

000  
 001 **SKILLWRAPPER:**  
 002 **GENERATIVE PREDICATE INVENTION FOR**  
 003 **SKILL ABSTRACTION**  
 004  
 005  
 006

007 **Anonymous authors**  
 008 Paper under double-blind review  
 009  
 010  
 011

012 **ABSTRACT**  
 013  
 014  
 015  
 016  
 017  
 018  
 019  
 020  
 021  
 022  
 023  
 024  
 025  
 026  
 027  
 028

Generalizing from individual skill executions to solving long-horizon tasks remains a core challenge in building autonomous agents. A promising direction is learning high-level, symbolic representations of the low-level skills of the agents, enabling reasoning and planning independent of the low-level state space. Recent advances in foundation models have made it possible to generate symbolic predicates that operate on raw sensory inputs—a process we call *generative predicate invention*—to facilitate downstream representation learning. However, it remains unclear *which* formal properties the learned representations must satisfy, and *how* they can be learned to guarantee these properties. In this paper, we address both questions by presenting a formal theory of generative predicate invention for skill abstraction, resulting in symbolic operators that can be used for provably sound and complete planning. Within this framework, we propose SKILLWRAPPER, a method that leverages foundation models to actively collect robot data and learn human-interpretable, plannable representations of black-box skills, using only RGB image observations. Our extensive empirical evaluation in simulation and on real robots shows that SKILLWRAPPER learns abstract representations that enable solving unseen, long-horizon tasks in the real world with black-box skills.

029  
 030 **1 INTRODUCTION**  
 031

032 An autonomous agent operating in the real world must process low-level sensory and motor signals  
 033 while reasoning about high-level objectives (Doncieux et al., 2018; Konidaris, 2019). Analogous to  
 034 how humans can perform complex tasks, like cooking or cleaning, without reasoning about muscle-  
 035 level control, agents should have internal models of their skills that abstract away nuanced activities  
 036 on the lower level. Such models must capture the necessary conditions for a skill to be executed  
 037 (e.g., “*pouring a teapot requires holding it first*”) and the consequences of doing so (e.g., “*pouring*  
 038 *a teapot leaves it empty*”). These two properties, known as *preconditions* and *effects* in the AI  
 039 planning literature, enable compositional reasoning to identify long-horizon plans that can sequence  
 040 lower-level skills to solve a task. Typically, these models must be specified manually. However,  
 041 in real-world settings such skill representations may be nontrivial to acquire due to complex inter-  
 042 skill constraints specific to the agent’s embodiment. This calls for algorithms that learn symbolic  
 043 transition models of black-box skills without hand specification, enabling agents to directly utilize  
 044 those skills to solve long-horizon tasks with off-the-shelf AI planners.

045 Traditional approaches of skill abstraction often require factorizing the low-level state space to  
 046 learn classifiers for each symbolic representation, relying heavily on hand-collected transition  
 047 data (Konidaris et al., 2018). Recently, foundation models have enabled a new paradigm: generating  
 048 semantically meaningful predicates directly from raw observations and directly evaluating  
 049 their truth values on low-level observations (e.g., RGB images)—a process we refer to as *generative*  
 050 *predicate invention*. Recent work has explored how foundation models can be used for predicate  
 051 invention, by generating Python code to implement predicates (Liang et al., 2025) or sampling large  
 052 predicate pools followed by sub-selection. However, these methods produce ad-hoc planning rep-  
 053 resentations that cannot be guaranteed to solve a given task, and leave core questions on predicate  
 invention unanswered: *what* properties should these learned abstractions satisfy, and *how* can they  
 be learned to achieve these properties?

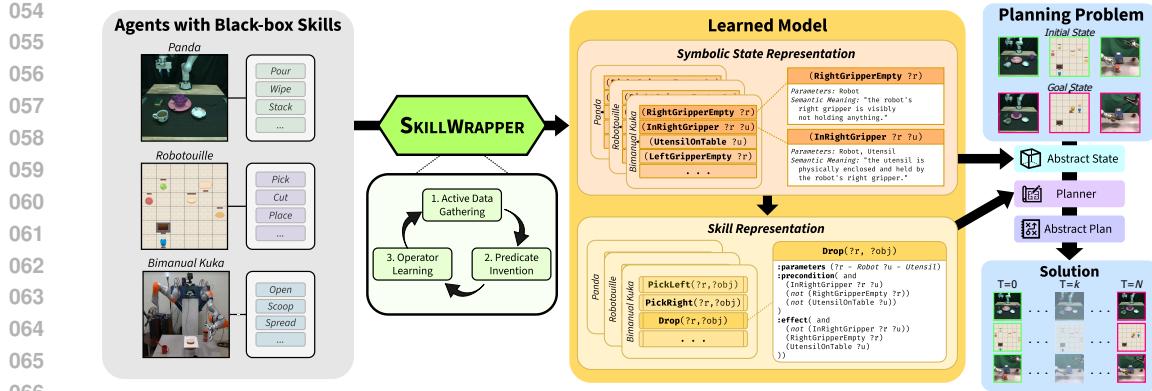


Figure 1: **Overview of SKILLWRAPPER.** For an agent equipped with black-box skills, SKILLWRAPPER learns skill representations that are compatible with off-the-shelf planners. These representations are comprised of predicates invented by the foundation model. Given a novel planning problem described using the initial state and goal state as RGB images, a foundation model produces the corresponding abstract states by applying the invented predicates to the low-level states. SKILLWRAPPER is *agnostic* to the agent, and we illustrate both real-world (robots) and simulated agents in this figure.

Our answer is twofold. First, we develop a formal theory of generative predicate invention for skill abstraction, precisely characterizing the conditions under which a learned skill representation will be provably *sound* and *complete* with respect to downstream planning. Building on this foundation, we introduce SKILLWRAPPER, a method explicitly designed to guarantee these theoretical criteria. SKILLWRAPPER uses foundation models in three ways: interactively collecting data in the environment, proposing predicates when the current model fails, and classifying predicate truth values based solely on RGB image observations. Using these data and predicates, SKILLWRAPPER learns symbolic representations of black-box skills that are both human-interpretable and directly usable for AI planning.

We highlight the following contributions: (1) A formal theory of generative predicate invention for *provably sound* and *complete* skill abstraction; (2) SKILLWRAPPER, a principled system built on this framework that leverages foundation models to learn interpretable symbolic representations of black-box skills; and (3) an extensive empirical evaluation of the system, demonstrating effectiveness in simulation and on two real robots.

## 2 PROBLEM SETTING

In this section, we briefly discuss our problem setting while defining it formally in Appendix A. We consider an agent equipped with a finite set of *object-centric skills*  $\Omega$ , modeled as black-box options (Sutton et al., 1999). The agent can execute any  $\omega \in \Omega$  and determine whether it succeeds, but it does not possess a symbolic transition model of these skills. Without such a model, the agent cannot plan over a long horizon without reasoning at the low-level state space  $\mathcal{S}$ , which is continuous, high-dimensional, and impractical for classical search. The goal of *Skill Model Learning* is to acquire a symbolic abstraction of skills that enables efficient composition via classical planning.

**Environment.** An *environment* is a tuple  $(\mathcal{S}, \mathcal{T}, \Omega, T)$ , where  $\mathcal{S}$  is the continuous state space,  $\mathcal{T}$  is a finite set of object types, and  $\Omega$  is the skill library. Each skill  $\omega \in \Omega$  is parameterized by object types drawn from the set  $\mathcal{T}$ . The environment dynamics are governed by an unknown transition function  $T : \mathcal{S} \times \Omega \rightarrow \mathcal{S}$ . A *setting* is defined as  $(s_0, \mathcal{O})$ , consisting of an initial state  $s_0 \in \mathcal{S}$  and a set of typed objects  $\mathcal{O}$  with  $\tau(o) \subseteq \mathcal{T}$  for each  $o \in \mathcal{O}$ .

**Black-box skills.** A skill  $\omega \in \Omega$  is a tuple  $(\mathcal{I}_\omega, \pi_\omega, \beta_\omega, \theta_\omega)$ , where  $\mathcal{I}_\omega \subseteq \mathcal{S}$  is the *initiation set*,  $\pi_\omega$  is the option policy,  $\beta_\omega \subseteq \mathcal{S}$  is the *termination set*, and  $\theta_\omega = (\tau_\omega^1, \dots, \tau_\omega^k)$  are the type constraints

108 on its  $k$  parameters. A skill instance  $\underline{\omega} = \omega(o_1, \dots, o_k)$  is valid for  $o_i \in \mathcal{O}$  if  $\tau_{\omega}^i \subseteq \tau(o_i)$  for all  $i$ .  
 109 Executing  $\omega$  from  $s \in \mathcal{I}_{\omega}$  terminates in some  $s' \in \beta_{\omega}$ , affecting only the state of the bound objects.  
 110

111 **Symbolic predicates and operators.** To enable abstract reasoning, we introduce a finite set of  
 112 predicates  $\mathcal{P}$ . Each  $\sigma \in \mathcal{P}$  is parameterized by  $\mathcal{T}$  and has a classifier  $\phi_{\sigma} : \mathcal{O}^n \times \mathcal{S} \rightarrow \{0, 1\}$  that  
 113 evaluates its truth value for arguments in a state. Grounding predicates with valid objects induces a  
 114 set of grounded predicates  $\bar{\mathcal{P}}$ , and hence an abstract state space  $\bar{\mathcal{S}} = 2^{\bar{\mathcal{P}}}$ . The *abstraction function*  
 115  $\Gamma : \mathcal{S} \rightarrow \bar{\mathcal{S}}$  maps each state to the set of grounded predicates true in that state, while the *grounding function*  
 116  $\mathcal{G} : \bar{\mathcal{S}} \rightarrow \mathcal{S}$  maps an abstract state to the states in which its grounded predicates hold.

117 An operator  $a \in \mathcal{A}$  is defined as  $(\omega_a, \theta_a, \text{PRE}_a, \text{EFF}_a)$ , where  $\omega_a \in \Omega$  is the associated skill,  $\theta_a$  are  
 118 typed operator parameters,  $\text{PRE}_a \subseteq \mathcal{P}$  is the precondition set, and  $\text{EFF}_a = (\text{EFF}_a^+, \text{EFF}_a^-)$  are the add  
 119 and delete effects. Grounding all operators  $a \in \mathcal{A}$  with objects  $o \in \mathcal{O}$  yields a set of abstract actions  
 120  $\bar{a} \in \bar{\mathcal{A}}$ , each executable whenever its ground preconditions hold in the current abstract state.  
 121

122 **Planning problems.** Given an environment  $(\mathcal{S}, \mathcal{T}, \Omega, T)$ , the agent’s objective is to learn an  
 123 abstract transition model  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$ , such that  $\mathcal{M}$  is sufficient for planning with  $\Omega$ . Formally, a  
 124 *skill planning problem* is  $(s_0, \mathcal{O}, \mathcal{S}_g)$ , where  $(s_0, \mathcal{O})$  is a setting and  $\mathcal{S}_g \subseteq \mathcal{S}$  are goal states. The  
 125 corresponding *abstract planning problem* is  $(\Gamma(s_0), \bar{\mathcal{A}}, \bar{\mathcal{S}}_g)$ , where  $\bar{\mathcal{S}}_g = \{\bar{s} \in \bar{\mathcal{S}} \mid \mathcal{G}(\bar{s}) \cap \mathcal{S}_g \neq \emptyset\}$ .  
 126 An abstract plan  $\bar{\pi} = \langle \bar{a}_1, \dots, \bar{a}_n \rangle$  is *valid* if its execution under  $\mathcal{M}$  yields an abstract trajectory  
 127 consistent with some feasible low-level trajectory under  $T$ , ending in a goal state.  
 128

129 **Problem statement.** The *Skill Model Learning* problem is: given experience in the form of  
 130 *state–skill-instance–next-state* tuples  $\langle s, \underline{\omega}, s' \rangle$ , learn a model  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$  such that every plan  
 131 found by a complete symbolic planner over  $\mathcal{M}$  corresponds to a feasible low-level skill plan.  
 132

### 3 SKILLWRAPPER

135 In this section, we introduce SKILLWRAPPER, a novel approach that autonomously learns symbolic  
 136 representations for black-box skills using the concepts defined in Appendix C. To produce a valid ab-  
 137 stract model that enables planning, SKILLWRAPPER iterates through a three-step process: 1) actively  
 138 proposing and executing exploratory skill sequences to collect data on the initiation and termina-  
 139 tion set of each skill, 2) incrementally building a set of predicates from scratch by contrasting positive  
 140 and negative examples, and then 3) constructing valid operators using these invented predicates,  
 141 from which further exploratory skill sequences can be proposed. This procedure is outlined in  
 142 Algorithm 1. As SKILLWRAPPER continues to collect data, add predicates, and update its planning  
 143 model, it learns a progressively more accurate abstract transition model. The resulting skill repre-  
 144 sentations, or *operators*, can be used with an off-the-shelf classical planner to solve task planning  
 145 problems specified using RGB images. We delve into the core components of our system (lines 4–9  
 146 of Algorithm 1) in the following subsections. Lastly, we provide strong theoretical results for the  
 147 soundness and completeness of SKILLWRAPPER in Section 3.4.

---

#### Algorithm 1 SKILLWRAPPER

---

```

1: Input: Set of skills  $\Omega$ , number of iterations  $m \in \mathbb{N}_1$ 
2: Output: Abstract transition model  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$ 
3:  $\mathcal{D}, \mathcal{P}, \mathcal{A} \leftarrow \emptyset$ 
4: for  $i \in \{1, \dots, m\}$  do
5:    $\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle \leftarrow \text{PROPOSESKILLSEQUENCE}(\Omega, \mathcal{D}, \mathcal{P}, \mathcal{A})$ 
6:    $\mathcal{D} \leftarrow \mathcal{D} \parallel \text{EXECUTESKILLS}(\underline{\omega}_1, \dots, \underline{\omega}_k)$ 
7:    $\mathcal{P} \leftarrow \text{INVENTPREDICATES}(\Omega, \mathcal{D}, \mathcal{P}, \mathcal{A})$ 
8:    $\mathcal{A} \leftarrow \text{LEARNOPERATORS}(\mathcal{D}, \mathcal{P})$ 
9: end for
10: return  $\mathcal{M}$ 

```

---

#### 3.1 ACTIVE DATA COLLECTION

158 To collect data for learning, our method commands the agent to execute its skills in the world and  
 159 then collects the resulting transitions. Each command is a sequence of skill instances  $\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle$ .  
 160

162 While executing these commands, the agent collects a dataset  $\mathcal{D}$  of transitions of the form  $\langle s, \underline{\omega}, s' \rangle$ ,  
 163 where  $s, s' \in \mathcal{S}$ . These transitions can answer two questions:

164

- 165 1. *Executability*: Can the skill instance  $\underline{\omega}$  be executed from state  $s$ ?
- 166 2. *Skill Dynamics*: If  $s \neq s'$ , what has changed in the environment due to executing  $\underline{\omega}$ ?

167

168 We guide the exploration of skill preconditions and effects using a foundation model, which pro-  
 169 poses skill sequences in natural language. Rather than naively sampling these sequences from the  
 170 foundation model’s token distribution, we prompt the foundation model for a batch of candidate  
 171 skill sequences and apply two scoring functions to bias the system toward promising sequences that  
 172 explore novel skill instance pairs and keep a balance between success and failure executions (see  
 173 Appendix B.1 for details). SKILLWRAPPER facilitates the efficient data collection process, which  
 174 results in a dataset  $\mathcal{D}$  that is critical to downstream processes, such as guiding predicate invention  
 175 with failure transitions, and improving the learned abstract model by eliminating unnecessary  
 176 preconditions, etc.

177

### 178 3.2 PREDICATE INVENTION

180 We now present our predicate invention algorithm, which, unlike prior work (Silver et al., 2023;  
 181 Liang et al., 2025), does not require an initial set of predicates to bootstrap the invention process.

182

184 **Conditions for predicate invention.** SKILLWRAPPER identifies two conditions under which the  
 185 current predicate vocabulary is insufficient, based on the desired properties of soundness and com-  
 186 pleteness (discussed fully in Appendix C). In these cases, the system must invent new predicates  
 187 to resolve incongruities between the observed data and the current abstract transition model. We  
 188 illustrate how SKILLWRAPPER can achieve the desired model properties in the Venn diagrams in  
 189 Appendix D.1.

190 To formally describe the conditions, we define two sets,  $\alpha_\omega$  and  $\zeta_\omega$ , representing the states in which  
 191 the model predicts that the skill may either be initiated (when  $s \in \alpha_\omega$ ) or terminated (when  $s \in \zeta_\omega$ ),  
 192 respectively. Both sets are derived from and defined with the learned operators, and their formal  
 193 definitions can be found in Appendix C.

194 The first condition arises when SKILLWRAPPER detects that the symbolic vocabulary cannot express  
 195 a necessary precondition for a skill. Concretely, this occurs when two transitions involve instances  
 196 of the same skill, one successful and one failed, both satisfy the initiation condition of the skill under  
 197 the current predicates. Because the vocabulary cannot distinguish between these initial states, an  
 198 additional predicate is required. Formally, this condition is expressed as:

199

$$200 \exists \langle s_i, \underline{\omega}_i, s'_i \rangle, \langle s_j, \underline{\omega}_j, s'_j \rangle \in \mathcal{D} \text{ s.t. } s_i \in \alpha_{\underline{\omega}_i}, s_j \in \alpha_{\underline{\omega}_j}, \text{ but } s_i \in \mathcal{I}_{\underline{\omega}_i} \text{ while } s_j \notin \mathcal{I}_{\underline{\omega}_j}.$$

201

202 The second condition used by SKILLWRAPPER to trigger predicate invention is based on inconsis-  
 203 tencies in observed skill effects. Specifically, this occurs when two transitions that involve instances  
 204 of the same skill produce identical abstract effects, despite one succeeding and the other failing. In  
 205 a deterministic setting, this reduces to a successful skill execution producing no effects, though the  
 206 condition naturally extends to stochastic settings with mid-execution failures. Formally, we express  
 207 the condition as:

208

$$209 \exists \langle s_i, \underline{\omega}_i, s'_i \rangle, \langle s_j, \underline{\omega}_j, s'_j \rangle \in \mathcal{D} \text{ s.t. } s'_i \in \zeta_{\underline{\omega}_i}, s'_j \in \zeta_{\underline{\omega}_j}, \text{ but } s_i \in \mathcal{I}_{\underline{\omega}_i} \text{ and } s_j \notin \mathcal{I}_{\underline{\omega}_j}.$$

210

$$211$$

212 **Contrastive predicate proposal** When a satisfying transition pair is identified under the condi-  
 213 tions of predicate invention, SKILLWRAPPER prompts the foundation model with the two transitions  
 214 and their corresponding states (RGB images) to propose a candidate predicate that can potentially  
 215 distinguish the transition pair. The transition pair offers contrastive clues for the foundation model  
 to propose predicates that precisely resolve the incongruity.

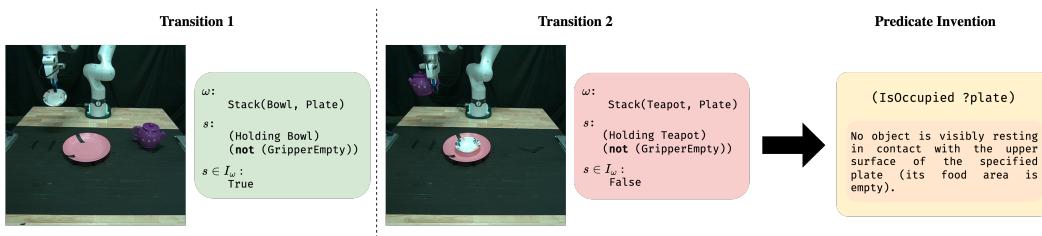


Figure 2: **Example of Predicate Invention.** The initial states of two transitions are both said to satisfy the preconditions of certain operators learned from the same skill, while transition 1 is successful, but transition 2 is not. In this case, the first condition (precondition) is triggered, and the foundation model is prompted with both transitions to invent a new predicate.

