From Reals to Logic and Back: Learning World Models for Planning from Raw Data

Naman Shah[1, 2, *], **Jayesh Nagpal**[1], **and Siddharth Srivastava**[1]

[1]School of Computing and Augmented Intelligence, Arizona State Univeristy, Tempe, AZ, USA
[2]Brown University, Providence, RI, USA
naman_shah@brown.edu, nagpal.jayesh@asu.edu, siddharths@asu.edu

Abstract

Humans build generalizable world models for planning from sparse amounts of data. Today's robots however, are extremely limited in their ability to create world models that generalize, and rely upon expert human input to obtain the world models necessary for solving complex tasks. This dependence is particularly prominent and limiting in settings where test tasks feature more objects and longer horizons than those encountered during learning. We present the first approach for autonomously learning generalizable, logic-based world models for planning, starting from small batches of unannotated high-dimensional, real-valued robot trajectories. Our approach uses unsupervised learning to invent a relational vocabulary in first-order logic along with high-level actions and logic-based models for the invented actions. Empirical results show that generalizable world models can be learned from just a handful of robot trajectories; that the learned models include but go beyond the classical, hand-crafted conceptualizations of useful relations and actions; and that the learned models allow robot planning to scale to tasks that were previously beyond the scope of planning without hand-crafted models.

1 Introduction

Enabling robots to learn how to accomplish complex tasks is essential for autonomous, reliable robotics. Recent research makes progress on robust training and adaptation for tasks with either short horizons (e.g., picking up an object or closing a door) (Hafner et al. 2023) and behavior cloning for longer horizons in settings where test tasks have only minor differences from training scenarios (Fu, Zhao, and Finn 2024). This is in large part because the ability to learn from simple problems and then solve more complex problems requires the invention and **transfer of generalizable, abstract concepts and world models**. For instance, all the concepts necessary for clearing a table cluttered with cups are present in one robot trajectory that picks up and places a cup (with no annotations or segmenting). It should be possible to clear a table scattered with cups given handful of demonstrations for picking and placing a single cup. Yet, this problem remains a difficult generalization task for robotics – in large part due to the lack of methods for learning without human intervention, a transferrable conceptual vocabulary that can support the learning of generalizable world models.

Classically, such problems are solved in AI using logic-based world models that are crafted by humans. The resulting approaches offer impressive robustness on tasks that are within the scope of the model (Srivastava et al. 2014; Garrett, Lozano-Pérez, and Kaelbling 2020; Shah et al. 2020), but require extensive domain-engineering effort and are limited to the problem domains envisaged prior to deployment. This precludes the deployment of autonomous robots in dynamic scenarios where environments and tasks can differ from their idealized design-stage conceptions – precisely the scenarios where autonomy is needed.

This paper presents the first approach to learning logic-based world models from raw demonstrations of robot trajectories without any human annotation, segmenting, or guidance about primitive controllers, actions, or concepts. Our results show that in a variety of mobile manipulation tasks, this approach enables the robot to autonomously invent a generalizable concept vocabulary and learn generalizable world models using these self-invented concepts. The robot is then able to reason and plan over the learned models to solve new tasks that feature much longer execution horizons and many more objects in a zero-shot fashion.

Prior research on the topic indicates that partial world models can be learned if the agent is provided with primitive concepts (Silver et al. 2022, 2023) or primitive actions (Konidaris, Kaelbling, and Lozano-Pérez 2018). This paper shows for the first time that it is possible to learn such models entirely from raw data, without human input of such primitives. It develops a transferrable notion of "salient regions of the state space" in terms of relational critical regions, which yield new approaches for inventing a general form of a logical predicate vocabulary from solutions to simple problems. This vocabulary turns out to be sufficient for autonomously inventing high-level "actions" that take the agent from one salient region to another, and for learning powerful logic-based world models for such actions. The resulting knowledge allows the agent to solve, in zero-shot fashion, problems that are much larger than those used for learning the concepts and actions.

