11 782–11 791.
- [9] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, “Learning Symbolic Operators for Task and Motion Planning,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3182–3189.
- [10] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, “Learning Neuro-Symbolic Skills for Bilevel Planning,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.
- [11] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum, “Predicate Invention for Bilevel Planning,” in *Proceedings of The AAAI Conference on Artificial Intelligence (AAAI)*, vol. 37, 2023, pp. 12 120–12 129.
- [12] Y. Liang, N. Kumar, H. Tang, A. Weller, J. B. Tenenbaum, T. Silver, J. F. Henriques, and K. Ellis, “VisualPredicator: Learning Abstract World Models With Neuro-Symbolic Predicates For Robot Planning,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.23156>
- [13] N. Kumar, W. McClinton, R. Chitnis, T. Silver, T. Lozano-Pérez, and L. P. Kaelbling, “Learning Efficient Abstract Planning Models That Choose What to Predict,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.
- [14] N. Kumar, T. Silver, W. McClinton, L. Zhao, S. Proulx, T. Lozano-Pérez, L. P. Kaelbling, and J. Barry, “Practice Makes Perfect: Planning To Learn Skill Parameter Policies,” in *Proceedings of the Robotics: Science And Systems (RSS)*, 2024.
- [15] A. Li and T. Silver, “Embodied Active Learning of Relational State Abstractions for Bilevel Planning,” in *Proceedings of the Conference on Lifelong Learning Agents (CoLLAs)*, 2023, pp. 358–375.
- [16] B. Li, T. Silver, S. Scherer, and A. Gray, “Bilevel Learning for Bilevel Planning,” in *Proceedings of the Robotics: Science And Systems (RSS)*, 2025.
- [17] M. Helmert, “The Fast Downward Planning System,” *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.
- [18] J. Hoffmann, “FF: The fast-forward planning system,” *AI magazine*, vol. 22, no. 3, pp. 57–57, 2001.
- [19] L. P. Kaelbling and T. Lozano-Pérez, “Hierarchical Task and Motion Planning in the Now,” in *Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 1470–1477.
- [20] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, “Integrated Task and Motion Planning,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021.
- [21] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, “Pddlstream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning,” in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 30, 2020, pp. 440–448.
- [22] G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez, “From skills to symbols: Learning symbolic representations for abstract high-level planning,” *Journal of Artificial Intelligence Research (JAIR)*, 2018. [Online]. Available: <https://jair.org/index.php/jair/article/view/11175/26380>
- [23] A. Athalye, N. Kumar, T. Silver, Y. Liang, T. Lozano-Pérez, and L. P. Kaelbling, “Predicate Invention from Pixels via Pretrained Vision-Language Models,” *arXiv preprint arXiv:2501.00296*, 2024.
- [24] M. Han, Y. Zhu, S.-C. Zhu, Y. N. Wu, and Y. Zhu, “InterPreT: Interactive Predicate Learning from Language Feedback for Generalizable Task Planning,” *arXiv preprint arXiv:2405.19758*, 2024.
- [25] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, *et al.*, “Gemini: A Family of Highly Capable Multimodal Models,” *arXiv preprint arXiv:2312.11805*, 2023.

- [26] OpenAI, “Gpt-4v(ision) system card,” 2023, accessed: 2024-08-22. [Online]. Available: https://cdn.openai.com/papers/GPTV_System_Card.pdf
- [27] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, and H. Liu, “Llm³: large language model-based task and motion planning with motion failure reasoning,” in *arXiv preprint*, 2024.
- [28] S. Kambhampati, K. Valmeekam, L. Guan, K. Stechly, M. Verma, S. Bhambri, L. Saldyt, and A. Murthy, “Llms can’t plan, but can help planning in llm-modulo frameworks,” in *arXiv preprint*, 2024.
- [29] A. Curtis, N. Kumar, J. Cao, T. Lozano-Pérez, and L. P. Kaelbling, “Trust the PRC3s: Solving long-horizon robotics problems with LLMs and constraint satisfaction,” in *Conference on Robot Learning (CoRL)*, 2024. [Online]. Available: <https://openreview.net/forum?id=r6ZhiVYriY>
- [30] K. Valmeekam, A. Olmo, S. Sreedharan, and S. Kambhampati, “Large language models still can’t plan (a benchmark for llms on planning and reasoning about change),” in *NeurIPS 2022 Foundation Models for Decision Making Workshop*.
- [31] A. Ahmetoglu, E. Oztop, and E. Ugur, “Symbolic manipulation planning with discovered object and relational predicates,” *IEEE Robotics and Automation Letters*, vol. 10, no. 2, p. 1968–1975, Feb. 2025. [Online]. Available: <http://dx.doi.org/10.1109/LRA.2025.3527338>
- [32] J. Mao, T. Lozano-Pérez, J. Tenenbaum, and L. Kaelbling, “PDS-ketch: Integrated Domain Programming, Learning, and Planning,” in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 36972–36984.