**Empirical predicate selection.** Although foundation models provide a strong prior on which information is skill-relevant for predicate construction, they may still produce errors and hallucinations. To ensure robustness, we introduce a scoring function that estimates the usefulness of a candidate predicate by adding it to the vocabulary and learning hypothetical operators. For each successful transition in  $\mathcal{D}$ , there must exist at least one operator with preconditions satisfied by the transition’s initial state; for each failed transition, no such operator may exist. Effect evaluation follows the same principle. After evaluating all transitions, we decide whether to add the candidate predicate based on an empirical threshold (Details of Algorithm 6 can be found in the appendix.)

### 3.3 OPERATOR LEARNING

Our operator learning procedure extends the *associative model learning* paradigm (Arora et al., 2018) to the setting of skill abstraction.

**Associative model learning.** A single skill may induce multiple distinct abstract state changes depending on the context of execution. To represent these *conditional effects*, SKILLWRAPPER clusters observed transitions based on their lifted (object-agnostic) effect sets, enabling it to learn a single operator across distinct instantiations of a skill. The preconditions of each operator are then computed as the intersection of all initial abstract states in the corresponding transitions, ensuring that each operator is both minimal and consistent.

**Multi-type object-centrism.** In realistic domains, objects do not fit neatly into single type categories, but rather belong to multiple overlapping categories (e.g., a Cup is fillable, while a Bottle is both fillable and openable). This complicates the process of generalizing grounded transitions into lifted operators, because it may be ambiguous which object attribute enables successful execution. We adopt a conservative strategy: SKILLWRAPPER assigns arguments of each operator using the lowest level of the type hierarchy consistent with the data, preventing over-generalization while retaining compositional structure.

**Predicate re-evaluation.** Predicates are generated sequentially, and early inventions may bias later stages if left unchecked. To mitigate this, SKILLWRAPPER re-applies the scoring function to the entire predicate set after each iteration of data collection. This allows spurious or redundant predicates to be discarded as more data is collected. In addition, tautological predicates—those that are always true or always false—are automatically filtered. As a result, the learned predicate set remains compact, informative, and aligned with the most recent transition data.

### 3.4 THEORETICAL ANALYSIS

We now provide theoretical guarantees for SKILLWRAPPER. These results show that the learned symbolic model is sound with respect to observed data and converges to a complete model with high probability. Full proofs are deferred to Appendix D.

**Theorem 1** (Soundness of SKILLWRAPPER). *Every operator  $a \in \mathcal{A}_n$  in the model  $\mathcal{M}_n$  learned by SKILLWRAPPER is supported by at least one observed transition  $\langle s, \omega, s' \rangle \in B_n$ . That is,  $\underline{s} \models \text{PRE}_a$  and  $\underline{s}' \models \text{EFF}_a$ .*

270 *Sketch.* SKILLWRAPPER constructs each operator directly from sampled transitions; thus no unsupported operator can appear.  
 271

272 **Lemma 1.** *For each  $\omega \in \Omega$ , the initiation set  $I_\omega$  and termination set  $\beta_\omega$  inferred by SKILLWRAPPER  
 273 match exactly the corresponding predicate sets  $\alpha_\omega$  and  $\zeta_\omega$  derived from  $B_n$ , i.e.  $I_\omega = \alpha_\omega$  and  
 274  $\beta_\omega = \zeta_\omega$ .*  
 275

276 *Sketch.* Any mismatch would contradict the termination condition of SKILLWRAPPER; thus initiation  
 277 and termination sets are consistent with observed data.

278 **Theorem 2** (Probabilistic-completeness of SKILLWRAPPER). *Let  $\mathcal{M}^*$  be the true complete model.  
 279 With probability at least  $1 - |\mathcal{H}| \exp(-n\epsilon)$ , the model  $\widehat{\mathcal{M}}_n$  learned from  $n$  i.i.d. samples satisfies  
 280  $d_{\text{compl}}(\widehat{\mathcal{M}}_n, \mathcal{M}^*) \leq \epsilon$ , i.e. it misses fewer than an  $\epsilon$ -fraction of feasible transitions.*  
 281

282 *Sketch.* Since  $\widehat{\text{Err}}(\widehat{\mathcal{M}}_n) = 0$  by construction, a Chernoff bound and union bound over the finite  
 283 hypothesis class imply that the true error is small with high probability.  
 284

285 Together, these results establish that SKILLWRAPPER learns symbolic operators that are sound with  
 286 respect to observed transitions, consistent in their preconditions and effects, and probabilistically  
 287 complete relative to the true underlying model. These properties justify SKILLWRAPPER as a reliable  
 288 model-learning procedure for planning. Next, we discuss empirical evaluation for SKILLWRAPPER.  
 289

## 290 4 EXPERIMENTS

291 For all experiments in this section, we consider images as fully observable state representations,  
 292 assuming that an abstract state can be inferred from an image without uncertainty. Both the initial  
 293 and goal states of each problem are specified using RGB images. These images may come from  
 294 diverse sources, including a top-down view of an animated game, a third-person camera observing  
 295 a robot, or the robot’s own egocentric perspective. All quantitative results reported in this section  
 296 are averaged over five independent runs for simulation experiments and three runs for real robot  
 297 experiments.  
 298

### 299 4.1 IMPLEMENTATION OF PREDICATES

300 We employ foundation models (specifically vision-language models or VLMs) for both predicate  
 301 generation and evaluation:

- 303 • A foundation model gives us a string that can be used as a lifted predicate (generates  
 304 interpretable relational classifiers with a good heuristic).
- 305 • After grounding with valid parameters, the predicate can be prompted to the foundation  
 306 model again to acquire the truth value. In other words, a foundation model can be used as  
 307 a relational classifier.

308 With these two properties, we can use the VLM’s response in string form as a relational predicate,  
 309 and a grounded version of the predicate can be used as a classifier. We use GPT-5 (OpenAI, 2025)  
 310 for predicate generation and evaluation. In addition to the system performance reported in the main  
 311 paper, we also conducted comprehensive studies of the component-wise reliability of the VLM in  
 312 Appendix G.

### 313 4.2 SIMULATION

315 We first conduct experiments in Robotouille (Gonzalez-Pumariega et al.,  
 316 2025), which is a simulated grid world kitchen domain with an agent  
 317 that has five high-level skills: *Pick*, *Place*, *Cut*, *Cook*, and *Stack*. In the  
 318 environment, there are several objects: a patty, lettuce, a top bun, and a  
 319 bottom bun; there is also a cutting board and a stove for cutting the lettuce  
 320 and cooking the patty, respectively. We design and categorize 50 abstract  
 321 planning problems: 20 easy problems, whose optimal solutions have no  
 322 more than 7 steps; 20 hard problems, whose optimal solutions have no  
 323 more than 15 steps; and 10 impossible tasks that cannot be realized in  
 the environment.

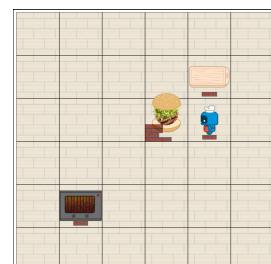


Figure 3: Screenshot of the Robotouille environment.

324 For a complete evaluation of SKILLWRAPPER, we compare SKILLWRAPPER against four baselines:  
 325

- 326 • **Expert Operators:** A human expert who is familiar with PDDL is asked to interact with  
 327 the environment and manually write predicates with semantics and PDDL operators.  
 328
- 329 • **System Predicates:** This baseline directly uses the built-in predicate set of the simulator,  
 330 which is designed to define any possible simulated state unambiguously. In addition, instead  
 331 of getting the truth values through classification with foundation models, this baseline has  
 332 access to the exact abstract states of the simulator. However, predicate invention is disabled  
 333 in this environment.  
 334
- 335 • **ViLa** (Hu et al., 2023): This baseline is a closed-loop VLM-based approach that iteratively  
 336 prompts a foundation model for the next action until the goal state is reached, given an  
 337 image observation and the agent’s action history.  
 338
- 339 • **Random Exploration:** Instead of proposing skill sequences, this baseline randomly sam-  
 340 ple a skill and populates the arguments with valid objects. This baseline shares the same  
 341 predicate invention and operator learning algorithms as SKILLWRAPPER.  
 342
- 343 • **No Heuristic:** This baseline is the same as SKILLWRAPPER, except that skill sequences are  
 344 selected randomly from the foundation model’s output without applying the heuristics.  
 345

346 For each baseline that performs operator learning, we run the learning algorithm for five iterations,  
 347 with each iteration proposing and executing one skill sequence consisting of 15 steps as their  
 348 interaction budgets. We then evaluate each method on the evaluation set and report the average  
 349 results in Table 1, where *Solved %* is the percentage of the problem set that was successfully solved  
 350 or where impossible tasks were correctly identified by returning an empty plan, and *PB* stands for  
 351 planning budgets—the number of plans that were tried before solving the problem (adopted from  
 352 Liang et al. (2025)). We set a planning budget cap of 10 across all problems; if the planning budget  
 353 has been used up for a problem, it is considered a failure. Theoretically, *PB* is an adequate metric that  
 354 reflects the completeness of the learned model, and the *impossible* problems reflect its soundness.  
 355

356 As shown in the table, SKILLWRAPPER outperforms all baselines that have no access to privileged  
 357 knowledge, and even surpasses the performance of the system predicates baseline (Sys Preds.) on  
 358 hard problems. Here we present the key insights, while leaving case studies, example operators, and  
 359 failure modes and analysis of SKILLWRAPPER in Appendix F.  
 360

361 Table 1: Baseline Comparison in Robotouille Environment

Method	Easy		Hard		Impossible
	Solved % $\uparrow$	PB $\downarrow$	Solved % $\uparrow$	PB $\downarrow$	
Expert Ops.	<b>81.0 <math>\pm</math> 3.7</b>	<b>1.9 <math>\pm</math> 0.4</b>	<b>58.1 <math>\pm</math> 3.9</b>	<b>4.2 <math>\pm</math> 0.4</b>	<b>100 <math>\pm</math> 0.0</b>
Sys Preds.	<b>79.0 <math>\pm</math> 3.7</b>	2.6 $\pm$ 0.2	22.0 $\pm$ 12.9	7.8 $\pm$ 1.3	42.0 $\pm$ 7.5
ViLa	46.0 $\pm$ 16.2	-	13.9 $\pm$ 11.6	-	20.0 $\pm$ 10.9
Random Exp.	4.0 $\pm$ 2.0	9.6 $\pm$ 0.2	0 $\pm$ 0.0	10.0 $\pm$ 0.0	100 $\pm$ 0.0
No Heuristic	<b>76.0 <math>\pm</math> 4.9</b>	<b>2.5 <math>\pm</math> 0.9</b>	24.0 $\pm$ 19.6	7.8 $\pm$ 1.8	80 $\pm$ 20.9
<b>Ours</b>	<b>74.0 <math>\pm</math> 3.7</b>	<b>2.7 <math>\pm</math> 0.4</b>	<b>40.0 <math>\pm</math> 3.2</b>	<b>6.3 <math>\pm</math> 0.4</b>	<b>100 <math>\pm</math> 0.0</b>

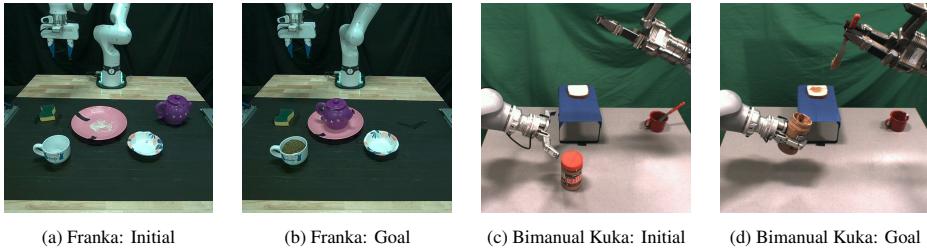
362 Compared to the expert-constructed operators, which serve as an approximate upper bound for  
 363 performance without privileged simulator access, SKILLWRAPPER demonstrates competitive accuracy  
 364 while requiring only a small number of exploratory interactions. In particular, SKILLWRAPPER  
 365 exhibits strong generalization from the easy to the hard set, indicating that the invented predicates  
 366 and learned operators capture meaningful abstractions rather than overfitting to specific training  
 367 traces. The gap between SKILLWRAPPER and ViLa highlights the benefit of explicitly learning a  
 368 symbolic model instead of relying solely on open-loop prompting, while the poor performance of  
 369 Random Exploration underscores the importance of guided data collection in learning effective  
 370 operators, and the unstable performance of No Heuristic in hard problems shows the importance of a  
 371 sufficient predicate set and how SKILLWRAPPER manages to reliably explore the state space and learn  
 372 it. These findings suggest that SKILLWRAPPER achieves a favorable trade-off between data efficiency  
 373 and model soundness and completeness, which is crucial for scaling to larger domains.  
 374

378  
379

## 4.3 REAL ROBOTS

380  
381  
382  
383  
384  
385  
386

To demonstrate the applicability of **SKILLWRAPPER** for real-world agentic settings, we designed two sets of experiments with two robotic platforms: a Franka Emika Panda robot (Figures 4a and 4b) and a bimanual platform with two Kuka iiwa robots (Figures 4c and 4d). For both robot experiments, we assume that all skills are deterministic, which is a common assumption made by existing work (Silver et al., 2023; Han et al., 2024; Liang et al., 2025; Athalye et al., 2025). We give **SKILLWRAPPER** 15 steps as its interaction budget per iteration. The supplementary material contains video demonstrations of both experiments.

387  
388  
389  
390  
391  
392  
393  
394  
395

396

Figure 4: Initial and Goal States for Real Robot Experiments.

397

398

**Generalization of SKILLWRAPPER.** In this setting, a tabletop Franka Emika Research 3 (Panda) manipulator has its skill set  $\Omega$  consisting of five black-box skills: *Pick*, *Place*, *Stack*, *Pour*, and *Wipe*. The object set  $\mathcal{O}$  contains five objects: a mug, a teapot, a plate, a bowl, and a sponge. The robotic agent can pick and place all objects except the mug and plate, pour ingredients from the teapot into the mug, and use the sponge to wipe the plate if it is dirty. To validate the generalization ability of the learned skill representations, we design three smaller training environments, such that each environment only contains a subset of  $\mathcal{O}$ , and thus only a subset of  $\Omega$  are executable. Each of the training environments contains fewer than 10 possible states. After running **SKILLWRAPPER** for exactly one iteration for each environment, we port the learned skill representations to the test environment that contains all objects in  $\mathcal{O}$ , which induces a state space of 34 possible abstract states. To quantify this generalization process, we similarly prepare an evaluation problem set that consists of five problems in the training environments, five problems in the test, and five *Impossible* problems across the two environments. The results are shown in Table 2.

411  
412

Table 2: Results of Generalization Experiment

413  
414  
415  
416  
417  
418  
419

Method	In-domain		Generalization		Impossible
	Solved % $\uparrow$	PB $\downarrow$	Solved % $\uparrow$	PB $\downarrow$	
Expert Ops.	66.7 $\pm$ 9.4	3.3 $\pm$ 0.9	53.3 $\pm$ 9.4	5.3 $\pm$ 0.9	46.7 $\pm$ 0.0
ViLa	46.7 $\pm$ 9.4	-	6.7 $\pm$ 9.4	-	6.7 $\pm$ 9.4
<b>Ours</b>	<b>76.7 <math>\pm</math> 9.4</b>	<b>2.7 <math>\pm</math> 0.9</b>	<b>60.0 <math>\pm</math> 0.0</b>	<b>4.0 <math>\pm</math> 0.0</b>	<b>66.7 <math>\pm</math> 9.4</b>

420

421

422  
423  
424  
425  
426  
427  
428  
429  
430  
431

**Learning Curve of SKILLWRAPPER.** In this bimanual manipulation setting, a robot with two Kuka iiwa arms is equipped with a skill set  $\Omega$  containing six black-box skills: *LeftArmPick*, *RightArmPick*, *Open*, *Scoop*, *Spread*, and *Drop*. The object set  $\mathcal{O}$  consists of three objects: a peanut butter jar, a knife, and a slice of bread. This robot can pick up the knife and jar, drop the knife, open the jar, scoop peanut butter with the knife, and spread it on the bread. Notably, this environment contains multiple dead ends, which would hinder the data gathering process. For example, the knife cannot be picked up again once dropped, the jar cannot be released once picked up, and the bread and knife cannot be cleaned once in contact with peanut butter. Moreover, the skills are heavily interdependent. We designed the experiment in this way to investigate the learning process of **SKILLWRAPPER** over several iterations. Again, we compare the performance against two baseline methods, ViLa and Expert Operators. An example of predictive truth value changes induced by a sequence is shown as in Figure 5. We observe that performance improves as **SKILLWRAPPER** progressively obtains more transition data and invents more predicates, finally surpassing the baselines. The performance improvement over iterations is shown in Figure 6.

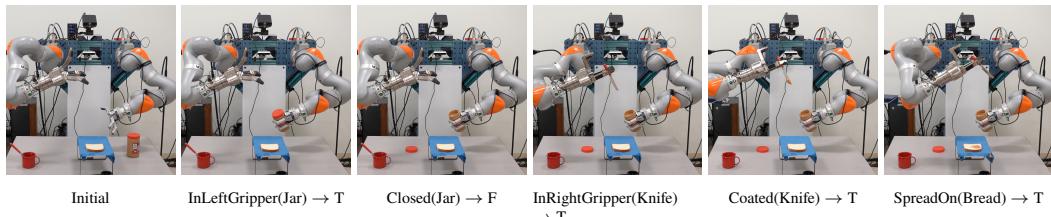


Figure 5: Sequence of Bimanual Robot Skill Execution with Predicate Value Changes

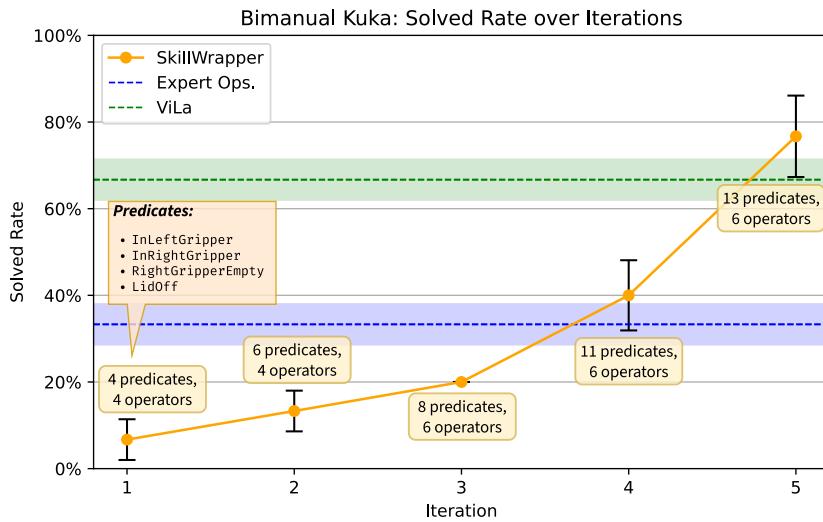


Figure 6: Bimanual Kuka Scenario Results over 5 iterations with invented predicate and learned operator count. As the number of predicates and operators grows, SKILLWRAPPER improves over the baseline methods.

**Discussion.** Our results demonstrate that SKILLWRAPPER is effective in real robot settings: our method generalizes skill representations learned in restricted domains to richer environments and progressively improves in more challenging scenarios with irreversible actions and interdependent skills. By outperforming both expert-defined operators and baseline methods, SKILLWRAPPER highlights the importance of predicate invention and iterative learning for scaling symbolic representations to embodied tasks.

## 5 RELATED WORKS

Our work uses pre-trained foundation models for learning symbolic representations of black-box robot skills useful to planning and close to human language and understanding. This work draws ideas from different fields of research such as model learning, abstraction learning, and task and motion planning (TAMP). Several methods have used foundation models (mainly LLMs) as high-level planners (Ahn et al., 2022; Rana et al., 2023; Driess et al., 2023). Several approaches have used foundation models as robot action models (Brohan et al., 2023; Shridhar et al., 2023) or to generate reward functions for robot tasks (Wang et al., 2024b). Concurrent work has also explored how representations can be learned directly from pixels (Athalye et al., 2025). Although these approaches show promising results for short-horizon single-skill problems, they fail to scale to complex long-horizon problems (Kambhampati et al., 2024). Lastly, multiple approaches (Han et al., 2024; Liang et al., 2025) have used foundation models to learn symbolic representations of robot skills, but require extensive feedback or prior knowledge from human experts. To the best of our knowledge, our work is the first to use a foundation model to automatically learn the human-interpretable symbolic characterization of robot skills with *theoretical guarantees*.