[*]Work primarily done at Arizona State University

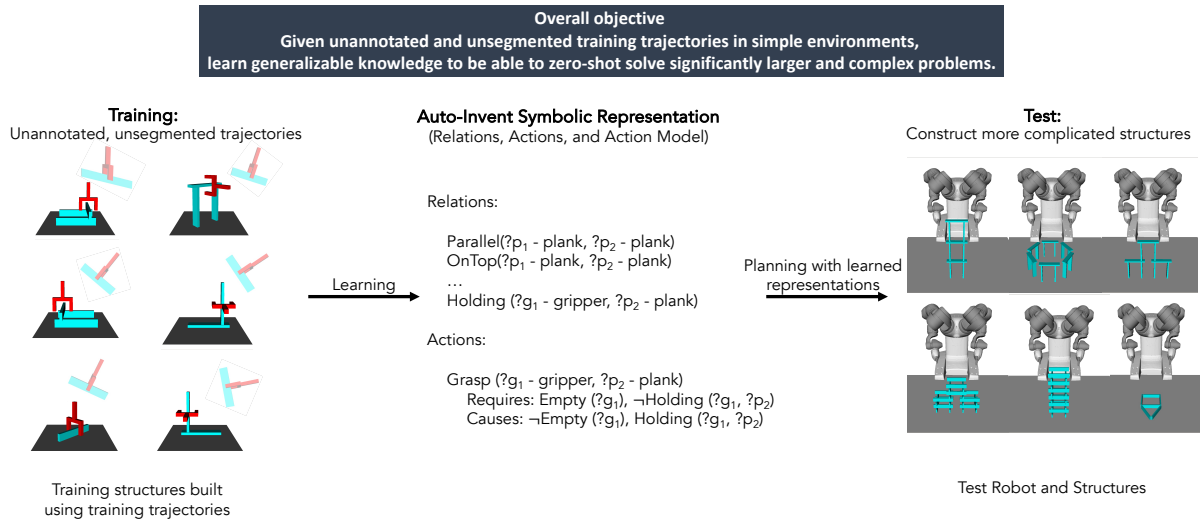


Figure 1: Our overall approach. We start with a set of demonstrations on relatively simple tasks using a simple robot and learn a symbolic model in the form of a set of relations and high-level actions. This symbolic model can be used with any off-the-shelf planner for solving unseen complex long-horizon planning problems with other similar robots.

2 Preliminaries

We consider a setting where the environment comprises objects and robots. A state of each object each defined as a 6D pose of the object. A robot, a special object, is a kinematic chain of links and joints, with its state as $\langle P_{\text{base}}, x \rangle$. Here, P_{base} is the 6D pose of the first link, and x is the joint value for each joint. For an environment with objects $\mathcal{O} = \langle o_1, \ldots o_n, r_1, \ldots r_m \rangle$, the state space is $\mathcal{X} = \mathcal{X}_{r_i} \times \mathcal{X}_{o_j}$ for every robot $r_i \in \mathcal{O}$ and object $o_j \in \mathcal{O}$. A collision function c divides the state space $\mathcal{X}$ into two sets: $\mathcal{X}_{\text{free}}$ (collision-free states) and $\mathcal{X}_{\text{obs}}$ (colliding states).

Native robot actions allow robots to alter their state, including configuration and base link pose, enabling movement and object manipulation within the environment. A primitive action a defines a deterministic function $a : x \mapsto x'$. Environment states $x \in \mathcal{X}$ and $x' \in \mathcal{X}$ indicate that applying action a in state x results in state x'. We define a robot planning problem as follows:

Definition 1 *A **robot planning domain** is defined as a tuple $\langle \mathcal{O}, \mathcal{T}, \mathcal{X}, \mathcal{A}, x_i, \mathcal{X}_g \rangle$ where $\mathcal{O}$ is a set of objects, $\mathcal{T}$ is a set of object types, $\mathcal{X}$ is a state space, and $\mathcal{A}$ is an uncountably infinite set of native deterministic actions. $x_i \in \mathcal{X}_{free}$ is an initial state of the environment, and $\mathcal{X}_g \subseteq \mathcal{X}$ is a set of goal states. A solution to a planning problem is a sequence of native actions $a_0, \ldots, a_n$ such that $a_n(\ldots(a_0(x_i))) \in \mathcal{X}_g$.*

Symbolic abstractions We consider symbolic world models as first-order logic models and represent them using the PDDL representation (McDermott et al. 1998). The PDDL domain consists of two components $\langle \mathcal{V}, \bar{\mathcal{A}} \rangle$. Here, $\mathcal{V}$ is a set of symbolic relations, and $\bar{\mathcal{A}}$ is a set of high-level robot actions. Relations $R \in \mathcal{V}$ parameterized by typed parameters describe object relations. Relations $R \in \mathcal{V}$ can be grounded with objects, called r when ungrounded and r' when grounded. Grounded relations R' act as Boolean classifiers, true in a low-level state x (denoted $R'_x = 1$) if the relation holds for the grounded objects in state x. An abstraction function $\alpha : x \mapsto s'$ evaluates all grounded relations in a low-level state $x \in \mathcal{X}$ to yield an abstract grounded state $s' \in 2^{\mathcal{V}'}$. A symbolic grounded state s' is a set of true grounded relations in the low-level state x, referred to as s', while the symbolic lifted state is s ($s \in 2^{\mathcal{V}}$).

$\bar{\mathcal{A}}$ outlines symbolic lifted actions using lifted relations $\mathcal{V}$. Each action $\bar{a} \in \bar{\mathcal{A}}$ has typed symbolic parameters. $\bar{a}$ is defined as a tuple $\langle pre_{\bar{a}}, eff_{\bar{a}} \rangle$, where $pre_{\bar{a}}$ is a conjunctive formula of parameterized relations $\mathcal{V}$. The action's effect eff_a is a tuple $eff_{\bar{a}} = \langle add_{\bar{a}}, del_{\bar{a}} \rangle$ adding relations $add_{\bar{a}}$ and removing relations $del_{\bar{a}}$ from the state once the action $\bar{a}$ is executed. Actions $\bar{a}$ can be grounded to specific objects, resulting in grounded actions $\bar{a}'$, generating grounded precondition $pre_{\bar{a}'}$ and effect $eff_{\bar{a}'}$. A grounded action $\bar{a}'$ is applicable in a state s only if $pre_{\bar{a}'} \models s$. Every deterministic grounded action $\bar{a}' \in \bar{\mathcal{A}}'$ maps each symbolic state s_i to a new state s_j.