- [33] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, “Openvla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024.
- [34] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, “Rdt-1b: a diffusion foundation model for bimanual manipulation,” *arXiv preprint arXiv:2410.07864*, 2024.
- [35] R. Team, “Rdt2: Enabling zero-shot cross-embodiment generalization by scaling up umi data,” September 2025. [Online]. Available: <https://github.com/thu-ml/RDT2>
- [36] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, “ π_0 : A vision-language-action flow model for general robot control,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [37] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky, “ $\pi_{0.5}$: a vision-language-action model with open-world generalization,” 2025. [Online]. Available: <https://arxiv.org/abs/2504.16054>
- [38] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. T. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K.-H. Lee, Y. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu, “Do as i can, not as i say: Grounding language in robotic affordances,” in *Conference on Robot Learning (CoRL)*, 2023.
- [39] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” in *arXiv preprint*, 2022.
- [40] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, “VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023, pp. 540–562.
- [41] D. Shah, B. Osinski, B. Ichter, and S. Levine, “LM-nav: Robotic navigation with large pre-trained models of language, vision, and action,” in *6th Annual Conference on Robot Learning*, 2022. [Online]. Available: <https://openreview.net/forum?id=UW5A3SweAH>
- [42] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” 2021. [Online]. Available: <https://arxiv.org/abs/2103.00020>
- [43] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, “Palm-e: An embodied multimodal language model,” in *arXiv preprint arXiv:2303.03378*, 2023.
- [44] Z. Yang, C. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling, “Guiding Long-Horizon Task and Motion Planning with Vision Language Models,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.02193>
- [45] N. Kumar, F. Ramos, D. Fox, and C. R. Garrett, “Open-World Task and Motion Planning via Vision-Language Model Inferred Constraints,” in *CoRL Workshop on Language and Robot Learning: Language as an Interface*, 2024.
- [46] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” in *Robotics: Science and Systems (RSS)*, 2023. [Online]. Available: <https://arxiv.org/pdf/2304.13705.pdf>
- [47] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” in *Robotics: Science and Systems (RSS)*, 2023. [Online]. Available: https://diffusion-policy.cs.columbia.edu/diffusion_policy_2023.pdf
- [48] L. Medeiros, “Language Segment-Anything,” 2024.
- [49] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, “Large language models as optimizers,” in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: <https://openreview.net/forum?id=Bb4VGOWELI>
- [50] H. B. Amor, L. Graesser, A. Iscen, D. D’Ambrosio, S. Abevruwan, A. Bewley, Y. Zhou, K. Kalirathinam, S. Mishra, and P. Sanketi, “Sas-prompt: Large language models as numerical optimizers for robot self-improvement,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 10 087–10 094.
- [51] D. McDermott, M. Ghallab, A. E. Howe, C. A. Knoblock, A. Ram, M. M. Veloso, D. S. Weld, and D. E. Wilkins, “PDDL-the Planning Domain Definition Language,” 1998.
- [52] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al., “Sam 2: Segment Anything in Images and Videos,” *arXiv preprint arXiv:2408.00714*, 2024.
- [53] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [54] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., “Relational Inductive Biases, Deep Learning, and Graph Networks,” *arXiv preprint arXiv:1806.01261*, 2018.
- [55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is All You Need,” in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.
- [56] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao, “Look Before You Leap: Unveiling the Power of GPT-4v in Robotic Vision-Language Planning,” *arXiv preprint arXiv:2311.17842*, 2023.
- [57] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al., “Grounding dino: Marrying dino with grounded pre-training for open-set object detection,” *arXiv preprint arXiv:2303.05499*, 2023.
- [58] J. Mendez-Mendez, “A Systematic Study of Large Language Models for Task and Motion Planning With PDDLStream,” *arXiv preprint arXiv:2510.00182*, 2025.
- [59] J. Mao, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, “Hybrid Declarative-Imperative Representations for Hybrid Discrete-Continuous Decision-Making,” in *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.
- [60] Y. I. Liu, B. Li, B. Eysenbach, and T. Silver, “SLAP: Shortcut Learning for Abstract Planning,” *arXiv preprint arXiv:2511.01107*, 2025.
- [61] A. Bagaria, J. K. Senthil, and G. Konidaris, “Skill Discovery for Exploration and Planning using Deep Skill Graphs,” in *International conference on machine learning*, 2021, pp. 521–531.
- [62] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, “Diversity is All You Need: Learning Skills without a Reward Function,” in *International Conference on Learning Representations*, 2025.
- [63] A. Curtis, H. Tang, T. Veloso, K. Ellis, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, “LLM-guided Probabilistic Program Induction for POMDP Model Estimation,” in *Conference on Robot Learning*, PMLR, 2025, pp. 3137–3184.