486 TAMP has long been used to solve complex robot tasks (Dantam et al., 2018; Shah et al., 2020;  
 487 Garrett et al., 2021). However, these approaches require symbolic models of the robot skills for task  
 488 planning. Various approaches have been developed to learn such symbolic models compatible with  
 489 TAMP solvers from high-dimensional inputs (Konidaris et al., 2018; Silver et al., 2023; Shah et al.,  
 490 2024). Additionally, the abstract representations learned through these methods are not human-  
 491 interpretable. On the other hand, we explicitly design our approach to work with high-dimensional  
 492 inputs and generate human-interpretable abstractions using pre-trained foundation models. We  
 493 consider abstractions as human-interpretable if they are semantically meaningful and use informative  
 494 language descriptions. SKILLWRAPPER also connects to other domains in robotics, and the full related  
 495 work can be found in Appendix I.

496

## 497 6 CONCLUSION

498

499 We characterize important properties of a learned symbolic model and present the first known  
 500 approach that employs off-the-shelf foundation models to invent symbolic representations for black-  
 501 box skills of an agent while providing strong guarantees of soundness and completeness of the  
 502 learned representations. By combining these theoretical guarantees with foundation model-driven  
 503 data collection and predicate evaluation, SKILLWRAPPER produces interpretable operators directly  
 504 usable by classical planners. Empirical results in a simulated burger domain and on real robots  
 505 demonstrate that SKILLWRAPPER enables efficient long-horizon planning without hand-engineered  
 506 abstractions, offering a principled path towards scalable skill reasoning.

507

## 508 7 REPRODUCIBILITY STATEMENT

509

510 To ensure reproducibility of our work, we provide source code and the prompts used in our experiments  
 511 as supplementary materials. Although the reproducibility of real-world robot experiments  
 512 is limited by hardware, simulation experiments run in Robotouille should be reliably reproduced,  
 513 granted that the checkpoints of the foundation model (i.e., OpenAI’s GPT-5 (OpenAI, 2025)) have  
 514 not been moved.

515

## 516 REFERENCES

517 Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea  
 518 Finn, et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In  
 519 *Proceedings of the 6th Conference on Robot Learning (CoRL)*, pp. 287–318, 14–18 Dec 2022.

520 Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montserrat Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, et al. AutoRT: Embodied Foundation Models for Large Scale  
 521 Orchestration of Robotic Agents. In *First Workshop on Vision-Language Models for Navigation and Manipulation (VLMNM) at ICRA 2024*, 2024.

522 Ankuj Arora, Humbert Fiorino, Damien Pellier, Marc Métivier, and Sylvie Pesty. A Review of  
 523 Learning Planning Action Models. *The Knowledge Engineering Review*, 33:e20, 2018.

524 Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Tomás Lozano-Pérez, and Leslie Pack  
 525 Kaelbling. Predicate Invention from Pixels via Pretrained Vision-Language Models. In *AAAI 2025 Workshop on Language Models for Planning (LM4Plan)*, 2025.

526 Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng,  
 527 Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei  
 528 Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng  
 529 Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan  
 530 Liu, Dunjie Lu, Rulin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang  
 531 Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang,  
 532 Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun  
 533 Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen  
 534 Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report.  
 535 *arXiv preprint arXiv:2511.21631*, 2025.

540 Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski,  
 541 Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-Language-Action  
 542 Models Transfer Web Knowledge to Robotic Control. In *Proceedings of the 7th Conference on*  
 543 *Robot Learning*, pp. 2165–2183, 06–09 Nov 2023.

544 Boyuan Chen, Zhuo Xu, Sean Kirmani, Brian Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei  
 545 Xia. SpatialVLM: Endowing Vision-Language Models with Spatial Reasoning Capabilities. In  
 546 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*,  
 547 pp. 14455–14465, 2024.

548 William Chen, Oier Mees, Aviral Kumar, and Sergey Levine. Vision-Language Models Provide  
 549 Promptable Representations for Reinforcement Learning. *Transactions on Machine Learning*  
 550 *Research (TMLR)*, 2025. ISSN 2835-8856.

552 Sijie Cheng, Zhicheng Guo, Jingwen Wu, Kechen Fang, Peng Li, Huaping Liu, and Yang Liu.  
 553 EgoThink: Evaluating First-Person Perspective Thinking Capability of Vision-Language Models.  
 554 In *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition*  
 555 (*CVPR*), pp. 14291–14302, 2024.

556 Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake,  
 557 and Shuran Song. Universal Manipulation Interface: In-The-Wild Robot Teaching Without In-  
 558 The-Wild Robots. In *Proceedings of Robotics: Science and Systems (RSS) XX*, 2024.

559 Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. An incremental  
 560 constraint-based framework for task and motion planning. *The International Journal of Robotics*  
 561 *Research*, 37(10):1134–1151, 2018.

562 S. Doncieux, D. Filliat, N. Díaz-Rodríguez, T. Hospedales, R. Duro, A. Coninx, D.M. Roijers,  
 563 B. Girard, N. Perrin, and O. Sigaud. Open-ended learning: a conceptual framework based on  
 564 representational redescription. *Frontiers in Neurorobotics*, 12:59, 2018.

565 Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter,  
 566 Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar,  
 567 Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc  
 568 Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An Embodied  
 569 Multimodal Language Model. In *Proceedings of the 40th International Conference on Machine*  
 570 *Learning (ICML)*, pp. 8469–8488, 23–29 Jul 2023.

571 Kuan Fang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Adaptive Procedural Task Generation for  
 572 Hard-Exploration Problems. In *Proceedings of the 9th International Conference on Learning*  
 573 *Representations (ICLR)*, 2021.

574 Kuan Fang, Toki Migimatsu, Ajay Mandlekar, Li Fei-Fei, and Jeannette Bohg. Active Task Ran-  
 575 domization: Learning Robust Skills via Unsupervised Generation of Diverse and Feasible Tasks.  
 576 *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems*  
 577 (*IROS*), pp. 1–8, 2022.

578 Kuan Fang, Fangchen Liu, Pieter Abbeel, and Sergey Levine. MOKA: Open-World Robotic Manip-  
 579 ulation through Mark-Based Visual Prompting. *Proceedings of Robotics: Science and Systems*  
 580 (*RSS*) XX, 2024.

581 Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack  
 582 Kaelbling, and Tomás Lozano-Pérez. Integrated Task and Motion Planning. *Annual Review of*  
 583 *Control, Robotics, and Autonomous Systems*, 4:265–293, 2021.

584 Gonzalo Gonzalez-Pumariega, Leong Su Yean, Neha Sunkara, and Sanjiban Choudhury. Robotouille:  
 585 An Asynchronous Planning Benchmark for LLM Agents. In *Proceedings of the 13th International*  
 586 *Conference on Learning Representations (ICLR)*, 2025.

587 Jiayuan Gu, Devendra Singh Chaplot, Hao Su, and Jitendra Malik. Multi-skill Mobile Manipulation  
 588 for Object Rearrangement. In *Proceedings of the 11th International Conference on Learning*  
 589 *Representations (ICML)*, 2022.

594 Huy Ha, Pete Florence, and Shuran Song. Scaling Up and Distilling Down: Language-Guided  
 595 Robot Skill Acquisition. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, pp.  
 596 3766–3777, 2023.

597 Muzhi Han, Yifeng Zhu, Song-Chun Zhu, Ying Nian Wu, and Yuke Zhu. InterPreT: Interactive  
 598 Predicate Learning from Language Feedback for Generalizable Task Planning. In *Proceedings of*  
 599 *Robotics: Science and Systems (RSS) XX*, 2024.

600 Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang  
 601 Gan. 3D-LLM: Injecting the 3D World into Large Language Models. In *Advances in Neural*  
 602 *Information Processing Systems (NeurIPS)*, volume 36, pp. 20482–20494, 2023.

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

605 Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language Models as Zero-Shot  
 606 Planners: Extracting Actionable Knowledge for Embodied Agents. In *Proceedings of the 39th*  
 607 *International Conference on Machine Learning (ICML)*, pp. 9118–9147, 2022.

608 Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan  
 609 Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda  
 610 Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied Reasoning  
 611 through Planning with Language Models. In *Proceedings of the 6th Conference on Robot Learning*  
 612 *(CoRL)*, pp. 1769–1782, 2023.

613 Hanxiao Jiang, Binghao Huang, Ruihai Wu, Zhuoran Li, Shubham Garg, Hooshang Nayyeri, Shen-  
 614 long Wang, and Yunzhu Li. RoboEXP: Action-Conditioned Scene Graph via Interactive Explo-  
 615 ration for Robotic Manipulation. In *Proceedings of the 8th Conference on Robot Learning*, pp.  
 616 3027–3052, 2025.

617 Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized Level Replay. In *Proceedings*  
 618 *of the 38th International Conference on Machine Learning (ICML)*, pp. 4940–4950. PMLR, 2021.

619 Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and  
 620 Ross Girshick. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual  
 621 Reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*,  
 622 pp. 2901–2910, 2017.

623 Brendan Juba, Hai S. Le, and Roni Stern. Safe Learning of Lifted Action Models. In *Proceedings*  
 624 *of the 18th International Conference on Principles of Knowledge Representation and Reasoning*  
 625 *(KR)*, pp. 379–389, 11 2021.

626 Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant  
 627 Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: LLMs Can't Plan, But Can Help  
 628 Planning in LLM-Modulo Frameworks. In *Proceedings of the 41st International Conference on*  
 629 *Machine Learning (ICML)*, 2024.

630 Michael Katz and Junkyu Lee. K\* and partial order reduction for top-quality planning. In *Proceedings*  
 631 *of the 16th Annual Symposium on Combinatorial Search (SoCS 2023)*. AAAI Press, 2023.

632 George Konidaris. On the Necessity of Abstraction. *Current Opinion in Behavioral Sciences*, 29:  
 633 1–7, 2019. ISSN 2352-1546.

634 George Konidaris and Andrew Barto. Skill Discovery in Continuous Reinforcement Learning  
 635 Domains using Skill Chaining. In *Advances in Neural Information Processing Systems (NIPS)*,  
 636 volume 22, 2009.

637 George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Pérez. From Skills to Symbols: Learn-  
 638 ing Symbolic Representations for Abstract High-Level Planning. *Journal of Artificial Intelligence*  
 639 *Research*, 61:215–289, 2018.

640 Leonardo Lamanna, Luciano Serafini, Mohamadreza Faridghasemnia, Alessandro Saffiotti, Alessan-  
 641 dro Saetti, Alfonso Gerevini, and Paolo Traverso. Planning for Learning Object Properties. *Pro-*  
 642 *ceedings of the AAAI Conference on Artificial Intelligence*, 37(10):12005–12013, Jun. 2023.

648 Amber Li and Tom Silver. Embodied Active Learning of Relational State Abstractions for Bilevel  
 649 Planning. In *Proceedings of The 2nd Conference on Lifelong Learning Agents (CoLLAs)*, pp.  
 650 358–375, 2023.

651 Zhaoyi Li, Kelin Yu, Shuo Cheng, and Danfei Xu. LEAGUE++: Empowering Continual Robot  
 652 Learning via Guided Skill Acquisition with Large Language Models. In *ICLR 2024 Workshop on*  
 653 *Large Language Model (LLM) Agents*, 2024.

654 Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B. Tenenbaum, Tom Silver, Joao F.  
 655 Henriques, and Kevin Ellis. VisualPredicator: Learning Abstract World Models with Neuro-  
 656 Symbolic Predicates for Robot Planning. In *Proceedings of the 13th International Conference on*  
 657 *Learning Representations (ICLR)*, 2025.

658 Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff,  
 659 Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li,  
 660 Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk,  
 661 Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Alexander Sax, and Aravind  
 662 Rajeswaran. OpenEQA: Embodied Question Answering in the Era of Foundation Models. In  
 663 *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition*  
 664 (*CVPR*), pp. 16488–16498, 2024.

665 Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The Neuro-  
 666 Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision.  
 667 In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

668 D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins.  
 669 PDDL – The Planning Domain Definition Language. Technical report, CVC TR-98-003/DCS  
 670 TR-1165, Yale Center for Computational Vision and Control, 1998.

671 Toki Migimatsu and Jeannette Bohg. Grounding Predicates through Actions. In *Proceedings of the*  
 672 *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3498–3504, 2022.

673 Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhui Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng  
 674 Dai, Yu Qiao, and Ping Luo. EmbodiedGPT: Vision-Language Pre-Training via Embodied Chain  
 675 of Thought. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.

676 Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie,  
 677 Danny Driess, Ayzaan Wahid, Zhuo Xu, Quan Vuong, Tingnan Zhang, Tsang-Wei Edward Lee,  
 678 Kuang-Huei Lee, Peng Xu, Sean Kirmani, Yuke Zhu, Andy Zeng, Karol Hausman, Nicolas  
 679 Heess, Chelsea Finn, Sergey Levine, and Brian Ichter. PIVOT: Iterative Visual Prompting Elicits  
 680 Actionable Knowledge for VLMs. In *Proceedings of the 41st International Conference on Machine*  
 681 *Learning (ICML)*, 2024.

682 OpenAI. Introducing GPT-5, 2025. URL <https://openai.com/index/introducing-gpt-5/>. Accessed: December 4, 2025.

683 Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Ray Mooney, Stefanie Tellex,  
 684 and David Paulius. CAPE: Corrective Actions from Precondition Errors using Large Language  
 685 Models. In *Proceedings of the 2024 IEEE International Conference on Robotics and Automation*  
 686 (*ICRA*), pp. 14070–14077, 2024.

687 Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf.  
 688 SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task  
 689 Planning. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, volume 229, pp.  
 690 23–72, 06–09 Nov 2023.

691 Allen Z. Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh.  
 692 Explore until Confident: Efficient Exploration for Embodied Question Answering. In *Proceedings*  
 693 *of Robotics: Science and Systems (RSS) XX*, 2024.

694 Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan,  
 695 Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J Joshi, et al. RoboVQA:  
 696 Multimodal Long-Horizon Reasoning for Robotics. In *Proceedings of the 2024 IEEE International*  
 697 *Conference on Robotics and Automation (ICRA)*, pp. 645–652. IEEE, 2024.

702 Naman Shah, Deepak Kala Vasudevan, Kislay Kumar, Pranav Kamojjhala, and Siddharth Srivastava.  
 703 Anytime Integrated Task and Motion Policies for Stochastic Environments. In *Proceedings of the*  
 704 *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9285–9291. IEEE,  
 705 2020.

706 Naman Shah, Jayesh Nagpal, Pulkit Verma, and Siddharth Srivastava. From Reals to Logic and  
 707 Back: Inventing Symbolic Vocabularies, Actions and Models for Planning from Raw Data. *arXiv*  
 708 *preprint arXiv:2402.11871*, 2024.

709

710 C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27  
 711 (3):379–423, 1948.

712 Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-Actor: A Multi-Task Transformer for  
 713 Robotic Manipulation. In *Proceedings of the 6th Conference on Robot Learning*, volume 205, pp.  
 714 785–799, 14–18 Dec 2023.

715

716 Tom Silver, Rohan Chitnis, Nishanth Kumar, Willie McClinton, Tomás Lozano-Pérez, Leslie Kael-  
 717 bling, and Joshua B. Tenenbaum. Predicate Invention for Bilevel Planning. *Proceedings of the*  
 718 *AAAI Conference on Artificial Intelligence*, 37(10):12120–12129, Jun. 2023.

719

720 Theodore Sumers, Kenneth Marino, Arun Ahuja, Rob Fergus, and Ishita Dasgupta. Distilling  
 721 Internet-Scale Vision-Language Models into Embodied Agents. In *Proceedings of the Fortieth*  
 722 *International Conference on Machine Learning (ICML)*, pp. 32797–32818, 2023.

723

724 Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual Inference via Python Execution  
 725 for Reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*  
 (ICCV), pp. 11888–11898, October 2023.

726

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

729

730 Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah  
 731 Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan,  
 732 Vladimír Vondruš, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira,  
 733 Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training Home  
 734 Assistants to Rearrange their Habitat. In *Advances in Neural Information Processing Systems*  
 (NeurIPS), volume 34, pp. 251–266, 2021.

735

736 Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the  
 737 Planning Abilities of Large Language Models - A Critical Investigation. In *Advances in Neural*  
 738 *Information Processing Systems (NeurIPS)*, volume 36, pp. 75993–76005, 2023.

739

740 Pulkit Verma, Shashank Rao Marpally, and Siddharth Srivastava. Discovering User-Interpretable  
 741 Capabilities of Black-Box Planning Agents. In *Proceedings of the 19th International Conference*  
 742 *on Principles of Knowledge Representation and Reasoning (KR)*, volume 19, pp. 362–372, 2022.

743

744 Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and  
 745 Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models.  
 746 *Transactions on Machine Learning Research (TMLR)*, 2024a. ISSN 2835-8856.

747

748 Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired Open-Ended Trailblazer  
 749 (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and  
 750 Their Solutions. *arXiv preprint arXiv:1901.01753*, 2019.

751

752 Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson.  
 753 RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback. In  
 754 *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 51484–  
 755 51501, 21–27 Jul 2024b.

756

757 Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zack-  
 758 ery Erickson, David Held, and Chuang Gan. RoboGen: Towards Unleashing Infinite Data for  
 759 Automated Robot Learning via Generative Simulation. In *Proceedings of the 41st International*  
 760 *Conference on Machine Learning (ICML)*, 2024c.

756 Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6D Pose Estimation  
757 and Tracking of Novel Objects. In *Proceedings of the 2024 IEEE/CVF Conference on Computer*  
758 *Vision and Pattern Recognition (CVPR)*, pp. 17868–17879, 2024.

759  
760 Kai Xi, Stephen Gould, and Sylvie Thiébaux. Neuro-Symbolic Learning of Lifted Action Models  
761 from Visual Traces. *Proceedings of the International Conference on Automated Planning and*  
762 *Scheduling (ICAPS)*, 34(1):653–662, May 2024.

763 Jingkang Yang, Yuhao Dong, Shuai Liu, Bo Li, Ziyue Wang, Haoran Tan, Chencheng Jiang, Jiamu  
764 Kang, Yuanhan Zhang, Kaiyang Zhou, et al. Octopus: Embodied Vision-Language Programmer  
765 from Environmental Feedback. In *Proceedings of the 2024 European Conference on Computer*  
766 *Vision (ECCV)*, pp. 20–38, 2024.

767  
768 Naoki Yokoyama, Alex Clegg, Joanne Truong, Eric Undersander, Tsung-Yen Yang, Sergio Arnaud,  
769 Sehoon Ha, Dhruv Batra, and Akshara Rai. ASC: Adaptive Skill Coordination for Robotic Mobile  
770 Manipulation. *IEEE Robotics and Automation Letters*, 9(1):779–786, 2024.

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

## APPENDIX

## A FORMAL FRAMEWORK

We consider a problem setting in which an agent is equipped with a set of predefined, “black-box” skills. The agent can evaluate whether a skill is executable in the current state, but it does not have a complete transition model of the skills a priori, and therefore cannot compose its skills to solve long-horizon problems without considering low-level details. However, if the agent were to learn symbolic models of its skills, it could use classical planning to efficiently compose them to solve new tasks. In this section, we formalize this setting as the problem of *Skill Model Learning*.

## A.1 PRELIMINARIES

**Environment Model.** We define an *environment* as a tuple  $(\mathcal{S}, \mathcal{T}, \Omega, T)$ , where the *state space*  $\mathcal{S}$  is assumed to be high-dimensional, continuous, and fully observable; the *type set*  $\mathcal{T}$  enumerates the possible object types; and the *skills*  $\Omega$  are object-centric, such that each  $\omega \in \Omega$  is parameterized by object types drawn from the *type set*  $\mathcal{T}$ . The *transition function*  $T : \mathcal{S} \times \Omega \rightarrow \mathcal{S}$  characterizes the environment dynamics but is unknown to the agent. For example, `Pour(?teapot, ?mug)` may be used to pour tea from a teapot into a mug. However, certain environmental aspects (e.g., which mugs are available to pour into) may differ between settings during learning. We therefore define a specific *setting* by the tuple  $(s_0, \mathcal{O})$ , specifying an initial state  $s_0 \in \mathcal{S}$  and a set of typed objects  $\mathcal{O}$ , where the type(s) of object  $o \in \mathcal{O}$  are denoted by  $\tau_o \subseteq \mathcal{T}$ .