Symbolic plans cannot be executed by a robot. It needs to be converted to a sequence of primitive actions that a robot can execute. Task and motion planning approaches use abstract symbolic models along with *pose generators* for computing a sequence of primitive actions for planning problems. A pose generator defines an inverse abstraction function. Let Γ_R be a pose generator for a lifted symbolic predicate $R \in \mathcal{V}$. For a grounded relation R', a pose generator $\Gamma_{R'} = \{x | x \in \mathcal{X} \wedge R'_x = 1\}$. A pose generator for a grounded state s' is defined as $\bigcap_{\forall R' \in s'} \Gamma_{R'}$.

Critical regions We use the concept of *critical regions* for automatically inventing a predicate vocabulary. Molina, Kumar, and Srivastava (2020) and Shah and Srivastava (2022) propose the concept of a critical region in the configuration space of a robot for learning propositional symbolic ab-

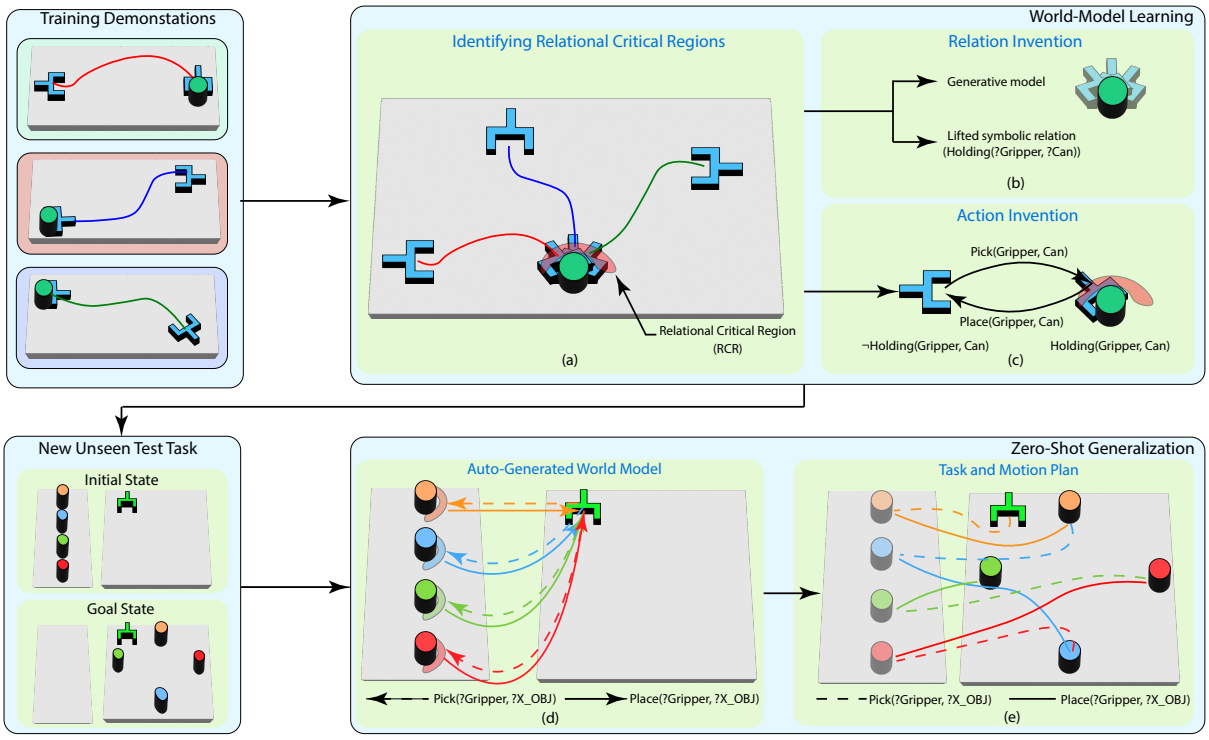


Figure 2: Illustration of our overall method. We start with a set of unlabeled, unsegmented training demonstrations and learn relational salient region predictors (a). Each of these generative predictors define a unique relation between a pair of objects (b) and actions that make this relation true or false (c). Given a new test task, we use the learned predictors to generate salient regions and high-level robot actions (d) that support off-the-shelf task and motion planning (e).

stractions. Critical regions generalize the concepts of hubs or access points and bottlenecks or pinch points in a single concept. Earlier work defines critical regions in a goal-agnostic manner, however, in this work we consider goal-conditioned critical regions. Intuitively, as the name suggests, goal-conditioned critical regions learn critical regions for a specific training problem. In this work, we learn goal-conditioned critical regions for each training task and combine them in order to compute the set of critical regions. Given a robot with a configuration space $\mathcal{X}$, goal-conditioned regions are defined as follows.

Definition 2 *Given a set of solutions for a robot planning problem T, the measure of criticality of a Lebesgue-measurable open set $\rho \subseteq \mathcal{X}$, $\mu(\rho)$, is defined as $\lim_{s_n \to^+ r} \frac{f(r)}{v(s_n)}$ where $f(r)$ is a fraction of observed motion plans solving the task T that pass through s_n, $v(s_n)$ is the measure of s_n under a reference density (usually uniform), and $\to^+$ denotes the limit from above along any sequence $\{s_n\}$ of sets containing ρ ($\rho \subseteq s_n, \forall n$).*

3 Learning Relational Vocabulary

We postulate that high-level robotic actions currently crafted by experts are effectively transitions to and from certain salient regions within the environment and that such regions can be automatically discerned. A region is deemed salient if it is essential for the completion of a specified task. For instance, as illustrated in Fig. 2, where the gripper is tasked with picking up the can, the area surrounding the can, from which it can be grasped, constitutes a salient region. The "Pick" and "Place" actions correspondingly transition the gripper into and out of this salient region. Our hypothesis is that if we can automatically identify such salient regions, such high-level actions can also be invented autonomously, subsequently enabling autonomous formulations of generalizable concepts for long-horizon problems.

3.1 Relational Critical Regions

We start by characterizing such salient regions in the environment as *relational critical regions (RCRs)*. Critical regions (CRs) (Shah and Srivastava 2022) help identify these regions in the robot's configuration space (LaValle 2006). However, for long-horizon reasoning involving various objects, salient regions are in the relative space between objects, not the robot's configuration space. For instance, Fig 2(a) shows the salient region for the gripper and the can (shaded red) in their relative space, regardless of the can's location, forming a relational critical region. We thus extend critical regions to the relative state spaces between object pairs and introduce *relational critical regions*. Given a pair of objects, we define a relational critical region as a region of the relative state space that has a high density of successful solutions given a distribution of tasks. Formally, we define relational critical regions as follows.

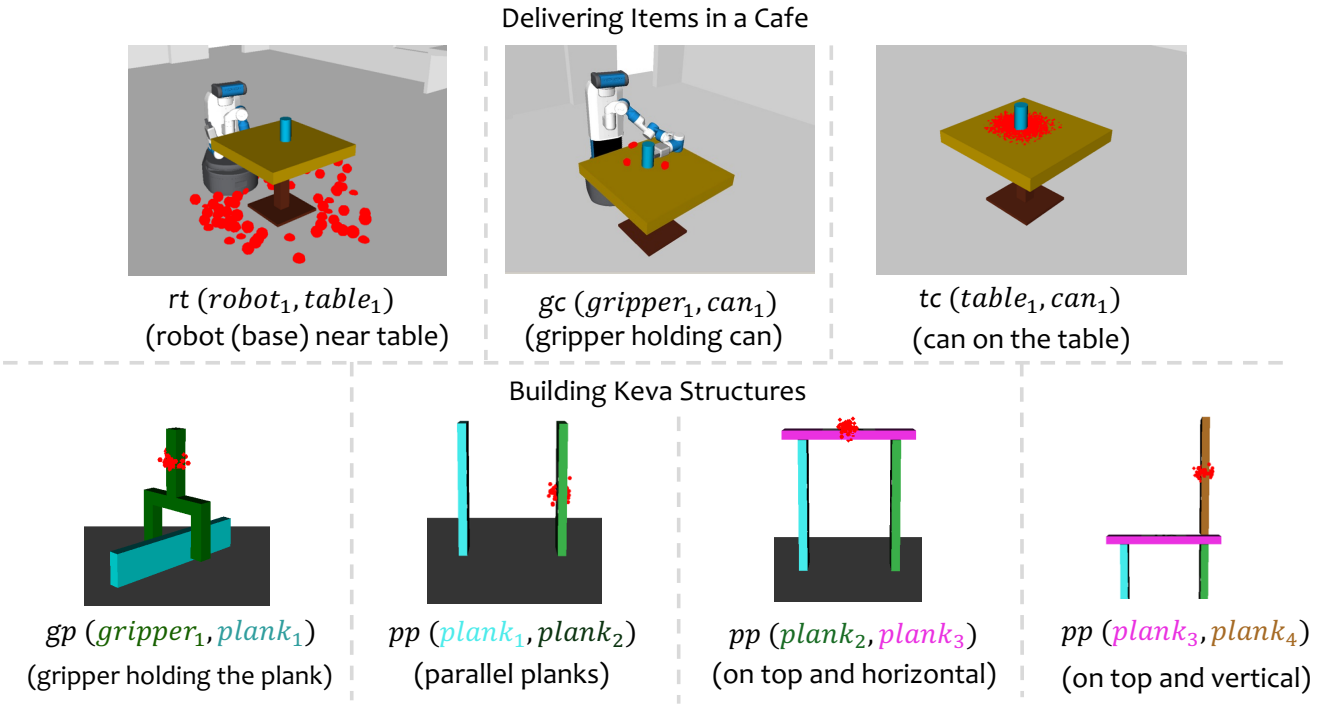


Figure 3: Different relations invented by our approach and their corresponding critical regions. Each image shows one binary predicate and its semantic interpretation. The red dots show sampled possible poses for the object in the relational critical region.