**Black-box Skills.** We model skills as object-centric options (Sutton et al., 1999) with discrete, object-typed parameters. Formally, a *skill*  $\omega \in \Omega$  is defined by a tuple  $(\mathcal{I}_\omega, \pi_\omega, \beta_\omega, \Theta_\omega)$ , where the *initiation set*  $\mathcal{I}_\omega \subseteq \mathcal{S}$  contains states from which the skill may be executed; the *policy*  $\pi_\omega$  controls the agent during the skill; the *termination set*  $\beta_\omega \subseteq \mathcal{S}$  is the set of states at which the skill immediately terminates; and the *skill parameters*  $\Theta_\omega = (\theta_\omega^1, \dots, \theta_\omega^k)$  specify type constraints  $\tau_{\theta_\omega^i} \subseteq \mathcal{T}$  for valid skill arguments. Specifically, a skill  $\omega \in \Omega$  may be instantiated to create a *skill instance*  $\underline{\omega}$  using objects  $o_{1:k} \in \mathcal{O}^k$  if and only if for  $1 \leq i \leq k$ ,  $\tau_{\theta_\omega^i} \subseteq \tau_{o_i}$ . We assume that executing a skill only affects the state of objects passed as arguments.<sup>1</sup>

**Symbolic Abstractions.** Because the agent must evaluate  $\mathcal{I}_\omega$  on individual states, it cannot distinguish essential skill information from irrelevant details, making long-horizon planning combinatorially difficult. However, each skill affects only a few objects at once, leaving the rest of the world unchanged. The agent can exploit this property by instead learning a factored state representation—formalized here as first-order logic models in PDDL (McDermott et al., 1998)—providing an abstract transition model of its skills. Such abstractions enable the agent to use classical planning to compose its skills and accomplish unseen, long-horizon goals.

We use symbolic *predicates*  $\mathcal{P}$  to express abstract relations between objects. Each predicate  $\sigma \in \mathcal{P}$  is a tuple  $(C_\sigma, \Theta_\sigma)$ , where the *predicate classifier*  $C_\sigma$  tests whether the predicate holds in a state, given a binding of objects to the *predicate parameters*  $\Theta_\sigma = (\theta_\sigma^1, \dots, \theta_\sigma^n)$ . Each parameter  $\theta_\sigma^i$  specifies a type constraint  $\tau_{\theta_\sigma^i} \subseteq \mathcal{T}$  on corresponding object arguments. Formally, the partial function<sup>2</sup>  $C_\sigma : \mathcal{O}^n \rightharpoonup (\mathcal{S} \rightarrow \{0, 1\})$  is defined on objects  $o_{1:n} \in \mathcal{O}^n$  if and only if  $\tau_{\theta_\sigma^i} \subseteq \tau_{o_i}$  for  $1 \leq i \leq n$ . In such cases, we say that the objects  $o_{1:n}$  are *valid arguments* for  $\sigma$ .

By observing different effects, the agent must construct a set of *operators*  $\mathcal{A}$  that define abstract transition models for the agent’s skills. We define each operator  $a \in \mathcal{A}$  by  $(\omega_a, \Theta_a, \text{PRE}, \text{EFF}^+, \text{EFF}^-)$ , where  $\omega_a \in \Omega$  is the corresponding skill; the *operator parameters*  $\Theta_a = (\theta_a^1, \dots, \theta_a^m)$  impose type constraints  $\tau_{\theta_a^i} \subseteq \mathcal{T}$  on operator arguments; the *preconditions*  $\text{PRE}$  are a conjunction of literals over  $\mathcal{P}$  defining the conditions necessary to apply the operator; and the *add* and *delete effects*,  $\text{EFF}^+$  and  $\text{EFF}^-$ , are the subsets of  $\mathcal{P}$  that become true and false, respectively, after the operator is applied.

**Grounded Abstractions.** To apply an abstract transition model to a real-world setting, the agent must map the low-level state to an abstract state, which requires *grounding* the known predicates

<sup>1</sup>We do not, however, assume that a skill necessarily affects the state of *all* objects passed in as arguments.

<sup>2</sup>We denote a partial function using  $f : \mathcal{X} \rightharpoonup \mathcal{Y}$ .

864 using concrete objects so that their truth value may be determined. A predicate  $\sigma \in \mathcal{P}$  may  
 865 only be grounded using valid arguments, inducing a *grounded predicate*  $\underline{\sigma} = \sigma(o_{1:n})$ . Given  
 866 a low-level state, the classifier  $C_{\sigma} : \mathcal{S} \rightarrow \{0, 1\}$  tests whether the predicate holds for the object  
 867 arguments. We define the set of *grounded predicates* for a setting  $(s_0, \mathcal{O})$  under predicates  $\mathcal{P}$  as  
 868  $\mathcal{P} = \{\sigma(o_{1:n}) : \sigma \in \mathcal{P}, o_{1:n} \in \mathcal{O}^n, \bigwedge_{i=1}^n t_{\theta_{\sigma}^i} \subseteq t_{o_i}\}$ , inducing an *abstract state space*  $\underline{\mathcal{S}} = 2^{\mathcal{P}}$   
 869 where each abstract state corresponds to a specific combination of grounded relations.

870 Given a set of operators  $\mathcal{A}$  in a setting  $(s_0, \mathcal{O})$ , the *abstract action space*  $\underline{\mathcal{A}}$  is the set of all valid  
 871 groundings of the operators using objects from  $\mathcal{O}$ . Each *abstract action* (i.e., grounded operator) is  
 872 defined as  $\underline{a} = a(o_{1:m})$ , where  $a \in \mathcal{A}$ ,  $o_{1:m} \in \mathcal{O}^m$ , and for  $1 \leq i \leq m$ ,  $t_{\theta_a^i} \subseteq t_{o_i}$ . Grounding  
 873 an operator induces *ground preconditions*  $\underline{\text{PRE}}$ , which are a conjunction of ground literals over  $\underline{\mathcal{P}}$ ;  
 874 *ground add effects*  $\underline{\text{EFF}}^+ \subseteq \underline{\mathcal{P}}$ ; and *ground delete effects*  $\underline{\text{EFF}}^- \subseteq \underline{\mathcal{P}}$ .

875 The *grounding function*  $\mathcal{G} : \mathcal{P} \rightarrow 2^{\mathcal{S}}$  maps each grounded predicate to its *grounding set*  $\mathcal{G}(\underline{\sigma}) \subseteq \mathcal{S}$ ,  
 876 defined as  $\mathcal{G}(\underline{\sigma}) = \{s \in \mathcal{S} : C_{\sigma}(s) = 1\}$ . We overload this notation for abstract states  $\underline{s} \in \underline{\mathcal{S}}$  so that  
 877  $\mathcal{G}(\underline{s}) = \bigcap_{\sigma_i \in \underline{s}} \mathcal{G}(\underline{\sigma}_i)$ . Conversely, the *abstraction function*  $\text{ABSTRACT} : \mathcal{S} \rightarrow \underline{\mathcal{S}}$  maps each low-level  
 878 state  $s \in \mathcal{S}$  to the abstract state  $\underline{s} \in \underline{\mathcal{S}}$  defined by  $\text{ABSTRACT}(s) = \{\underline{\sigma} \in \underline{\mathcal{P}} : C_{\sigma}(s) = 1\}$ .  
 879

## 880 A.2 PROBLEM DEFINITION

882 **Definition 1.** Given an environment  $(\mathcal{S}, \mathcal{T}, \Omega, T)$  containing settings  $\{(s_0, \mathcal{O})\}_{i=1}^N$ , we define a Skill  
 883 Model Learning problem as learning an abstract transition model  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$  for the skills  $\Omega$ .

885 After a period of continual learning in one or more settings, an agent may be evaluated on an *skill*  
 886 *planning problem*  $\mathbf{p} = (s_0, \mathcal{O}, \mathcal{S}_g)$ , where  $\mathcal{S}_g \subseteq \mathcal{S}$  is the set of *goal states* to be reached. Given a  
 887 model  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$ , a classical planner can be used to search for an *abstract plan*  $[\underline{a}_1, \dots, \underline{a}_n]$  that  
 888 solves the *abstract planning problem*  $\mathbf{p} = (\underline{s}_0, \mathcal{O}, \underline{\mathcal{S}}_g)$ .

889 **Definition 2.** An abstract plan  $[\underline{a}_1, \dots, \underline{a}_n]$  is called a *solution* for skill planning problem  
 890  $\mathbf{p} = (s_0, \mathcal{O}, \mathcal{S}_g)$  iff for  $1 \leq i \leq n$ ,  $s_i = T(s_{i-1}, \underline{\omega}_i)$ ,  $s_{i-1} \in \mathcal{I}_{\underline{\omega}_i}$ , and  $s_n \in \mathcal{S}_g$ , where  $\underline{\omega}_i = \underline{\omega}_{\underline{a}_i}$ .

891 In Sec. 3, we describe how SKILLWRAPPER constructs  $\mathcal{M}$  from raw skill executions.

## 893 B ALGORITHMS

### 894 B.1 SKILL SEQUENCE PROPOSAL

897 Each skill sequence  $\sigma = [\underline{\omega}_1, \dots, \underline{\omega}_m]$  (Section 3.1) proposed by the foundation model is scored  
 898 using two heuristics: coverage ( $C$ ) and chainability ( $Ch$ ). This section provides more details on how  
 899 these heuristics are computed and algorithmically used to assign scores to each sequence.  
 900

901 **Overview.** We prompt a foundation model is provided with the agent’s skill set  $\Omega$  and the current  
 902 set of abstract predicates  $\mathcal{P}$  to generate and propose sequences of skills  $\sigma$ , with which the agent  
 903 collects a dataset of transitions  $\mathcal{D}$ . The skill sequence proposal procedure (Algorithm 2) assigns a  
 904 score tuple  $(C, Ch)$  to all sequences and maintains a subset of pareto-front sequences that cannot  
 905 strictly dominate another sequence, i.e.,  $(C_i < C_j) \vee (Ch_i < Ch_j) \vee (C_i \leq C_j \wedge Ch_i \leq Ch_j)$   
 906 where  $i \neq j$ . An output skill sequence is finally chosen from this pareto-front subset.

907 **Coverage ( $C$ ).** Coverage (Algorithm 3) evaluates the information gain on all possible pairs of  
 908 consecutively executed skills over existing transitions after executing a new skill sequence. Specifi-  
 909 cally, the information gain is measured by the increase in Shannon entropy (Shannon, 1948) over the  
 910 distribution of all consecutive skill pairs resulting from executing the proposed skill sequence  
 911

$$912 C = \frac{\mathcal{Q}'}{\Sigma \mathcal{Q}'} \times \log\left(\frac{\mathcal{Q}'}{\Sigma \mathcal{Q}'}\right) - \frac{\mathcal{Q}}{\Sigma \mathcal{Q}} \times \log\left(\frac{\mathcal{Q}}{\Sigma \mathcal{Q}}\right) \quad (1)$$

914 where  $\mathcal{Q}$  and  $\mathcal{Q}'$  are matrices that tabulate the number of pairs of consecutively executed skills  
 915 that occur before and after executing a new skill sequence, respectively. Maximizing coverage  
 916 encourages the generation of proposed skill sequences that contain the least explored skill pairs.  
 917 More importantly, this would allow our method to uncover a larger set of interdependencies across  
 918 the preconditions and effects of all skills.

918 **Chainability (Ch).** Chainability predicts the ratio of successful to failed pairs of consecutively  
 919 executed skills. By computing chainability, we estimate the degree to which the preconditions of  
 920 operators learned in each iteration are satisfied, and executability can be inferred from the estimated  
 921 symbolic states and the operators. With an appropriate chainability score, the collected dataset  
 922 of skill execution traces maintains a balance between number of successful executions and failure  
 923 executions, which is ideal for identifying possible mismatched pairs and thus inventing predicates.  
 924  
 925

---

**Algorithm 2** Propose Skill Sequences
 

---

```

927 1: Input: Skill set  $\Omega$ , skill execution traces  $\mathcal{D}$ , predicate set  $\mathcal{P}$ , operator set  $\mathcal{A}$ , batch size  $n$ 
928 2: Output: Proposed skill sequence  $\sigma$ 
929 3:  $\text{seq\_batch} \leftarrow \text{GENERATESKILLSEQUENCES}(\Omega, n)$  ▷ Propose a batch of skill sequences with FM
930 4:  $\text{Scores} \leftarrow \{\}$ 
931 5: for  $\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle$  in  $\text{seq\_batch}$  do
932 6:    $\text{cov} \leftarrow \text{COVERAGE}(\mathcal{D}, \langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle)$ 
933 7:    $\text{chain} \leftarrow \text{CHAINABILITY}(\mathcal{A}, \langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle)$ 
934 8:    $\text{Scores}[\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle] \leftarrow (\text{cov}, \text{chain})$ 
935 9: end for
10: return  $\text{PARETOOPTIMAL}(\text{Scores})$ 

```

---

**Algorithm 3** Coverage
 

---

```

941 1: Input: Skill execution traces  $\mathcal{D}$ , proposed skill sequence  $\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle$ 
942 2: Output: Coverage score  $C$ 
943 3:  $\mathcal{Q} \leftarrow \text{zero matrix of size } |\Omega| \times |\Omega|$  ▷ Construct a matrix of skill-pair counts
944 4: for  $\langle s_i, \underline{\omega}_i, s'_i \rangle, \langle s_{i+1}, \underline{\omega}_{i+1}, s'_{i+1} \rangle$  in  $\mathcal{D}$  do ▷ Iterate over all consecutive pairs of transitions
945 5:    $\mathcal{Q}[\underline{\omega}_i, \underline{\omega}_{i+1}] = \mathcal{Q}[\underline{\omega}_i, \underline{\omega}_{i+1}] + 1$ 
946 6: end for
947 7:  $\mathcal{Q}' \leftarrow \mathcal{Q}$  ▷ New skill-pair count initialized
948 8: for  $\langle \underline{\omega}_i, \underline{\omega}_{i+1} \rangle$  in  $\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle$  do
949 9:    $\mathcal{Q}'[\underline{\omega}_i, \underline{\omega}_{i+1}] = \mathcal{Q}'[\underline{\omega}_i, \underline{\omega}_{i+1}] + 1$ 
950 10: end for
951 11:  $\text{cov} \leftarrow \text{COVERAGE}(\mathcal{Q}') - \text{COVERAGE}(\mathcal{Q})$  ▷ Compute coverage score using Eq. 1
952 12: return  $\text{cov}$ 

```

---

**Algorithm 4** Chainability
 

---

```

956 1: Input: Operator set  $\mathcal{A}$ , Proposed skill sequence  $\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle$ 
957 2: Output: Chainability score  $chain$ 
958 3:  $\text{exec\_count} \leftarrow 0$  ▷ Total number of executable skills
959 4:  $\text{sequence\_length} \leftarrow \text{LENGTH}(\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle)$ 
960 5:  $\text{seq} \leftarrow [s_0]$  ▷ Store the trace of after-execution state
961 6: for  $\underline{\omega}_i$  in  $\langle \underline{\omega}_1, \dots, \underline{\omega}_k \rangle$  do
962 7:   for  $a \in \mathcal{A}_{\underline{\omega}_i}$  do
963 8:     if  $\Gamma(\text{seq}[-1]) \models \text{PRE}_a$  then ▷ Successful execution predicated by the current model
964 9:        $\text{exec\_count} = \text{exec\_count} + 1$ 
965 10:      break
966 11:    end if
967 12:   end for
968 13:    $s_{new} \leftarrow \text{APPLYOPERATOR}(\text{seq}[-1].a)$  ▷ Calculate the abstract state after execution
969 14:    $\text{seq} \leftarrow \text{seq} \parallel \mathcal{G}(s_{new})$  ▷ Append current low-level state to the trace
970 15: end for
971 16:  $chain \leftarrow |\text{exec\_count}/\text{sequence\_length} - 0.5|$ 
972 17: return  $chain$ 

```

---

972 B.2 PREDICATE INVENTION

973

974

975

976

977

978 **Algorithm 5** Invent Predicates

---

 979  
 980 1: **Input:** Skill set  $\Omega$ , skill execution traces  $\mathcal{D}(\omega) = \{\langle s, \underline{\omega}, s' \rangle\}_{\omega}$ , predicate set  $\mathcal{P}$ , operator set  $\mathcal{A}$ .  
 981 2: **Output:** Predicate set  $\mathcal{P}$   
 982 3: **for**  $\omega \in \Omega$  **do**  
 983 4:     **while**  $\exists \langle s_i, \underline{\omega}_i, s'_i \rangle, \langle s_j, \underline{\omega}_j, s'_j \rangle \in \mathcal{D}$  s.t.  $s_i \in \alpha_{\underline{\omega}_i}, s_j \in \alpha_{\underline{\omega}_j}$ , but  $s_i \in \mathcal{I}_{\underline{\omega}_i}, s_j \notin \mathcal{I}_{\underline{\omega}_j}$  **do**  
 984 5:          $\sigma \leftarrow \text{NEWPREDICATE}$   
 985 6:          $\mathcal{P} \leftarrow \mathcal{P} \parallel \sigma$  **if**  $\text{SCOREPRECOND}(\sigma, \mathcal{P}, \omega, \mathcal{D})$   
 986 7:         **end while**  
 987 8:         **while**  $\exists \langle s_i, \underline{\omega}_i, s'_i \rangle, \langle s_j, \underline{\omega}_j, s'_j \rangle \in \mathcal{D}$  s.t.  $s'_i \in \zeta_{\underline{\omega}_i}, s'_j \in \zeta_{\underline{\omega}_j}$ , but  $s_i \in \mathcal{I}_{\underline{\omega}_i}, s_j \notin \mathcal{I}_{\underline{\omega}_j}$  **do**  
 988 9:          $\sigma \leftarrow \text{NEWPREDICATE}$   
 989 10:          $\mathcal{P} \leftarrow \mathcal{P} \parallel \sigma$  **if**  $\text{SCOREEFF}(\sigma, \mathcal{P}, \omega, \mathcal{D})$   
 990 11:         **end while**  
 991 12:     **end for**  
 992 13:     **return**  $\mathcal{P}$   
 993  
 994  
 995  
 996  
 997
998 **Algorithm 6** Scoring Functions for Invented Predicates

---

 999  
 1000 1: **Input:** New predicate  $\sigma$ , existing predicate set  $\mathcal{P}$ , skill  $\omega$ , and skill execution traces  $\mathcal{D}$ .  
 1001 2: **Parameters:** Threshold  $h$   
 1002  
 1003 3: **ScorePrecond:**  
 1004 4:  $\mathcal{P}' \leftarrow \mathcal{P} \cup \{\sigma\}$   
 1005 5:  $\mathcal{A}' \leftarrow \text{LEARNOPERATORS}(\mathcal{D}, \mathcal{P}')$  ▷ Hypothetical operators after including  $\sigma$   
 1006 6: total  $\leftarrow 0$   
 1007 7: valid  $\leftarrow 0$   
 1008 8: **for**  $\langle s, \underline{\omega}, s' \rangle \in \mathcal{D}$  **do**  
 1009 9:     **if**  $\exists a' \in \mathcal{A}', \mathbf{o} \subseteq \mathcal{O}$ , s.t.  $s \in \text{PRE}_{a'}$ , and  $s \in \mathcal{I}_{\underline{\omega}}$  **then**  
 1010 10:         valid = valid + 1  
 1011 11:     **end if**  
 1012 12:     total = total + 1  
 1013 13: **end for**  
 1014 14: **return** valid/total  $> h$   
 1015  
 1016 15: **ScoreEff:**  
 1017 16:  $\mathcal{P}' \leftarrow \mathcal{P} \cup \{\sigma\}$   
 1018 17:  $\mathcal{A}' \leftarrow \text{LEARNOPERATORS}(\mathcal{D}, \mathcal{P}')$   
 1019 18: total  $\leftarrow 0$   
 1020 19: valid  $\leftarrow 0$   
 1021 20: **for**  $\langle s, \underline{\omega}, s' \rangle \in \mathcal{D}$  **do**  
 1022 21:     **if**  $\exists a' \in \mathcal{A}', \mathbf{o} \subseteq \mathcal{O}$ , s.t.  $\Gamma_{\mathcal{P}'}(s') \setminus \Gamma_{\mathcal{P}'}(s) = \text{EFF}_{a'}$ , and  $s \in \mathcal{I}_{\underline{\omega}}$  **then**  
 1023 22:         valid = valid + 1  
 1024 23:     **end if**  
 1025 24:     total = total + 1  
 1026 25: **end for**  
 1027 26: **return** valid/total  $> h$   
 1028  
 1029  
 1030

1026 B.3 OPERATOR LEARNING  
10271028 **Algorithm 7** Learn Operators  
1029

---

```

1030 1: Input: Skill execution traces  $\mathcal{D}(\omega) = \{\langle s, \underline{\omega}, s' \rangle\}_{\omega}$ , predicate set  $\mathcal{P}$ 
1031 2: Output: Operator set  $\mathcal{A}$ 
1032 3:  $\text{eff\_dict} \leftarrow \text{defaultdict}()$  ▷ Store clustered effects
1033 4: for  $\langle s, \underline{\omega}, s' \rangle \in \mathcal{D}$  do
1034 5:    $\text{eff} \leftarrow \Gamma(s') \setminus \Gamma(s)$ 
1035 6:    $\text{eff\_dict}[\text{eff}] \leftarrow \text{eff\_dict}[\text{eff}] \parallel (s, s')$ 
1036 7: end for
1037 8:  $\mathcal{A} \leftarrow []$ 
1038 9: for  $\text{eff} \in \text{eff\_dict}$  do
1039 10:    $\text{execution\_list} \leftarrow \text{eff\_dict}[\text{eff}]$ 
1040 11:    $\text{precond} \leftarrow \prod_{\langle s, \underline{\omega}, s' \rangle \in \text{execution\_list}} \Gamma(s)$ 
1041 12:    $\mathcal{A} \leftarrow \mathcal{A} \parallel [\text{precond}, \text{eff}]$ 
1042 13: end for
1043 14: return  $\mathcal{A}$ 

```

---

1042

1043

## C PROPERTIES OF LEARNED SYMBOLIC MODELS

1044

1045

Relational predicates are the basic units of the abstract representation of the low-level state space. In this section, we characterize the conditions of the learned representations of a finite set of skills using relational predicates to support high-level planning, in the context of model learning. From here, chaining the skills is enabled by applying predicates from the representation of each low-level skill to others for clustering (described in Section 3.3).