Definition 3 *Let T be a robot planning problem and $\mathcal{D}_T$ be a set of solution trajectories for the planning problem T. Let $o_1, o_2 \in \mathcal{O}$ be a pair of objects, and let $\mathcal{X}_{o_2}^{o_1}$ define the relative state space for object o_2 in the relative reference frame of the object o_1. The measure of the criticality of a Lebesgue-measurable open set $\rho \subseteq \mathcal{X}_{o_2}^{o_1}$, $\mu(\rho)$, is defined as $\lim_{s_n \to^+ \rho} \frac{f(\rho)}{v(s_n)}$ where $f(\rho)$ is a fraction of observed solution trajectories solving for the planning problem T that contains a relative pose $P_{o_2}^{o_1}$ such that, $P_{o_2}^{o_1} \in \rho$ $v(s_n)$ is the measure of s_n under a reference density (usually uniform), and $\to^+$ denotes the limit from above along any sequence $\{s_n\}$ of sets containing ρ ($\rho \subseteq s_n, \forall\, n$).*

We begin with training demonstrations that solve simple tasks, converting them to trajectories between object pairs in the relative space. Using Def. 3, we identify relative critical regions Ψ for these tasks. Our method employs multivariate Gaussian mixture models to learn generative multivariate Gaussian predictors for each region. Formally, let $\Psi_{ij} \subset \Psi$ represent relational critical regions for object types τ_i and τ_j. With a threshold ϵ, the Gaussian mixture model (GMM) estimates Gaussian parameters μ_ψ and Σ_ψ for every region $\psi \in \Psi_{ij}$ ensuring support for each pose $P \in \psi$ exceeds predefined threshold ϵ. We say a relative pose $p_{o_1}^{o_2}$ is in the relational critical region ψ (denoted by $p_{o_1}^{o_2} \in \psi$) iff $\psi(p_{o_1}^{o_2}) = 1$, i.e., support for the relative pose $p_{o_1}^{o_2}$ under the Gaussian parameters μ_ψ and Σ_ψ is greater than ϵ. For implementation, we use `label` from the `OpenCV` python package[1] to extract connected components in a region, utilizing that number as components for the multivariate mixture model in the scikit-learn package[2].

3.2 Generating Relational Vocabulary

elational critical regions represent salient regions in the environment. However, they are insufficient for generalizable long-horizon reasoning; high-level reasoning requires abstract actions, which in turn require a relational vocabulary to represent pre- and post-conditions of these actions. Therefore, for each relational critical region predicted by the learned multivariate Gaussian predictors, our method invents one unique binary relation between the corresponding objects. E.g., the relational critical region shown in Fig. 2(a) represents the relation concept equivalent to `Holding(Gripper, Can)` relation between the gripper and the can, and it is true when the gripper is in the shaded red region around the can.

More specifically, let $\mathcal{O}_{\tau_i}$ and $\mathcal{O}_{\tau_j}$ be the set of objects of type τ_i and τ_j, respectively, and let Ψ_{ij} be a set of relation critical regions predictors between $\mathcal{O}_{\tau_i}$ and $\mathcal{O}_{\tau_j}$. We define a unique binary relation $R_{ij}^k : \mathcal{O}_{\tau_i} \times \mathcal{O}_{\tau_j} \to \{true, false\}$ for each relation region predictor $\psi^k \in \Psi_{ij}$ such that $R_{ij}^k(o_i, o_j) = \textit{true}$ iff for the relative pose $p_{o_i}^{o_j}$, $\psi^k(p_{o_i}^{o_j}) = 1$.

[1] https://github.com/opencv/opencv-python
[2] https://scikit-learn.org/1.5/modules/mixture.html

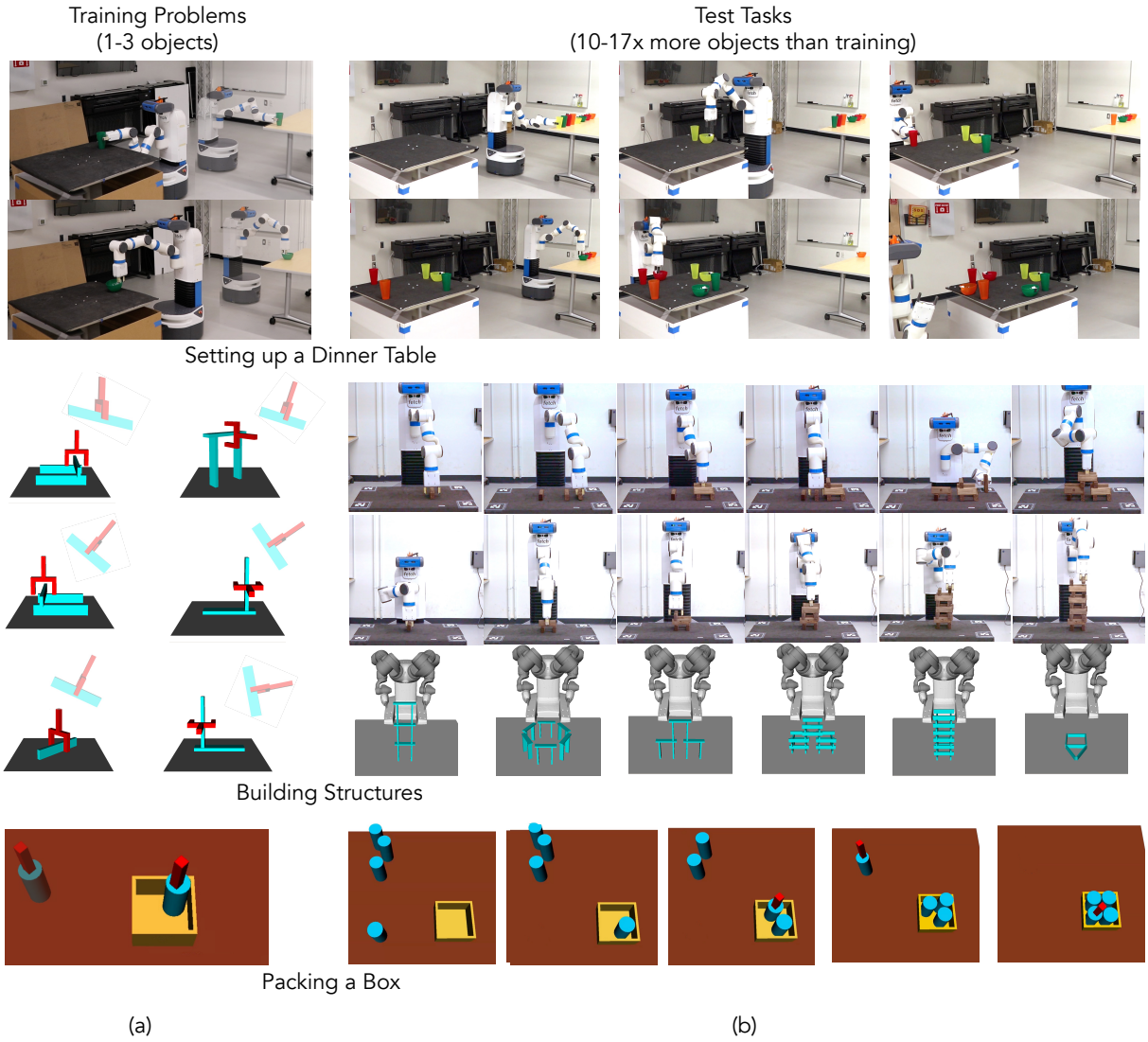


Figure 4: Generalization across different tasks. (a) shows the training tasks used to learn relational region predictors and (b) shows the test tasks used to evaluate the overall method. We show that the predictors learned from very simple tasks can generalize to significantly complex test tasks that include more complex goals, more complex environments, and a larger number of objects (up to 18x). Note: training demonstrations for Keva and Jenga structures include same structures but different robots to match the gripper configuration of the robot used while testing.

Here, $p_{o_i}^{o_j}$ is the pose of the object o_i relative to the object o_j.

We define two additional types of boolean relations. First, given the sets of objects $\mathcal{O}_{\tau_i}$ and $\mathcal{O}_{\tau_j}$, we define a relation $R'_{ij} : \mathcal{O}_{\tau_i} \times \mathcal{O}_{\tau_j} \rightarrow \{true, false\}$ such that $R'_{ij}(o_i, o_j) \implies \forall k \left[\neg R^k_{ij}(o_i, o_j)\right]$. Second, we define a relation for each relational region predictor for representing the occupancy of the predicted region. Intuitvely, this models the free volume in the predicted region. E.g., the relational critical region between a gripper and the object (Fig. 2) can be only occupied by a single object but the relational critical region for a table and a can (Fig. 3 can be occupied by multiple cans. Specifically, given the sets of objects $\mathcal{O}_{\tau_i}$ and $\mathcal{O}_{\tau_j}$, a set of relational regions predictors Ψ_{ij}, we define a boolean relation for each relational region predictor $\psi^k \in \Psi_{ij}$, $R^{free}_{ij} : \mathcal{O}_{\tau_i} \rightarrow \{true, false\}$ that is true iff for objects $o_i \in \mathcal{O}_{\tau_i}$ and $o_j \in \mathcal{O}_{\tau_j}$, $\rho_{free}(\psi^k, o_i) > \rho(o_j)$. Here, $\rho_{free}(\psi^k, o_i)$ is the free volume of the region predicted by ψ^k for the object o_i and $\rho(o_j)$ is the volume of the object $o_j \in \mathcal{O}_{\tau_j}$.

Given a new task T and the set of objects $\mathcal{O}_T$, we use the automatically learned relational region predictors to predict the critical regions for objects and then generate the relational vocabulary $\mathcal{V}_T$ for the new task T. This vocabulary now can be used to represent each configuration $x \in \mathcal{X}_T$ as a high-level state as a set of true relations given the configuration x.

4 Synthesis of Generalizable Actions and World Models

The last step in the overall world model learning algorithm is to synthesize generalizable and transferrable actions, models, and action interpreters. Recall that we hypothesized that high-level actions are transitions to and from relational critical regions. E.g., the transition in and out from the relational critical region in Fig. 2 induces high-level actions `Pickup(Gripper, Object)` and `Place(Gripper, Object)` respectively. Therefore, we use the predicted critical regions in training tasks to invent robot high-level actions. However, using the exhaustive enumeration of the set of predicted regions may lead to a large number of actions, most of which may be infeasible to realize. Instead, we create high-level robot actions that facilitate transitions between abstract states in our training demonstrations for forming the relational concept vocabulary. This approach ensures that all the created actions are practically achievable for the robot.