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

Effective skill planning requires an accurate abstract model and grounding function. All forms of abstractions are typically lossy, i.e., while learning an abstract transition model, certain low-level environment details may not be captured. Conversely, the learned model must accurately retain the information needed to produce sound and complete plans. In this section, we characterize the conditions under which an abstract model facilitates exact, sound, and/or complete planning, for the purpose of constraining how we can *construct* such a model.

To begin formalizing the relationship between a skill  $\omega \in \Omega$  and some abstract model  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$ , we define two sets,  $\alpha_{\omega}$  and  $\zeta_{\omega}$ , representing the states in which the model predicts that the skill may either be initiated (when  $s \in \alpha_{\omega}$ ) or terminated (when  $s \in \zeta_{\omega}$ ), respectively.

**Definition 3.** *Given a skill instance  $\underline{\omega} \in \underline{\Omega}$  and abstract model  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$ , we define  $\alpha_{\underline{\omega}}$  and  $\zeta_{\underline{\omega}}$  as follows:*

$$\alpha_{\underline{\omega}} = \bigcup_{\underline{a} \in \underline{\mathcal{A}}_{\underline{\omega}}} \mathcal{G}(\underline{P}_{RE_{\underline{a}}}) \quad (2)$$

$$\zeta_{\underline{\omega}} = \bigcup_{\underline{a} \in \underline{\mathcal{A}}_{\underline{\omega}}} \mathcal{G}((\underline{P}_{RE_{\underline{a}}} \setminus \underline{E}_{FF_{\underline{a}}^-}) \cup \underline{E}_{FF_{\underline{a}}^+}) \quad (3)$$

1062

1063

1064

1065

1066

1067

For a non-instantiated skill  $\omega \in \Omega$ , we define  $\alpha_{\omega} = \bigcup_{\underline{\omega}} \alpha_{\underline{\omega}}$  and  $\zeta_{\omega} = \bigcup_{\underline{\omega}} \zeta_{\underline{\omega}}$ .

1068

1069

1070

1071

1072

1073

1074

1075

We define an exact abstract model as one that perfectly captures the initiation set and termination set of all skills. Although such a representation is infeasible to learn in practice, its properties provide an “ideal case” from which other definitions can weaken assumptions.

**Definition 4 (Exact Model).** *Let  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$  be a model for environment  $(\mathcal{S}, \mathcal{T}, \Omega, T)$  and the set of skills  $\Omega$ , and let  $\alpha_{\omega}$  and  $\zeta_{\omega}$  be the approximate initiation and termination sets (Def. 3). The model  $\mathcal{M}$  is an exact model iff:*

$$\forall \omega \in \Omega, s \in \mathcal{S}: s \in \mathcal{I}_{\omega} \iff s \in \alpha_{\omega} \text{ and } \beta_{\omega}(s) = 1 \iff s \in \zeta_{\omega}. \quad (4)$$

1076

1077

1078

1079

An exact planning model supports accurate planning as it precisely characterizes skills’ initiation and termination sets. However, this accuracy comes at the cost of practical feasibility, as any exact model achieves very little in terms of *abstraction*: it must express the full initiation and termination

1080 sets of each skill. Therefore, in many settings, alternative model properties that approximate the  
 1081 exactness of the learned model, namely *soundness*, *suitability*, and *correctness*, may be preferred as  
 1082 objectives for model learning. We now define these properties.

1083 A symbolic model is *sound* if it correctly predicts the effects of a plan: whenever a complete  
 1084 and sound planner predicts that a sequence of skills will reach some abstract state, executing the  
 1085 corresponding skills in the environment truly leads there. Soundness rules out spurious symbolic  
 1086 transitions that do not correspond to realizable outcomes. Formally, we define soundness as follows:  
 1087 Let  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$  denote a symbolic planning model, where  $\mathcal{P}$  is a finite set of predicates defining  
 1088 an *abstract state space*  $\bar{\mathcal{S}}$ , and  $\mathcal{A}$  is a set of *abstract actions* (skills) with preconditions and effects  
 1089 expressed in terms of  $\mathcal{P}$ . Each abstract state  $\bar{s} \in \bar{\mathcal{S}}$  is obtained by a learned *grounding function*  
 1090  $\Gamma : \mathcal{S} \rightarrow \bar{\mathcal{S}}$  that maps low-level agent states  $s \in \mathcal{S}$  to truth assignments over  $\mathcal{P}$ .

1091 **Definition 5** (Soundness). *The model  $\mathcal{M}$  is sound iff, for any valid plan  $\pi$  produced by a complete  
 1092 symbolic planner over  $\mathcal{M}$  and for all task instances  $\mathbf{p}_i \in \mathbf{P}$ ,*

$$\Gamma(\mathcal{T}(\pi, s_0)) = \bar{\mathcal{T}}_{\mathcal{M}}(\pi, \Gamma(s_0)),$$

1095 where  $s_0$  is the initial state,  $\mathcal{T}(\pi, s_0)$  is the set of states reachable by executing  $\pi$  from  $s_0$ , and  
 1096  $\bar{\mathcal{T}}_{\mathcal{M}}(\pi, \Gamma(s_0))$  is the abstract state predicted by  $\mathcal{M}$  after executing  $\pi$  from  $\Gamma(s_0)$ .

1097 A symbolic model is *complete* if it never omits real solutions: whenever the environment admits a  
 1098 way to solve a task, the planner can find a corresponding abstract plan in the model. Completeness  
 1099 rules out gaps in symbolic coverage that would make feasible problems appear unsolvable.

1100 **Definition 6** (Completeness). *The model  $\mathcal{M} = (\mathcal{P}, \mathcal{A})$  is complete if, for any task instance  $\mathbf{p}_i \in \mathbf{P}$   
 1101 and any sequence of low-level actions that achieves the goal from an initial state  $s_0$ , there exists a  
 1102 symbolic plan  $\pi$  over  $\mathcal{M}$  such that*

$$\Gamma(\mathcal{T}(\pi, s_0)) \models G_i,$$

1104 where  $G_i$  is the goal condition of  $\mathbf{p}_i$ .

1105 A symbolic model is *suitable* if it correctly characterizes when a skill can be applied, meaning the  
 1106 symbolic preconditions predicted by the model align with the skill's real initiation conditions. A  
 1107 skill is applicable in an abstract state iff it is applicable in the corresponding grounded state.

1108 **Definition 7** (Suitability). *The model  $\mathcal{M}$  is suitable if, for any valid plan  $\pi$  produced by a complete  
 1109 symbolic planner and for all task instances  $\mathbf{p}_i \in \mathbf{P}$ ,*

$$\Gamma(\mathcal{T}(\underline{a}_i, s_0)) \in \bar{I}_{\underline{a}_{i+1}} \iff \mathcal{T}(\omega_i, s_0) \in I_{\omega_{i+1}}, \quad \forall a \in \mathcal{A}, \forall \omega \in \Omega$$

1110 where  $s_0$  is an initial agent state,  $\mathcal{T}(\pi, s_0)$  is the state reached by executing  $\pi$  from  $s_0$ , and  $\bar{I}_a$  is the  
 1111 abstract initiation set of abstract skill  $a$ .

1112 The minimum requirement for a model  $\mathcal{M}$  to solve abstract planning problems is that it is *suitable*  
 1113 and *complete*, such that an abstract plan can always be found, and that it is always executable, if there  
 1114 exists a skill plan as a solution. Although methods for constructing such models exist and have been  
 1115 investigated in previous work, they do not provide these guarantees.

## 1116 D PROOFS

1117 **Lemma 2** (Predicate invention). *Let  $\mathcal{D}$  be the set of transition tuples  $\langle s, \omega, s' \rangle$  with  $s, s' \in \mathcal{S}$  and  
 1118  $\omega \in \Omega$ . Let  $B_n$  be a buffer of  $n$  i.i.d. samples from  $\mathcal{D}$ , each labeled with outcome and containing  
 1119 at least one successful and one failed transition. Let  $\mathcal{M}_n = (\mathcal{P}_n, \mathcal{A}_n)$  be the model learned by  
 1120 SKILLWRAPPER from  $B_n$ , where each operator in  $\mathcal{A}_n$  corresponds to some skill  $\omega \in \Omega$ . Then, for  
 1121 every  $\omega \in \Omega$  appearing in  $B_n$ ,*

$$I_{\omega} \triangle \alpha_{\omega} = \emptyset, \tag{5}$$

$$\beta_{\omega} \triangle \zeta_{\omega} = \emptyset, \tag{6}$$

1122 where  $\alpha_{\omega} = \bigcup_{a \in \mathcal{A}_{\omega}} \mathcal{G}(PRE_a)$ ,  $\zeta_{\omega} = \bigcup_{a \in \mathcal{A}_{\omega}} (\mathcal{G}(PRE_a) \setminus \mathcal{G}(EFF_a^-)) \cup \mathcal{G}(EFF_a^+)$ , and  $\triangle$  is the  
 1123 symmetric difference.

1134 *Proof sketch.* Suppose for contradiction that  $\exists \langle s, \omega, s' \rangle \in B_n$  with  $s \in I_\omega \Delta \alpha_\omega$ . Two cases arise:  
 1135  
 1136 **(i) False positive.**  $s \notin I_\omega$  but  $s \in \alpha_\omega$ . By construction of  $\alpha_\omega$ , there must also exist  $s_j \in I_\omega \cap \alpha_\omega$ .  
 1137 Thus  $1_{\alpha_\omega}(s) = 1_{\alpha_\omega}(s_j)$  while  $I_\omega(s) \neq I_\omega(s_j)$ , contradicting the update rule.  
 1138 **(ii) False negative.**  $s \in I_\omega$  but  $s \notin \alpha_\omega$ . Then some  $s_j \notin I_\omega \cap \alpha_\omega$  must also exist, yielding the same  
 1139 contradiction.  
 1140 Hence, no such  $s$  exists, and Eq. (6) holds. The proof for Eq. (7) is identical, replacing  $\alpha_\omega$  with  $\zeta_\omega$   
 1141 defined from operator effects.  $\square$   
 1142

1143 **Theorem 3 (Probabilistic-completeness of SKILLWRAPPER).** *Let  $\mathcal{M}^*$  be a Complete Model for  
 1144 a set of skills  $\Omega$ , where each  $\omega \in \Omega$  has initiation set  $I_\omega \subseteq \mathcal{S}$  and termination set  $\beta_\omega \subseteq \mathcal{S}$ . Let  
 1145  $\mu$  be a probability distribution over  $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ . Consider a finite hypothesis class  $\mathcal{H}$ , where  
 1146 each  $\mathcal{M} \in \mathcal{H}$  assigns a learned initiation set  $\widehat{I}_\omega = \bigcup_{a \in \mathcal{A}_\omega} \mathcal{G}(\text{PRE}_a)$  and termination set  $\widehat{\beta}_\omega =$   
 1147  $\bigcup_{a \in \mathcal{A}_\omega} (\mathcal{G}(\text{PRE}_a) \setminus \mathcal{G}(\text{EFF}_a^-)) \cup \mathcal{G}(\mathcal{G}_a^+)$  to each  $\omega \in \Omega$ .*

1148 *For any  $\mathcal{M} \in \mathcal{H}$ , define*

$$d_{\text{compl}}(\mathcal{M}, \mathcal{M}^*) = \Pr_{(s, \omega, s') \sim \mu} [(s \in \widehat{I}_\omega \Delta I_\omega) \vee (s' \in \widehat{\beta}_\omega \Delta \beta_\omega)].$$

1151 *Let  $n$  i.i.d. samples  $\{(s_i, \omega_i, s'_i)\}_{i=1}^n$  be drawn from  $\mu$ . Then, for every  $\epsilon > 0$ ,*

$$\Pr[d_{\text{compl}}(\widehat{\mathcal{M}}_n, \mathcal{M}^*) \leq \epsilon] \geq 1 - |\mathcal{H}| e^{-n\epsilon},$$

1155 *i.e., with high probability,  $\widehat{\mathcal{M}}_n$  misses fewer than an  $\epsilon$ -fraction of feasible transitions under  $\mu$ .*

1156 *Proof sketch.* For  $\mathcal{M} \in \mathcal{H}$ , define the *true error*

$$\text{Err}(\mathcal{M}) = d_{\text{compl}}(\mathcal{M}, \mathcal{M}^*),$$

1159 and the *empirical error*

$$\widehat{\text{Err}}(\mathcal{M}) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}[(s_i \in \widehat{I}_\omega \Delta I_\omega) \vee (s'_i \in \widehat{\beta}_\omega \Delta \beta_\omega)].$$

1163 By Lemma 2,  $\widehat{\text{Err}}(\widehat{\mathcal{M}}_n) = 0$ . Suppose some  $\mathcal{M} \in \mathcal{H}$  satisfies  $\text{Err}(\mathcal{M}) \geq \epsilon$ . Then each sample has  
 1164 probability at least  $\epsilon$  of revealing an error. The chance of seeing none in  $n$  i.i.d. draws is at most  
 1165  $e^{-n\epsilon}$  (by Hoeffding/Chernoff).  
 1166

1167 Applying a union bound over all  $\mathcal{M} \in \mathcal{H}$ ,

$$\Pr[\exists \mathcal{M} \in \mathcal{H} : \text{Err}(\mathcal{M}) \geq \epsilon \wedge \widehat{\text{Err}}(\mathcal{M}) = 0] \leq |\mathcal{H}| e^{-n\epsilon}.$$

1168 Since  $\widehat{\mathcal{M}}_n$  has  $\widehat{\text{Err}} = 0$ , the event  $\text{Err}(\widehat{\mathcal{M}}_n) \geq \epsilon$  is contained in this bound. Thus, with probability  
 1169 at least  $1 - |\mathcal{H}| e^{-n\epsilon}$ ,  $d_{\text{compl}}(\widehat{\mathcal{M}}_n, \mathcal{M}^*) \leq \epsilon$ .  $\square$

1170 **Theorem 4 (Soundness of SKILLWRAPPER).** *Let  $\mathcal{T}$  be the set of transition tuples  $\langle s, \omega, s' \rangle$  with  
 1171  $s \in \mathcal{S}$  and  $\omega \in \Omega$ . Let  $B_n$  be an experience buffer of  $n$  samples drawn from  $\mathcal{T}$ , each labeled with  
 1172 outcome. Suppose SKILLWRAPPER learns a model*

$$\mathcal{M}_n = (\mathcal{P}_n, \mathcal{A}_n),$$

1173 *where each  $a \in \mathcal{A}_n$  corresponds to some skill  $\omega \in \Omega$ . Then, for every operator  $a \in \mathcal{A}_n$  associated  
 1174 with  $\omega$ , there exists a real transition  $\langle s, \omega, s' \rangle \in B_n$  such that*

$$\underline{s} \models \text{PRE}_a \quad \text{and} \quad \underline{s}' \models \text{EFF}_a.$$

1181 *Proof sketch.* By construction, SKILLWRAPPER derives each operator  $a \in \mathcal{A}_n$  from transitions in  $B_n$   
 1182 through its wrapper procedure. If  $a$  is associated with skill  $\omega$ , then its preconditions  $\text{PRE}_a$  are obtained  
 1183 from the abstract representation of some observed  $s$ , and its effects  $\text{EFF}_a$  from the corresponding  $s'$ .  
 1184 Hence there must exist  $\langle s, \omega, s' \rangle \in B_n$  such that  $\underline{s} \models \text{PRE}_a$  and  $\underline{s}' \models \text{EFF}_a$ .  
 1185

1186 If no such transition existed, then  $a$  would be unsupported by data and would not have been generated.  
 1187 Thus every operator in  $\mathcal{A}_n$  corresponds to at least one valid observed transition, proving soundness.  
 1188  $\square$

1188 **Proof (by Contradiction).** Assume, for contradiction, that there is an operator  $a \in \mathcal{A}_n$  for which  
 1189 no transition  $\langle s, \omega, s' \rangle \in B_n$  supports it. That would mean:

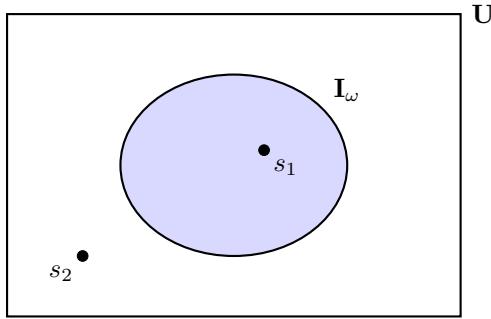
1190  
 1191 (1)  $a$  is in the learned model,  
 1192  
 1193  
 1194 (2)  $a$  has no real sample  $\langle s, \omega, s' \rangle \in B_n$  s.t.  $\underline{s} \models \text{PRE}_a \wedge \underline{s}' \models \text{EFF}_a$ .

1195 However, SKILLWRAPPER introduces or refines operators *only* in response to observed transitions  
 1196  $\langle s, \omega, s' \rangle$  that cannot be explained by any existing operator in  $\mathcal{A}_n$ . Therefore, if  $a$  exists in the final  
 1197 model, it must have been created when the system encountered a transition  $\langle s, \omega, s' \rangle$  with  $\underline{s}$  and  $\underline{s}'$   
 1198 not accounted for by any previously existing operator. That transition becomes the “anchor” for  $a$ ’s  
 1199 preconditions and effects.