We invent high-level robot actions as transitions between high-level states in the training demonstrations. First, we convert every native action transition $\langle x_0, \dots x_n \rangle$ to abstract transitions $\langle s'_0, \dots, s'_n \rangle$ using the learned relational vocabulary $\mathcal{V}$ and the set of objects $\mathcal{O}$. Here, each abstract state $s'_i = \{R^k(o_i, o_j) | \forall o_i, o_j \in \mathcal{O}, \forall R^k \in \mathcal{V}, x \models R^k(o_i, o_j)\}$. We then convert these grounded abstract transitions to lifted transitions $\langle s_0, \dots, s_n \rangle$ by replacing specific objects in the training transitions and replacing them with placeholder objects of the object type. Next, for each transition $C_{ij} = s_i \rightarrow s_j$, we compute sets of added and deleted relations such that $C^+_{ij} = s_j \setminus s_i$ and $C^-_{ij} = s_i \setminus s_j$ and cluster all transitions from the training demonstrations that induce the same $\langle C^+_{ij}, C^-_{ij} \rangle$. Each cluster induces a high-level action $\bar{a_i} \in \bar{\mathcal{A}}$.

Once a set of high-level actions $\bar{\mathcal{A}}$ is identified, we use associative learning to learn a symbolic model for each high-level action $\bar{a} \in \bar{\mathcal{A}}$. A symbolic model for an action is represented in terms of its symbolic effects, symbolic preconditions, and action parameters. Our approach also learns the symbolic model for each high-level action using the set of training demonstrations $\mathcal{D}_{train}$ as follows.

In our setting, effect of an action $\bar{a}$ is represented as $\mathit{eff}_{\bar{a}} = \langle \mathit{add}_{\bar{a}}, \mathit{del}_{\bar{a}} \rangle$. Each cluster $c_i \in \mathcal{C}$ is generated by clustering transitions the sets of changed relations. These changed relations correspond to added and removed relations as an effect of executing the action induced by the cluster. Therefore, for an action $\bar{a}_i$ induced by the cluster c_i with a set of changed relations $C_i = \langle {}^+C_i, {}^-C_i \rangle$, $\mathit{add}_{\bar{a}_i} = {}^+C_i$ and $\mathit{del}_{\bar{a}_i} = {}^-C_i$.

To learn the precondition of an action, we take the intersection of all states where the action is applicable. Given a set of relations, this approach generates a maximal precondition that is conservative yet sound (Wang 1994; Stern and Juba 2017). We learn the precondition of an action $\bar{a} \in \bar{\mathcal{A}}$ corresponding to a cluster $c = \langle S_i \rightarrow S_j, C_{ij} \rangle$ $\mathit{pre}_{\bar{a}} = \cap_{s \in S_i} s$.

Each action can have spurious preconditions corresponding to static relations that do not change when the action is applied, but are still true in all the pre-states. Therefore, we remove relations from the learned precondition that (i) are not parameterized by any of the objects that are changed by the action and (ii) are not changed at any point in any of the demonstrations. This removes any predicate from the precondition that is spurious with respect to the data.

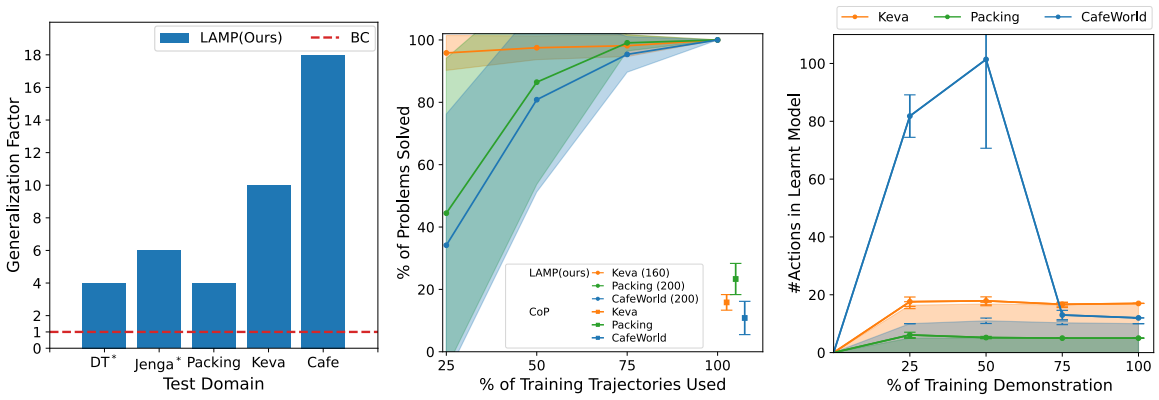
Once the precondition and effect of an action are learned, the final step is to learn the parameters of the action that can be replaced with objects in order to ground the action. In this step, the relations in precondition and effect are processed in order. These relations are processed in alphanumeric order and each of their parameters is added to the action's parameter list, if not already added. This process leads to an ordered list of parameters of the action, which can be grounded with compatible objects.

5 Empirical Evaluation

We evaluate our method in five different settings that reflect real-world complex challenges. These settings involve different robots operating in both simulated and real-world environments. In two of these settings, robots (YuMi and Fetch) use Keva planks and Jenga planks, respectively, to build various structures based on input goals. Another two settings simulate a café environment where the robot acts as an assistant to deliver food items in one case (simulated), and to set up dining utensils (bowls and glasses) on tables in another (real-world). Finally, we assess our approach in a factory setting where the robot needs to efficiently pack all objects into a small box. In each setting, we collect training demonstrations in the simulator and evaluate the learned models either in the simulator (for the café, Keva structures, and packing tasks) or in the real world (for the dining table and Jenga structures).