1200 Hence, there *must* be at least one real transition  $\langle s, \omega, s' \rangle \in B_n$  matching the preconditions and  
 1201 effects of  $a$ , contradicting assumption (2). Consequently, our assumption is false, and each operator  
 1202 indeed has a supporting transition in  $B_n$ . This completes the proof.  $\square$

#### 1203 D.1 ILLUSTRATION OF PREDICATE INVENTION IN LOW-LEVEL SPACE

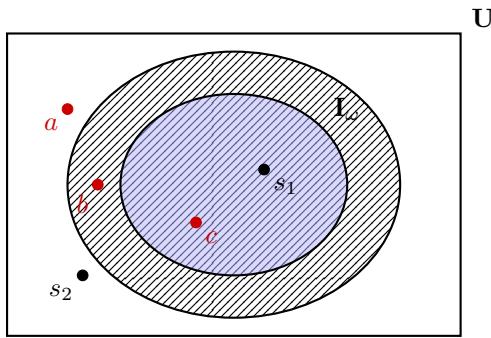
1204 Following the last section, we illustrate how SKILLWRAPPER invents new predicates under all cir-  
 1205 cumstances with guarantees using the condition. We discuss the case of precondition here, and  
 1206 guarantees of effect follow the same logic.



1207 By assumption, there exist two transitions:  
 1208  $\langle s_1, \omega, s'_1 \rangle, \langle s_2, \omega, s'_2 \rangle$  such that  $s_1 \in I_\omega, s_2 \notin I_\omega$   
 1209 initially.

1210 There are three possible circumstances of the resulting learning model from  $\langle s_1, \omega, s'_1 \rangle$  and  
 1211  $\langle s_2, \omega, s'_2 \rangle$ . For each of them, we discuss all possible cases of more transitions with the starting state  
 1212 falling into each section of the state space.

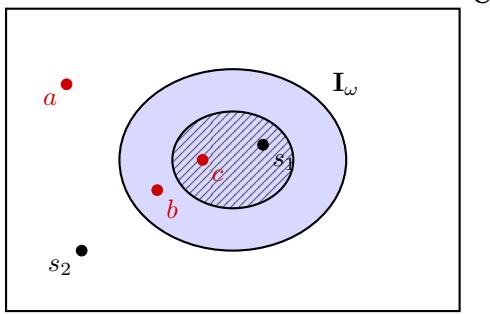
1213 (1). The learned model has  $I_\omega \subset \alpha$ . Then, for each section that initial states of new transitions can  
 1214 fall in:



- (a):  $a \notin \alpha, a \notin I_\omega$ .  
 Thus,  $\nexists \langle s, \omega, s' \rangle$  such that  $\mathbf{1}_\alpha(a) = \mathbf{1}_\alpha(s)$  while  $I_\omega(a) \neq I_\omega(s)$ . No additional predicate need to be invented.
- (b):  $b \in \alpha, a \notin I_\omega$ .  
 Thus,  $\mathbf{1}_\alpha(b) = \mathbf{1}_\alpha(s_1)$  while  $I_\omega(b) \neq I_\omega(s_1)$ . New predicate will be invented.
- (c):  $a \in \alpha, a \in I_\omega$ .  
 Thus,  $\nexists \langle s, \omega, s' \rangle$  such that  $\mathbf{1}_\alpha(a) = \mathbf{1}_\alpha(s)$  while  $I_\omega(a) \neq I_\omega(s)$ . No additional predicate need to be invented.

1215 (2). The learned model has  $\alpha \subset I_\omega$ . Then, for each section that initial states of new transitions can  
 1216 fall in:

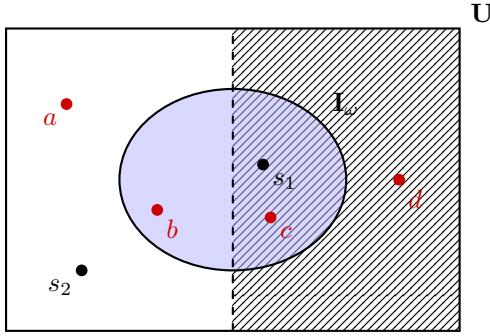
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252



1253  
1254  
1255

(3). The learned model has  $\alpha \cap I_\omega \neq \emptyset, \alpha \not\subseteq I_\omega, I_\omega \not\subseteq \alpha$ . Then, for each section that initial states of new transitions can fall in:

1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269



- (a):  $a \notin \alpha, a \notin I_\omega$ .  
Thus,  $\nexists \langle s, \omega, s' \rangle$  such that  $\mathbf{1}_\alpha(a) = \mathbf{1}_\alpha(s)$  while  $I_\omega(a) \neq I_\omega(s)$ . No additional predicate need to be invented.
- (b):  $b \notin \alpha, a \in I_\omega$ .  
Thus,  $\mathbf{1}_\alpha(b) = \mathbf{1}_\alpha(s_1)$  while  $I_\omega(a) \neq I_\omega(s)$ . New predicate will be invented.
- (c):  $a \in \alpha, a \in I_\omega$ .  
Thus,  $\nexists \langle s, \omega, s' \rangle$  such that  $\mathbf{1}_\alpha(a) = \mathbf{1}_\alpha(s)$  while  $I_\omega(a) \neq I_\omega(s)$ . No additional predicate need to be invented.

1270  
1271  
1272

So far, we have discussed all possible cases where initial states of new transitions can fall in, and the predicate invention condition is proven to handle all cases.

1273

## E ADDITIONAL DETAILS ON THE HYPOTHESIS CLASS AND SAMPLE COMPLEXITY

1274  
1275  
1276

Recall that our learned symbolic model has the form

1277  
1278  
1279  
1280

$$\mathcal{M} = (\mathcal{P}, \mathcal{A}),$$

where  $\mathcal{P}$  is a set of predicates and  $\mathcal{A} = \bigcup_{\omega \in \Omega} \mathcal{A}_\omega$  is a set of abstract operators, with  $\mathcal{A}_\omega$  the set of operators associated with skill  $\omega \in \Omega$ .

1281

Each operator  $a \in \mathcal{A}_\omega$  is defined by its preconditions and add/delete effects:

1282  
1283

$$a \equiv (\omega, \Theta_a, \text{PRE}_a, \text{EFF}_a^+, \text{EFF}_a^-), \quad \text{PRE}_a, \text{EFF}_a^+, \text{EFF}_a^- \subseteq \mathcal{P}.$$

1284  
1285  
1286  
1287

In our implementation, the number of VLM calls is finite, and the resulting models use a small number of predicates in environments with finitely many objects. For the *theoretical analysis*, we make this implicit resource bound explicit:

1288  
1289  
1290  
1291

- Let  $P_{\max}$  denote a fixed maximum number of predicates that SKILLWRAPPER is allowed to invent.
- Let  $\mu_{\max}$  denote the maximum arity of any predicate  $\sigma \in \mathcal{P}$ .

1292  
1293  
1294  
1295

Because SKILLWRAPPER learns one operator per lifted effect cluster, we can derive an upper bound on the number of operators per skill  $\omega \in \Omega$  based on the number of possible effect sets. For any operator  $a \in \mathcal{A}_\omega$  and predicate  $\sigma \in \mathcal{P}$ , there are three possible cases:  $\sigma \in \text{EFF}_a^+$ ,  $\sigma \in \text{EFF}_a^-$ , or  $\sigma \notin \text{EFF}_a^- \cup \text{EFF}_a^+$ . Because the upper bound of possible instances of  $p$  is  $|\mathcal{O}|^{\mu_{\max}}$ , we can express the maximum number of operators as

1296  
1297  
1298

$$A_{\max} = 3^{P_{\max} \cdot |\mathcal{O}|^{\mu_{\max}}}$$

1299 We therefore define the hypothesis class analyzed in Theorem 2 as

1300  
1301  
1302

$$\mathcal{H} = \left\{ (\mathcal{P}, \mathcal{A}) \mid |\mathcal{P}| \leq P_{\max}, |\mathcal{A}_\omega| \leq A_{\max} \forall \omega \in \Omega \right\}. \quad (7)$$

1303 Predicate re-evaluation and removal during learning do *not* expand  $\mathcal{H}$ ; they only move the learned  
1304 model within this resource-bounded class by altering which predicates and operators are actively  
1305 used.1306  
1307E.1 UPPER BOUND ON  $|\mathcal{H}|$ 1308  
1309We now derive a practical upper bound on the size of  $\mathcal{H}$  in equation 7.1310  
1311Fix a predicate set  $\mathcal{P}$  with  $|\mathcal{P}| \leq P_{\max}$ . For each operator  $\alpha$ , its symbolic definition is given by three  
1312 subsets of  $\mathcal{P}$ :

$$\text{PRE}_a, \text{EFF}_a^+, \text{EFF}_a^- \subseteq \mathcal{P}.$$

1313  
1314  
1315We consider negative precondition, so there are three possibilities for one predicate  $p$ :  $p \in \text{PRE}_a$ ,  
1316  $-p \in \text{PRE}_a$ , and  $p \notin \text{PRE}_a$ . Since  $\text{EFF}_a^+$  and  $\text{EFF}_a^-$  are considered in  $A_{\max}$ , a single operator has at  
most

$$3^{P_{\max} \cdot |\mathcal{O}|^{\mu_{\max}}} \quad (8)$$

1317  
1318  
1319  
1320  
1321distinct configurations of preconditions.  
For a fixed skill  $\omega \in \Omega$ , we allow at most  $A_{\max}$  operators. Treating each of the  $A_{\max}$  operator “slots”  
1322 as independently choosing one of the  $3^{P_{\max} \cdot |\mathcal{O}|^{\mu_{\max}}}$  possible configurations in equation 8, the total  
1323 number of operator-sets  $\mathcal{A}_\omega$  for that skill is bounded by

$$(3^{P_{\max} \cdot |\mathcal{O}|^{\mu_{\max}}})^{A_{\max}} = 3^{P_{\max} A_{\max} |\mathcal{O}|^{\mu_{\max}}}. \quad (9)$$

1324  
1325  
1326  
1327Across all skills  $\omega \in \Omega$ , we obtain the bound

$$|\mathcal{H}| \leq \prod_{\omega \in \Omega} 3^{P_{\max} A_{\max} |\mathcal{O}|^{\mu_{\max}}} = 3^{P_{\max} A_{\max} |\Omega| |\mathcal{O}|^{\mu_{\max}}}. \quad (10)$$

1328  
1329  
1330  
1331E.2 SAMPLE COMPLEXITY FOR A TARGET  $(\epsilon, \delta)$ Theorem 2 states that, for any  $\epsilon > 0$ ,1332  
1333

$$\Pr[d_{\text{compl}}(\widehat{\mathcal{M}}_n, \mathcal{M}^*) > \epsilon] \leq |\mathcal{H}| e^{-n\epsilon}, \quad (11)$$

1334  
1335  
1336where  $\widehat{\mathcal{M}}_n$  is the model returned by SKILLWRAPPER after observing  $n$  i.i.d. transitions, and  $d_{\text{compl}}$   
1337 is the completeness distance defined in the main text (probability of a “missed feasible” event under  
1338 the transition distribution).1339  
1340Substituting the bound on  $|\mathcal{H}|$  from equation 10 into equation 11 yields1341  
1342  
1343

$$\Pr[d_{\text{compl}}(\widehat{\mathcal{M}}_n, \mathcal{M}^*) > \epsilon] \leq 3^{P_{\max} A_{\max} |\Omega| |\mathcal{O}|^{\mu_{\max}}} e^{-n\epsilon}. \quad (12)$$

1344  
1345  
1346  
1347To guarantee that this probability is at most  $\delta \in (0, 1)$ , it suffices that

$$3^{P_{\max} A_{\max} |\Omega| |\mathcal{O}|^{\mu_{\max}}} e^{-n\epsilon} \leq \delta,$$

1348  
1349

which is equivalent to

$$n \geq \frac{1}{\epsilon} \left( P_{\max} A_{\max} |\Omega| |\mathcal{O}|^{\mu_{\max}} \ln 3 + \ln \frac{1}{\delta} \right). \quad (13)$$

Thus, the number of transitions required to ensure

$$d_{\text{compl}}(\widehat{\mathcal{M}}_n, \mathcal{M}^*) \leq \epsilon \quad \text{with probability at least } 1 - \delta$$

1350  
1351  
1352  
1353

is

$$n = \mathcal{O}\left(\frac{P_{\max} A_{\max} |\Omega| |\mathcal{O}|^{\mu_{\max}} + \log(1/\delta)}{\epsilon}\right). \quad (14)$$

1354  
1355  
1356  
1357

In our experiments, the arity and realized numbers of predicates are small. Setting  $P_{\max}$  to match the practical budget yields numerical values in equation 13 for the regimes we study. Our theoretical result therefore formalizes how increasing the capacity of the symbolic model (via larger  $P_{\max}$ ) trades off against the number of transitions needed to achieve a desired completeness level.

1358  
1359  
1360

## F LEARNED OPERATORS, CASE STUDIES, AND EXAMPLE TASKS

1361  
1362

### F.1 LEARNED OPERATORS

1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385

```
# Example Learned Predicate and Operator of Burger domain

# Predicate
name: cut_into_pieces
types:
- cuttable
semantic: "the cuttable appears as at least two non-touching visible pieces (has multiple disconnected regions), indicating it is cut rather than a single intact piece."
```

1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403

```
# Example Learned Predicate and Operator of Franka domain

# Predicate
name: plate_is_dirty
types:
- plate
semantic: the specified plate's upper surface contains visible food traces whose appearance differs from the plate's base surfac (visible residue to be wiped).

# Operator
(:action Stack_5
:parameters (?pickupable_p0 - pickupable
            ?pickupable_p1 - pickupable
            ?pickupable_p3 - pickupable
            ?plate_p4 - plate
            ?robot_p2 - robot)
:precondition (and
              (not (= ?pickupable_p0 ?pickupable_p1))
              (not (= ?pickupable_p0 ?pickupable_p3))
              (not (= ?pickupable_p1 ?pickupable_p3))
              (holding ?robot_p2 ?pickupable_p1)
              (plate_top_unoccupied ?plate_p4)
              (not (gripper_empty ?robot_p2))
              (not (holding ?robot_p2 ?pickupable_p0))
              (not (holding ?robot_p2 ?pickupable_p3))
              (not (plate_is_dirty ?plate_p4))
              (not (stacked_on ?pickupable_p1 ?plate_p4)))
:effect (and
          (cut_into_pieces ?cuttable_p0)))
```

```

1404      (gripper_empty ?robot_p2)
1405      (stacked_on ?pickupable_p1 ?plate_p4)
1406      (not (holding ?robot_p2 ?pickupable_p1))
1407      (not (plate_top_unoccupied ?plate_p4)))
1408
1409  # Example Learned Predicate and Operator of Bi-manual Kuka domain
1410
1411  # Predicate
1412  name: Coated
1413  types:
1414  - utensil
1415  semantic: a visible layer or clump of material adheres to the utensil's working end (e.g.,
1416  the blade shows a smear that was absent before).
1417
1418  # Operator
1419  (:action Scoop_156
1420  :parameters (?openable_p0 - openable
1421  ?robot_p1 - robot
1422  ?utensil_p2 - utensil)
1423  :precondition (and
1424  (HeldByRobot ?robot_p1 ?openable_p0)
1425  (InLeftGripper ?robot_p1 ?openable_p0)
1426  (InRightGripper ?robot_p1 ?utensil_p2)
1427  (LidOff ?openable_p0)
1428  (not (Closed ?openable_p0))
1429  (not (Coated ?utensil_p2))
1430  (not (InContainer ?utensil_p2))
1431  (not (LeftGripperEmpty ?robot_p1))
1432  (not (OpenableOnTable ?openable_p0))
1433  (not (RightGripperEmpty ?robot_p1))
1434  (not (UtensilOnTable ?utensil_p2)))
1435  :effect (and
1436  (Coated ?utensil_p2)))
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447

```

Additionally, we present here a predicate and an operator written by the PDDL expert:

```

1430  # Predicate and Operator Written by PDDL Expert
1431
1432  # Predicate
1433  name: is_on_station
1434  types:
1435  - pickupable
1436  - station
1437  semantic: "A `pickupable` object is on top of a `station`."
1438
1439  # Operator
1440  (:action Cut
1441  :parameters (?robot - robot
1442  ?cuttable - cuttable
1443  ?board - cuttingboard)
1444  :precondition (and
1445  (hand_empty)
1446  (obj_free ?cuttable)
1447  (is_on_station ?cuttable ?board)
1448  (not (is_cut ?cuttable)))
1449  :effect (and
1450  (is_cut ?cuttable)))
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
```

1458 that are underneath a stack. As a result, the lack of soundness induces a low solved rate for hard and  
 1459 impossible problems of System Predicates.  
 1460

1461 **Random exploration** is heavily limited by inefficient data gathering. However, if we take a closer  
 1462 look at the operators it learns, they are of good quality, which is the benefit of sharing the same  
 1463 predicate invention algorithm of SKILLWRAPPER. In fact, one frequent failure mode under this  
 1464 randomness is that the baseline may never execute the skills with complex preconditions successfully.  
 1465 For example, in the Robotouille setting, the *Cook* action requires the item to be on the stove and the  
 1466 agent’s hand to be empty. Since the same predicate invention algorithm with SKILLWRAPPER is being  
 1467 used, it can only learn operators for the skills that have been successfully executed in the observed  
 1468 transitions. As a result, it only learns operators for skills that are usually executable, such as *Pick*,  
 1469 and it thus can only solve the simplest pick and stack tasks. Additionally, random exploration also  
 1470 achieves 100% on *Impossible*, yet because the learned model is not plannable.

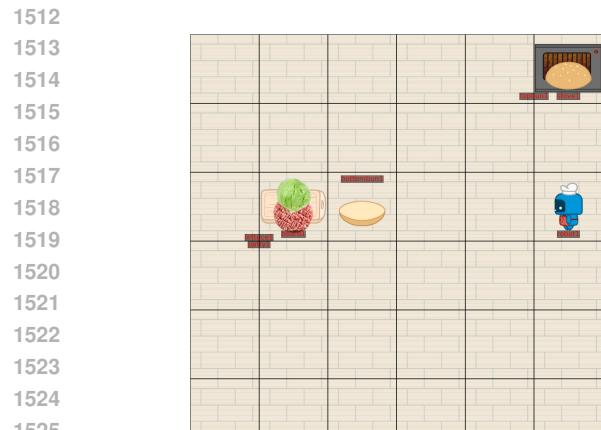
1471 **No Heuristic** shares a similar failure mode as System Predicates—achieves good performance  
 1472 in *Easy* problems but degrades significantly in *Hard*. With further investigation, we found that  
 1473 the baseline generally invents fewer predicates than SKILLWRAPPER, which results in occasionally  
 1474 missing critical ones (two out of the five total runs). This observation explains the large variances in  
 1475 *Hard* and *Impossible* problems, and also indicates that *Easy* problems could be solved even with an  
 1476 incomplete predicate set, which aligns with our findings in System Predicate. In turn, it supports the  
 1477 usefulness of the two engineered heuristics for skill sequence proposal. We believe improving the  
 1478 exploration strategy is a promising direction for future work. Another point we want to note is that  
 1479 either the solved rate or the planning budget only evaluates the learned operators from the planning  
 1480 outcome, while better metrics are needed for evaluating the exploration efficiency.

1481 **SKILLWRAPPER**’s failure mode is similar to the case of Random exploration. Comparing the poorly  
 1482 performing PDDL operators learned by SKILLWRAPPER to Expert Operators, the model only contains  
 1483 one extra predicate, *on\_cutting\_board(item)*, which divides the previous cluster that shares the  
 1484 same effect into even smaller clusters. Then, the transition data in these smaller clusters cannot  
 1485 support the model learning algorithm to effectively eliminate spurious preconditions. This case  
 1486 study points out that the balance between predicate invention and data gathering is a critical factor  
 1487 in the learning process: if too many predicates are invented without adequate transition data, the  
 1488 resulting operators could possibly contain spurious preconditions, such that it cannot generalize at  
 1489 all. In SKILLWRAPPER, the balance is controlled by empirically tuning the threshold of the scoring  
 1490 function and the length of each skill sequence proposed. Another sub-optimality is that our algorithm  
 1491 does not filter invented predicates that are semantic synonyms or antonyms to existing predicates,  
 1492 which increases the classification burden of the foundation model. Though these redundant predicates  
 1493 are usually well handled by the foundation model, a smarter prompting system could be designed to  
 1494 mitigate this issue and improve computation efficiency.

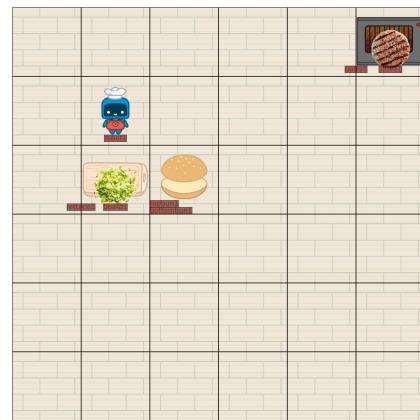
### 1494 F.3 EXAMPLE TASKS

1495 Here, we provide additional examples of the tasks, using planning problem from the evaluation data.  
 1496 Specifically, Robotouille tasks use images generated by the simulator, Franka tasks use images taken  
 1497 by a fixed camera in front of the robot, and Bimanual Kuka tasks use images taken by its egocentric  
 1498 camera.  
 1499

1500  
 1501  
 1502  
 1503  
 1504  
 1505  
 1506  
 1507  
 1508  
 1509  
 1510  
 1511

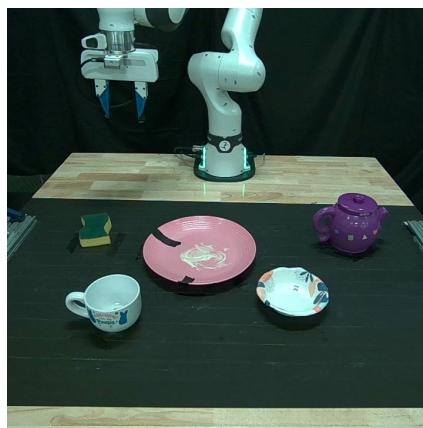


(a) Initial state



(b) Goal state

Figure 7: Example task in Robotouille.

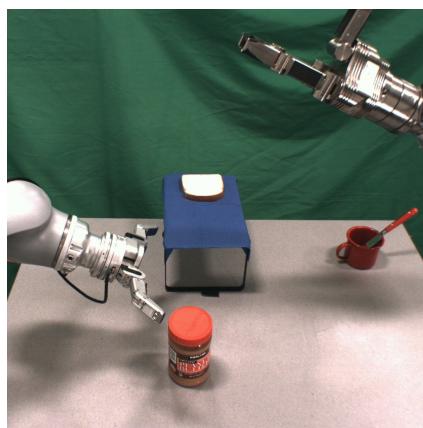


(a) Initial state

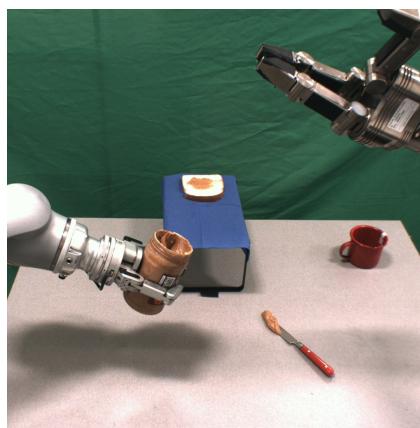


(b) Goal state

Figure 8: Example task in Franka.



(a) Initial state



(b) Goal state

Figure 9: Example task in Bimannual Kuka.

1566 **G VLM RELIABILITY STUDY**  
15671568 **G.1 CLASSIFICATION ACCURACY**  
15691570 We here analyze the classification accuracy of the vision-language model (VLM) used in both robotic  
1571 experiments. Since the predicates are generated on the fly by the VLM, we do not have the ground  
1572 truth values for them, and thus we must verify if the truth values match the images manually.  
15731574 **Per-predicate Classification Evaluation.** Since predicates are originally lifted and can be  
1575 grounded with different combinations of objects, we first define a classification over a low-level  
1576 state of a grounded predicate as correct if (1) all parameters appear in the scene (if the predicate is  
1577 not nullary) and (2) the truth value of the predicate match the low-level state specified by the image  
1578 input. Then, we define a classification of a lifted predicate over a low-level state as correct if all of  
1579 its grounded instances are classified correctly over that state.  
15801581 **Results and Analysis.** Over all predicates, the classification accuracy is 86.7% for the Franka  
1582 experiment, and 98.5% for the bimanual Kuka experiment. Compared to the planning performance  
1583 reported for both experiments, the classification accuracy is generally much higher. One reason for  
1584 this mismatch is that, due to the rigidity of symbolic planning, even flipping the truth value of a single  
1585 predicate can lead to a planning failure. To support this claim, we found specific poorly performing  
1586 predicates that hinder the planning task the most, and we provide more quantitative results in the  
1587 next paragraph.  
15881589 **Per-predicate Accuracy.** The learned symbolic model of the Franka experiment contains 6 predicates,  
1590 which have 11 possible grounded instances. The learned symbolic model of the bimanual Kuka experiment  
1591 contains 12 predicates, which have 13 possible grounded instances. We evaluate per-predicate accuracy for both in Table 3 and Table 4. From the results of the Franka experiment,  
1592 we identify the two predicates, `gripper_empty` and `holding`, that caused all planning failures, and  
1593 they fail almost simultaneously due to their semantic correlation. With further investigation, we  
1594 found that the misclassifications were induced by a single object, `Sponge`, which is possibly due to  
1595 the color of the object and the background being too similar. In the bimanual Kuka experiment, it is  
1596 `coated` (if the knife has peanut butter on it) that caused most of the planning failure, likely caused  
1597 by the lighting conditions. These observations suggest the accuracy of VLM is a limiting factor, and  
1598 resolving them poses a promising path to improving the performance of SKILLWRAPPER.  
1599

1600 Table 3: Per-predicate accuracy of Franka.

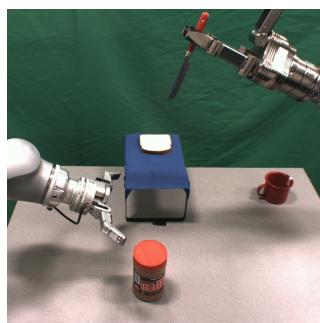
1601

	gripper_empty	holding	mug_full	plate_top_unoccupied	stacked_on	plate_is_dirty
Accuracy (%)	60.0	60.0	100.0	100.0	100.0	100.0

1602 Table 4: Per-predicate accuracy of Bimanual Kuka.

1603

	InLeftGripper	InRightGripper	RightGripperEmpty	LeftGripperEmpty	LidOff	InContainer
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	98.3
	OpenableOnTable	Closed	Coated	SpreadOn	HeldByRobot	UtensilOnTable
Accuracy (%)	96.7	100.0	88.3	98.3	100.0	100.0

1605 (a)  $\text{holding}(\text{Sponge}) = \mathbf{F}$ 1607 (b)  $\text{coated}(\text{Knife}) = \mathbf{T}$

1620  
1621

## G.2 REAL-WORLD ROBUSTNESS

1622  
1623  
1624  
1625

To evaluate the real-world robustness of the VLM, we additionally conduct experiments to investigate factors such as viewpoints, lighting conditions, or domain shifts. For each of them, we collect a held-out set of images by varying these factors. We report per-predicate accuracy, and all numbers are averaged across three individual runs.

1626

1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635

**Viewpoints.** We collect visual observation data from two viewpoints and sample five configurations for each viewpoint. From the results of Franka experiments, we observe that the classification accuracies of certain predicates are higher from the viewpoint closer to the corresponding objects: at viewpoint #1, all predicates can be perfectly classified, while predicates involving gripper or mug, such as `gripper_empty`, `holding` and `mug_full`, are significantly lower from viewpoint #2, which is farther from the objects. In bimanual Kuka experiments, the result is mostly stable across different viewpoints and generally better than in the Franka environment, which is possibly due to fewer background distractions. Though the accuracy varies across viewpoints, its performance remains reliable as long as the full observability assumption still holds.

1636  
1637

Table 5: Per-predicate accuracy of Franka at Viewpoint #1.

	<code>gripper_empty</code>	<code>holding</code>	<code>mug_full</code>	<code>plate_top_unoccupied</code>	<code>stacked_on</code>	<code>plate_is_dirty</code>
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	100.0

1641

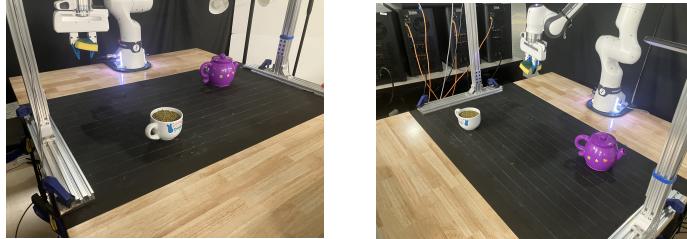
1642  
1643

Table 6: Per-predicate accuracy of Franka at Viewpoint #2.

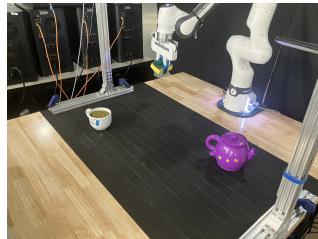
	<code>gripper_empty</code>	<code>holding</code>	<code>mug_full</code>	<code>plate_top_unoccupied</code>	<code>stacked_on</code>	<code>plate_is_dirty</code>
Accuracy (%)	80.0	80.0	90.0	100.0	100.0	100.0

1647

1648

(a) ✓ `holding(Sponge) = T`(b) ✗ `holding(Sponge) = F`

1649

1650  
1651  
1652  
1653  
1654  
1655  
1656

1657

1658

1659

1660

1661

Table 7: Per-predicate accuracy of Bimanual Kuka at Viewpoint #1.

	<code>InLeftGripper</code>	<code>InRightGripper</code>	<code>RightGripperEmpty</code>	<code>LeftGripperEmpty</code>	<code>LidOff</code>	<code>InContainer</code>
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	100.0
	<code>OpenableOnTable</code>	<code>Closed</code>	<code>Coated</code>	<code>SpreadOn</code>	<code>HeldByRobot</code>	<code>UtensilOnTable</code>
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	100.0

1667

1668

1669

Table 8: Per-predicate accuracy of Bimanual Kuka at Viewpoint #1.

	<code>InLeftGripper</code>	<code>InRightGripper</code>	<code>RightGripperEmpty</code>	<code>LeftGripperEmpty</code>	<code>LidOff</code>	<code>InContainer</code>
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	100.0
	<code>OpenableOnTable</code>	<code>Closed</code>	<code>Coated</code>	<code>SpreadOn</code>	<code>HeldByRobot</code>	<code>UtensilOnTable</code>
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	93.3

1674

1675

1676

1677

1678

1679

1680

1681

1682

1683

1684

1685

1686

1687

1688

1689

**Lighting conditions.** We collect visual observation data under two lighting conditions and sample five configurations for each one. We find that the VLM is generally robust to different lighting conditions, except for several extremely hard ones, such as under lighting condition #2 in Bimanual Kuka, where the objects are heavily shadowed.

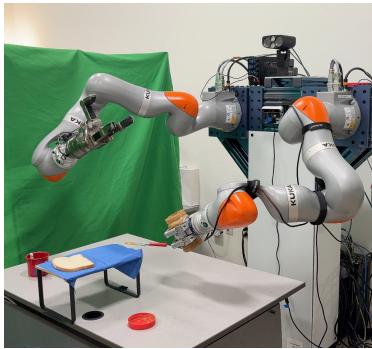
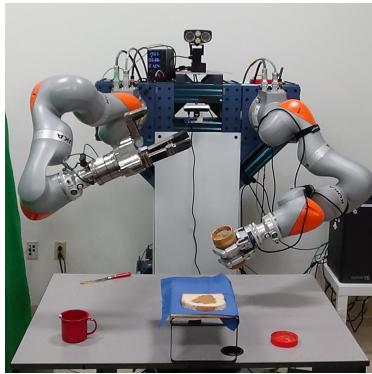
1693

1694

1695

(a) ✓ UtensilOnTable(Knife) = T

(b) ✗ UtensilOnTable(Knife) = F



1696

1697

1698

Table 9: Per-predicate accuracy of Franka under Lighting Condition #1.

	gripper_empty	holding	mug_full	plate_top_unoccupied	stacked_on	plate_is_dirty
Accuracy (%)	90.0	90.0	100.0	96.7	100.0	100.0

1699

1700

1701

1702

Table 10: Per-predicate accuracy of Franka under Lighting Condition #2.

	gripper_empty	holding	mug_full	plate_top_unoccupied	stacked_on	plate_is_dirty
Accuracy (%)	90.0	90.0	100.0	98.3	100.0	100.0

1703

1704

1705



1706

1707

1708

1709

1710

1711

1712

1713

1714

1715

1716

1717

1718

(a) ✓ stacked\_on(Teapot, Plate) = T

(b) ✓ stacked\_on(Teapot, Plate) = T

1719

1720

1721

1722

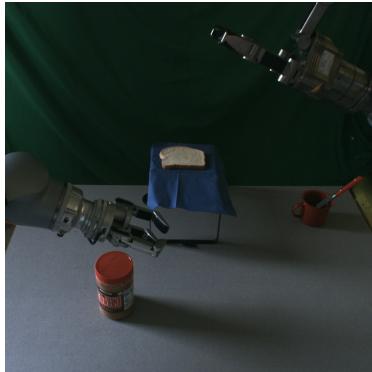
1723

Table 11: Per-predicate accuracy of Bimanual Kuka under Lighting Condition #1.

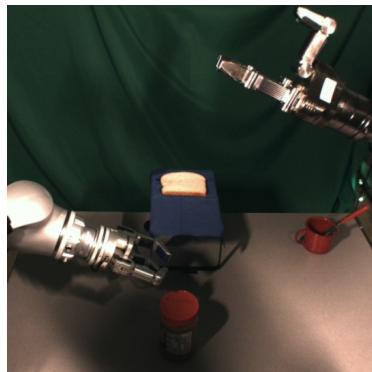
	InLeftGripper	InRightGripper	RightGripperEmpty	LeftGripperEmpty	LidOff	InContainer
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	100.0
	OpenableOnTable	Closed	Coated	SpreadOn	HeldByRobot	UtensilOnTable
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	100.0

Table 12: Per-predicate accuracy of Bimanual Kuka under Lighting Condition #2.

	InLeftGripper	InRightGripper	RightGripperEmpty	LeftGripperEmpty	LidOff	InContainer
Accuracy (%)	100.0	80.0	80.0	100.0	100.0	73.3
	OpenableOnTable	Closed	Coated	SpreadOn	HeldByRobot	UtensilOnTable
Accuracy (%)	100.0	100.0	100.0	86.7	100.0	100.0



(a) ✓ RightGripperEmpty() = T



(b) ✗ RightGripperEmpty() = F

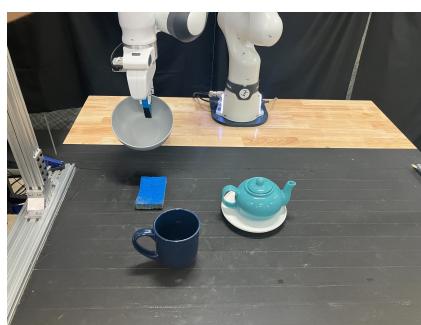
**Domain shift.** SKILLWRAPPER relies entirely on semantics to prompt the VLM, abstracting raw states into symbolic states without using visual features. The only requirement for generalizing to novel objects is that the VLM can correctly identify the object referents in the image based on their type information provided in the language prompt. To evaluate this generalization capability, we collected visual observation data (five images per environment) under domain shift by swapping objects with new instances and sampling two configurations. From this observation, we found that the only failure mode introduced by domain shifts occurs when the VLM cannot recognize an object because its visual appearance does not align with the semantics. For example, a plate that is too small might be misclassified as a saucer, leading to incorrect symbolic states.

Table 13: Per-predicate accuracy of Franka under Domain Shift.

	gripper_empty	holding	mug_full	plate_top_unoccupied	stacked_on	plate_is_dirty
Accuracy (%)	90.0	90.0	100.0	100.0	73.3	100.0



(a) ✓ stacked\_on(Teapot, Plate) = T



(b) ✗ stacked\_on(Teapot, Plate) = F

Table 14: Per-predicate accuracy of Bimanual Kuka under Domain Shift.

1782  
1783

	InLeftGripper	InRightGripper	RightGripperEmpty	LeftGripperEmpty	LidOff	InContainer
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	100.0
	OpenableOnTable	Closed	Coated	SpreadOn	HeldByRobot	UtensilOnTable
Accuracy (%)	100.0	100.0	100.0	100.0	100.0	100.0

1788

1789

1790

1791

1792

1793

1794

1795

1796

1797

1798

1799

1800

1801

1802

1803

1804

1805

1806

1807

## G.3 OTHER VLMs

We further examine the possibility of using open-source VLMs as alternatives for SKILLWRAPPER. We choose Qwen3-VL-235B (Bai et al., 2025) for comparison. To evaluate its capability, we conduct two sets of preliminary experiments: predicate classification and predicate invention.

1811

1812

1813

1814

1815

1816

1817

**Predicate Classification.** We collected a subset of images (five from Franka and ten from Bimanual Kuka) and evaluated the truth values of each predicate with the two models. From the result, we observed that the two models have different failure patterns, and a prominent one is that Qwen3 can reliably detect if the gripper is holding an object, except for occasional classification errors on the objects being held. In general, we found two models perform on par with each other, and thus we believe they can be used interchangeably for predicate classification.

1818

1819

Table 15: Per-predicate accuracy of Franka.

	gripper_empty	holding	mug_full	plate_top_unoccupied	stacked_on	plate_is_dirty
GPT-5 Acc. (%)	60.0	60.0	100.0	100.0	100.0	100.0
Qwen3 Acc. (%)	100.0	80.0	100.0	100.0	100.0	80.0

1823

1824

1825

1826

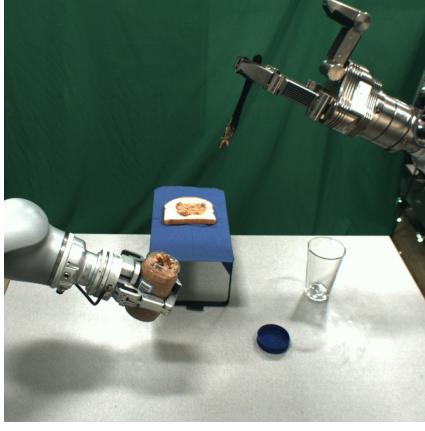
Table 16: Per-predicate accuracy of Bimanual Kuka.

	InLeftGripper	InRightGripper	RightGripperEmpty	LeftGripperEmpty	LidOff	InContainer
GPT-5 Acc. (%)	100.0	100.0	100.0	100.0	100.0	100.0
Qwen3 Acc. (%)	100.0	100.0	100.0	100.0	100.0	100.0
	OpenableOnTable	Closed	Coated	SpreadOn	HeldByRobot	UtensilOnTable
GPT-5 Acc. (%)	100.0	100.0	100.0	100.0	100.0	100.0
Qwen3 Acc. (%)	100.0	100.0	90.0	80.0	100.0	100.0

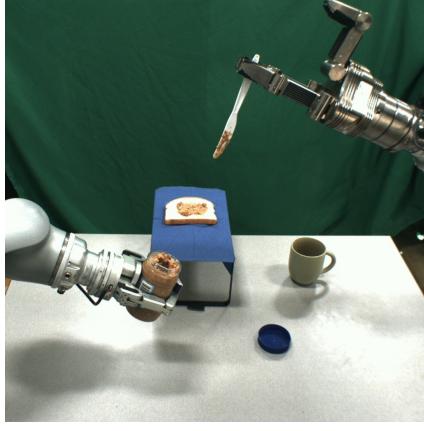
1832

1833

**Predicate Invention.** We qualitatively compare the performance of both models on inventing predicates by reasoning over contrastive pairs of transitions. For each environment, we curated two contrastive pairs, and each model is prompted by the same input to invent one new predicate. A

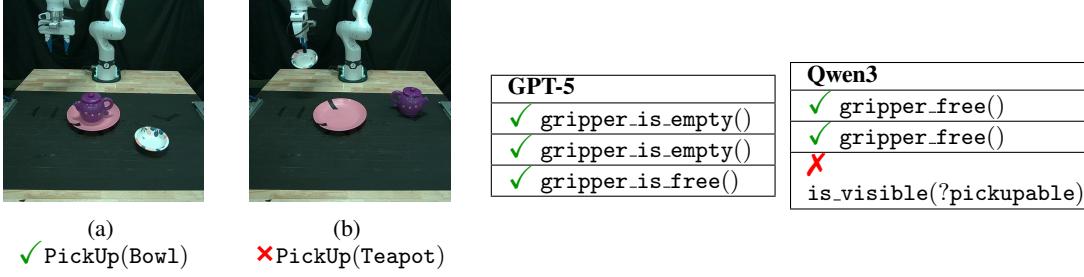


(a) ✓ coated(Knife) = T

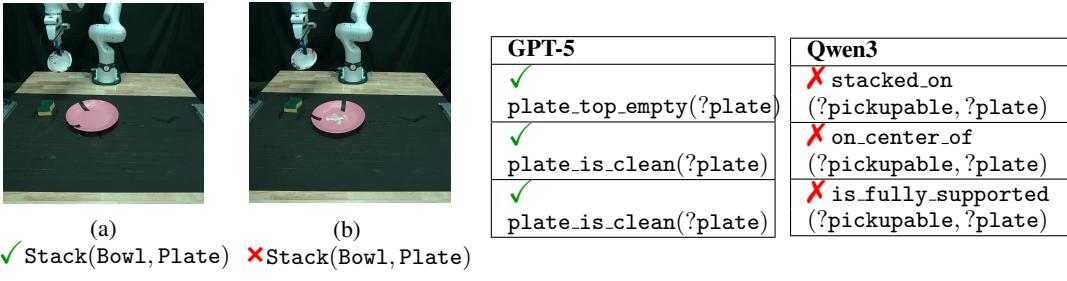


(b) ✓ coated(Knife) = T

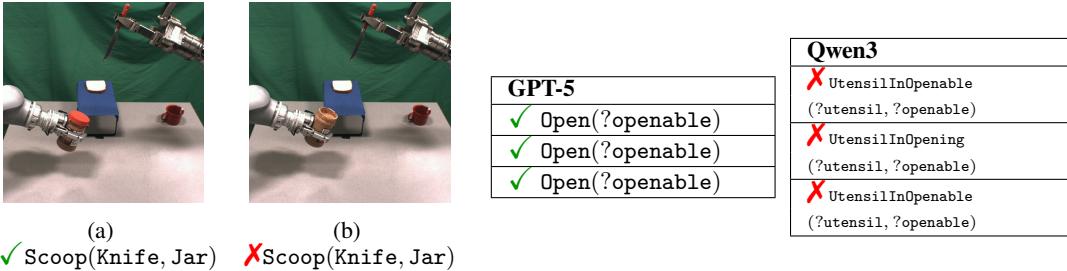
1836 predicate is considered correct if it is a semantic synonym or antonym of the target predicate. From  
 1837 the result, we can conclude that GPT-5 is much more reliable in reasoning over the transitions for  
 1838 predicate invention, and thus Qwen3 cannot be used as an alternative for this specific task. (We  
 1839 omitted ? robot from all predicates' arguments for simplicity.)



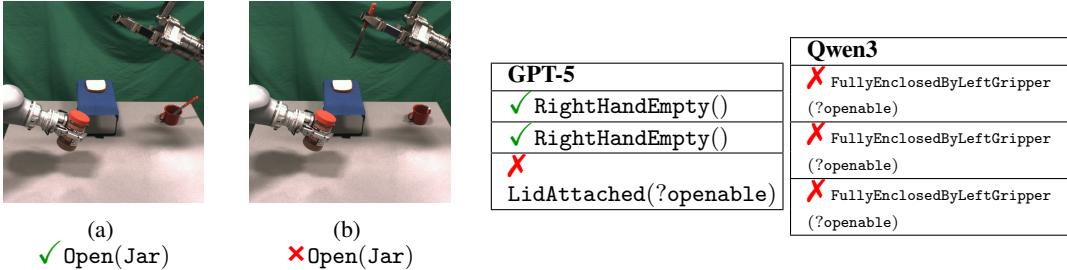
1850 **Figure 17: Predicate Invention Case #1 in Franka.** Target predicate: GripperEmpty()  
 1851 Existing predicates:  $\emptyset$



1863 **Figure 18: Predicate Invention Case #2 in Franka.** Target predicate: PlateIsDirty(?plate)  
 1864 Existing predicates: GripperEmpty(), Holding(?pickupable)



1876 **Figure 19: Predicate Invention Case #1 in Bi-Kuka.** Target predicate: LidOff(?openable)  
 1877 Existing predicates: InLeftGripper(?openable), InRightGripper(?utensil)



1886 **Figure 20: Predicate Invention Case #2 in Bi-Kuka.** Target predicate: RightGripperEmpty()  
 1887 Existing predicates: InLeftGripper(?openable), LidOff(?openable)

1890 **H IMPLEMENTATION DETAILS**1891 **H.1 PLANNER AND PLANNING TIME**

1892 We use K\* planner (Katz & Lee, 2023) to generate top  $K$  optimal plans, where  $K$  in practice is  
 1893 the maximum planning budget. We use an i9-13900F CPU for running all the planning tasks. On  
 1894 average, each planning problem takes 0.0599 seconds. Specifically, in Robotouille experiments,  
 1895 easy problems take 0.0549 seconds, hard problems takes 0.0583 seconds, and impossible problems  
 1896 take 0.0565 seconds per problem; in Franka experiments, in-domain problems take 0.0529 seconds,  
 1897 generalization problems take 0.5175 seconds, and impossible tasks take 0.0516 seconds per problem;  
 1898 in Bimanual Kuka experiments, all problems take 0.0553 seconds on average.  
 1899

1900 **H.2 API CALL**

1901 For running the experiments, we made roughly 9300 calls to GPT-5, which cost \$96.59 in total.  
 1902

1903 **H.3 HYPERPARAMETERS**

1904 We here report and summarize all hyperparameters of SKILLWRAPPER used for the experiment to  
 1905 provide better reproducibility. For all experiments, we set the batch size of skill sequence proposal  
 1906 to be 5 and interaction budget per iteration to be 15, and we run SKILLWRAPPER for 5 iterations.  
 1907 For Robotouille experiments, we set the threshold  $h$  to be 0.6. For Franka and bimanual Kuka  
 1908 experiments, we set the threshold  $h$  to be 0.5.  
 1909

1910 **H.4 ROBOT EXPERIMENTS**

1911 **Single-Arm Manipulation.** We employ a Franka Emika Research 3 robotic arm equipped with  
 1912 a UMI gripper (Chi et al., 2024). The workspace is observed by a single Intel RealSense D455  
 1913 exocentric RGB-D camera, oriented to capture both the tabletop scene and the robot. The RGB data  
 1914 from this camera are used for learning symbolic models, while the depth information supports object  
 1915 pose estimation. Object poses are estimated using FoundationPose (Wen et al., 2024), which leverages  
 1916 high-fidelity 3D scanned models of the target objects. System-level communication and coordination  
 1917 are implemented in ROS 2 (Humble), which interfaces with motion planning, perception, and control  
 1918 modules. This setup supports five parameterized skills: *Pick*, *Place*, *Stack*, *Pour*, and *Wipe*. The  
 1919 first four skills (*Pick*, *Place*, *Stack*, and *Pour*) are executed through motion planning with the MoveIt  
 1920 framework, conditioned on both the end-effector and object poses. The *Wipe* skill is implemented  
 1921 by replaying a teleoperated trajectory.  
 1922

1923 **Bimanual Manipulation.** We use a robot with two horizontally mounted KUKA LBR iiwa 7 R800  
 1924 manipulators, one with a BarrettHand BH8-282 gripper, and the other with a Schunk Dextrous Hand  
 1925 2.0 gripper. The robot collects RGB data used for learning symbolic models with a MultiSense S7  
 1926 camera mounted on a Pan-Tilt unit, while using an Intel RealSense D455 camera for RGB-D data  
 1927 used in pose estimation (Wen et al., 2024) of the objects in the scene. We use ROS 1 and KUKA FRI  
 1928 to communicate with the robot and utilize the built-in joint impedance control with position target as  
 1929 the low-level controller. At the high level, we create collision models of all objects in the scene and  
 1930 use a task and motion planner to generate motion plans for each skill. The *Pick* skills (compatible  
 1931 with knife and peanut butter jar) are implemented using motion planning. The *OpenJar*, *Scoop*, and  
 1932 *Spread* skills are implemented using a combination of motion planning and pre-defined trajectory  
 1933 playback.  
 1934

1935 **H.5 LANGUAGE MODEL PROMPTS**

1936 In this section, we provide the prompts used for the core components of SKILLWRAPPER (specifically  
 1937 skill sequence proposal, predicate invention, and predicate evaluation) as well as the ViLA (Hu  
 1938 et al., 2023) baseline. For predicate evaluation (Appendix H.8), we empirically observed that it is  
 1939 more accurate when evaluation is done in batches, where the truth values of multiple predicates are  
 1940 evaluated at once rather than one at a time. In addition, when asking for a fixed and structured  
 1941 output, the accuracy is significantly lower than a free-form output. Therefore, we adopt a two-stage  
 1942 evaluation process: in the first stage, the foundation model generates a response in any format, and  
 1943 in the second stage, it provides a summary of the output from the previous step.

1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997

## H.6 SKILL SEQUENCE PROPOSAL

### System Prompt

<AGENT\_DESCRIPTION> is attempting to learn the preconditions and effects for a finite set of skills by executing exploratory skill sequences and exploring the environment.

### Skill Sequence Proposal Prompt

Propose a set of skill sequences for a robot to execute. The robot is attempting to learn the preconditions and effects for a finite set of operators. The robot can navigate the environment freely but only has one gripper. The robot has access to the following skills with their associated arguments:

[SKILL\_PROMPT]

The list of objects the robot has previously encountered in the environment are:

[OBJECT\_IN\_SCENE]  
[ENV\_DESCRIPTION]

The pairs of consecutive skills (skill1, skill2) that have been least explored are: [[LEAST\_EXPLORED\_SKILLS]]. Certain skills have similar names and arguments, but different preconditions and effects. Using the list of objects and the skill preconditions / effects learned, generate 5 skill sequences and their sequence of skills such that:

- (1) the skill sequences should violate their preconditions occasionally.
- (2) at least 1 unexplored skill pair is used in each skill sequence.
- (3) all skill sequences have at least 15 skills in sequence.
- (4) there are no same skills with same arguments consecutively in the sequence.

Output only the sequence of skills to execute, ensuring to follow the naming/syntax/arguments for skills provided. Output 1 skill every new line, following the format below:

Skill Sequence 1:  
GoTo(CounterTop)  
PickUp(Apple, CounterTop)

Skill Sequence 2:

1998

## H.7 PREDICATE INVENTION

1999

2000

2001

2002

2003

2004

2005

## Predicate Invention Prompt

2006

[AGENT\_DESCRIPTION]

2007

The robot has been programmed with the skill [LIFTED\_SKILL] two times. In the first execution, the grounded skill [GROUNDED\_SKILL\_1] [SUCCESS\_1], and in the second execution, [GROUNDED\_SKILL\_2] [SUCCESS\_2]. The difference in outcomes suggests that the existing predicate set is insufficient to fully capture the preconditions for successful execution of this skill.

2008

Your task is to propose a single new high-level predicate and its semantic meaning based on the visual comparison of the two input images taken before each execution.

2009

Predicates should meet these criteria:

2010

- The predicate must be grounded in visual state only (e.g., "gripper is open," "object is above table," "arm is holding object").
- Describe object state or spatial relations relevant to task success (e.g., gripper open/closed, object on left/right of gripper, object touching/supporting another object, etc.)
- Do not infer properties like affordances (is\_graspable), alignment with grippers, or success likelihood that are vaguely defined and cannot be clearly determined visually.
- Avoid using concept like grasping zone or robot's reachability to define the predicate since they are not defined by common sense.
- Use at most 2 parameters (e.g., predicate(x), predicate(x, y), predicate()), where robot arm must be included for any robot-environment relation.
- Avoid predicates that assume internal properties like is\_graspable, is\_properly\_aligned, or any accessibility/reachability reasoning that cannot be determined visually.
- The semantic meaning should be a grounded and objective description of the predicate in terms of the physical scene (e.g., "the object is fully enclosed by the robot's gripper"), not about execution success or skill dynamics.
- The parameters of the predicate must be subset of the parameters of the skill.

2011

Format your output as follows:

2012

'predicate\_name(parameters) ': semantic\_meaning.

2013

for example:

2014

'CloseTo(arm, location) ': the robot arm is close to the location.

2015

Current predicates: [PRED\_LIST]

2016

Previously proposed but rejected predicates: [TRIED\_PRED]

2017

Avoid duplicates or near-duplicates of existing predicates and rejected predicates. Reason over using a paragraph and generate the predicate and the semantic meaning in the given format in a separate line.

2018

One new predicate candidate for improving the representation of the precondition for [LIFTED\_SKILL] (Don't use any parameter other than [PARAMETERS]):

2052  
2053

## H.8 PREDICATE EVALUATION

2054  
2055

## Predicate Evaluation Prompt: Step 1

2056  
2057  
2058

Given the current observation of the simulated kitchen domain, the object types, and the list of predicates, what are the true grounded predicates?

2059  
2060

[ENVIRONMENTAL\_DESCRIPTION]

2061  
2062

Objects:  
[OBJECTS]

2063  
2064  
2065

Predicates:  
[PREDICATES]

2066  
2067

## Predicate Evaluation Prompt: Step 2

2068  
2069  
2070  
2071  
2072  
2073  
2074

Summarize what the true grounded predicates are from this response, and list them in the format of predicate\_name(arg1, arg2, ...) in separate lines with no any formatting. If the response contains typos of object names or redundant indices, you should correct them. Correct object names are: [OBJECT\_NAMES]. If the response include redundant predicates that are not in this list, you should filter them. Correct predicates are: [PRED\_NAMES]. The response is:

2075  
2076  
2077  
2078

"""  
[RESPONSE]  
""

2079

## H.9 ViLA

2080  
2081  
2082  
2083  
2084  
2085  
2086

## ViLA Prompt

You are [AGENT\_DESCRIPTION]. As a robot, you are able to execute the following skills:  
[SKILLS]

2087  
2088  
2089

Here are the objects and their types that are compatible with your skills:

[OBJECTS]

2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099

You are given two images: The first one captures your current observation, and the second one specifies your goal. Given both images, your job is to generate a plan starting from the \*current state\* to the goal state. You should first reason about the goal of the task and how the skills can be chained to solve it in the first paragraph. After the reasoning, return the plan from the current state in a new paragraph by listing skills in separate lines with no additional explanation, header, or numbering. Use "Done" in the skill list to indicate the task is complete, and report if the task is impossible to solve by simply returning "Impossible".

2100  
2101  
2102  
2103  
2104  
2105

## 2106 I RELATED WORKS

2108 **Skill Abstraction.** There has been a long track of works focusing on building hierarchies that  
 2109 abstract away high-dimensional details with low-dimensional abstractions for planning (Konidaris  
 2110 & Barto, 2009; Konidaris et al., 2018; Shah et al., 2024), and those applied to robotics are usually  
 2111 connected to task and motion planning (TAMP) (Shah et al., 2020; Garrett et al., 2021). These  
 2112 approaches, however, are incapable of handling high-dimensional sensory-motor signals (such as  
 2113 images) as input. Research on action model learning (Xi et al., 2024; Juba et al., 2021) learn  
 2114 symbolic action models for input skills. However, unlike our method, these approaches require  
 2115 symbols to be provided as input. Similar to our system’s integration of self-play and focus on  
 2116 uncovering skill conditions, Verma et al. (2022) focus on assessing capabilities of black-box agents  
 2117 for grid world-like tasks while assuming that the agent is an oracle. A tangential research effort on  
 2118 chaining various skills in novel environments involves training extra models (Yokoyama et al., 2024)  
 2119 and STRIPS task planner with action primitives (Gu et al., 2022; Szot et al., 2021).

2120 **Predicate Learning for Robotic Tasks.** Predicates provide a convenient way to abstract away low-  
 2121 level details of the environment and build efficient and compact representations. Prior to foundation  
 2122 models, previous attempts to build classifiers for predicates from raw image inputs originated from  
 2123 the neuro-symbolic domain (Johnson et al., 2017; Mao et al., 2019), and their initial application  
 2124 for robotics took a similar supervised learning approach with labeled demonstrations (Migimatsu  
 2125 & Bohg, 2022) or generated tasks (Lamanna et al., 2023). After the emergence of foundation  
 2126 models, recent works guide skill learning with predicates generated by LLM or together with human  
 2127 interaction. Li et al. (2024) invents symbolic skills for reward functions used for RL training but  
 2128 cannot generalize to skills learned through latent objectives, which is more commonly seen in  
 2129 imitation learning. Li & Silver (2023) and Han et al. (2024) leverage human experts to provide  
 2130 feedback to the LLM to help it improve the learned predicates and skills.

2131 **Task Generation for Robotics.** The approach of automatically proposing tasks has been studied  
 2132 for active learning and curriculum learning in grid worlds and games (Wang et al., 2019; Jiang et al.,  
 2133 2021) to robotic domains (Fang et al., 2021; 2022). Lamanna et al. (2023) generates tasks in PDDL  
 2134 as training sets to learn classifiers for object properties in predicates format, while they assume the  
 2135 action operators are given. With the commonsense reasoning ability of foundation models, recent  
 2136 works have applied the idea of automatic task proposing and self-playing for exploration (Nasiriany  
 2137 et al., 2024; Ren et al., 2024), data collection (Wang et al., 2024c; Yang et al., 2024; Ahn et al.,  
 2138 2024), boosting skills learning (Ha et al., 2023; Wang et al., 2024a), and scene understanding (Jiang  
 2139 et al., 2025). These works indicate a promising direction for generating robotic data and scaling  
 2140 up. Following the idea, we equipped our system with a task-proposing module for generating  
 2141 skill sequences specific to skills and predicates, which serves the idea of both data collection and  
 2142 exploration.

2143 **Embodied Reasoning with Foundation Models.** There has been a track of work on leveraging  
 2144 large language models (LLMs) for embodied decision-making (Huang et al., 2022; Raman et al.,  
 2145 2024) and reasoning (Huang et al., 2023), while vision-language models (VLMs) are often considered  
 2146 to have limited embodied reasoning ability due to their pre-training corpora that focus primarily on  
 2147 language generation (Valmeeekam et al., 2023). Common ways of addressing this issue include fine-  
 2148 tuning on datasets from a specific domains (Hong et al., 2023; Mu et al., 2024; Chen et al., 2024) or  
 2149 knowledge distillation (Sumers et al., 2023; Yang et al., 2024). Meanwhile, many works manage to  
 2150 leverage preexisting models without further training from direct visual observation (Fang et al., 2024;  
 2151 Nasiriany et al., 2024) to complete robotic tasks (Jiang et al., 2025). In these works, the embodied  
 2152 reasoning ability of the foundation models serves as the central part of the systems. However, most  
 2153 benchmarking works evaluate the embodied reasoning ability of the models in a question-answering  
 2154 fashion (Sermanet et al., 2024; Majumdar et al., 2024; Cheng et al., 2024; Chen et al., 2025), where  
 2155 it remains unclear whether they are capable of solving robotic tasks.

## 2156 J USE OF LARGE LANGUAGE MODELS

2157 Our work incorporates language models as part of SKILLWRAPPER, particularly for the three important  
 2158 components of our system discussed in Section 3. We utilize OpenAI’s GPT-5 (OpenAI, 2025) as

2160 our foundation model of choice. We acknowledge that all the content of the manuscript has been  
2161 generated by the authors. However, we have used LLMs for basic editing, polishing, and grammar  
2162 checking.

2163

2164

2165

2166

2167

2168

2169

2170

2171

2172

2173

2174

2175

2176

2177

2178

2179

2180

2181

2182

2183

2184

2185

2186

2187

2188

2189

2190

2191

2192

2193

2194

2195

2196

2197

2198

2199

2200

2201

2202

2203

2204

2205

2206

2207

2208

2209

2210

2211

2212

2213