The closest existing line of work is known as behavior cloning. These approaches (Brohan et al. 2022; Zhao et al. 2024; Black et al. 2024) use similar training demonstrations as our approach; however, instead of learning structured world models, they learn a single latent policy that directly predicts primitive robot actions. These approaches suffer from two major limitations: (i) lack of generalization, i.e., the learned policies only apply to the training settings with the same objects and configurations, and (ii) low sample efficiency, i.e., they require a large number of training demonstrations.

Generalization We emphasize that a generalist robot must have the ability to generalize to different quantities and configurations of objects, i.e., it must have a high *generalization factor*. We define the generalization factor as the ratio of the maximum number of objects in test tasks to the maximum number of objects in training tasks. In all of our settings, we learn our relational world models in training environments with only 1-3 objects and evaluate their performance on tasks that involve significantly more objects and different settings than the training tasks. Fig. 5(a) shows the generalization factor for our approach in the test tasks. These generalization factors are computed over 12 unique test tasks for each setting with varying numbers of objects or problem settings. In the café settings , we successfully tested our approach with 20 and 8 objects in simulated and real-world set-



(a) (b) (c)

Figure 5: Empirical evaluation of the system. (a) shows the generalization achieved by our approach. The x-axis shows the problem setting, and the y-axis shows the generalization factor (ration of maximum number of test objects to the maximum number of training objects). The dotted red-lines shows the expected generalization of state-of-the-art behavior cloning approaches. (b) shows the robustness of our approach along with the comparison with Code-as-policies. The x-axis shows the amount of training demonstrations used to learn the world model, and the y-axis shows the fraction of test tasks that our approach can solve. (c) shows the succinctness of the learned world models. The x-axis shows the fraction of the total training demonstrations used (similar to (b)), the y-axis shows the number of high-level actions invented in the world model. The shaded region shows the training actions used to solve the test tasks.

tings, respectively, while the training tasks contained at most two objects, and i.e., the generalization factor was **20** for the simulated setting and **4** for the real-world experiments. Similarly, for the Keva and Jenga settings, we learned the world model in settings with 3 plans and successfully scaled to building structures with 30 Keva planks in simulated settings (generalization factor of 10) using the YuMi robot and 18 Jenga planks in real-world settings (generalization factor of 6) using the Fetch robot. An important note is that we only scale up to 4x objects in the box packing setting, as the box can only accommodate 4 cans, and the training environment included only a single can.

Sample efficiency and robustness The next area where current methods are limited is sample efficiency—the number of training demonstrations needed to learn a policy. Many approaches require a substantial amount of training data, often tens of thousands of examples for each task (Brohan et al. 2022; Zhao et al. 2024). In contrast, our method demonstrates exceptional sample efficiency, requiring a maximum of only 200 training demonstrations to learn generalizable world models. These demonstrations consist of an equal mix of successful (50%) and failed (50%) attempts across all training tasks. To further evaluate our approach, we reduce the number of training demonstrations used to learn the world models and test them on a complex set of tasks, assessing the overall system's robustness. Figure 5(b) illustrates that our method can learn useful world models using as few as 40 training demonstrations.

Performance Foundation models (OpenAI 2024; Touvron et al. 2023) have recently emerged as a popular option for creating implicit abstractions. Several methods have attempted to take advantage of the information retrieval capabilities of foundation models to address long-horizon robot planning challenges (Rana et al. 2023; Yu et al. 2023; Liang et al. 2023; Driess et al. 2023; Wu et al. 2023; Han et al. 2024). However, foundation models often struggle to recognize robot- and problem-specific requirements for these abstractions and fail to handle complex tasks effectively. Additionally, it has been shown that foundation models possess limited reasoning abilities for long-horizon planning (Valmeekam et al. 2023a,b; Kambhampati et al. 2024). In our empirical study, we compared two state-of-the-art approaches using the same test tasks: (i) a stochastic task and motion planner (Shah et al. 2020) that employs expert-crafted world models, and (ii) Code as Policies (CoP) (Liang et al. 2023), which utilizes large language models (LLMs) trained on internet-scale data. As expected, STAMP, with its expert-crafted world models, was able to solve all test tasks. In contrast, CoP was only able to solve very simple tasks that involved at most XX objects. (Fig. 5(b)). It is important to note that we had to modify CoP to re-attempt tasks when it initially failed, creating a feedback loop. Additionally, we also provided expert-crafted high-level actions as input in the form of Python interfaces that CoP can use to execute different high-level robot actions—a core component of the world model that our approach automatically learns.

Succinctness Different approaches have sought to learn various components of world models. However, many of these approaches learn a significant number of relations and actions that are not useful for solving test tasks. We also assess the quality of the automatically learned world models through our approach in terms of succinctness, which refers to the number of invented relations and actions, as well as the number of those relations and actions utilized for solving the test tasks. Figure 5(c) illustrates the succinctness of

the automatically learned world models. In most cases, the learned world model consists of only a few actions and relations, with almost 90% of these learned actions and relations being employed to solve the test tasks. This demonstrates the reusability and transferability of the learned world models.

Interpretability One of the main advantages of our approach is that the learned relationships and high-level actions are semantically interpretable and intuitive for humans. Figure 3 illustrates a few automatically learned relationships by our method, which closely resemble human-understandable concepts, such as `On(Object, Table)`, `Near(Robot, Table)`, and `Parallel(Plank1, Plank2)`. These relationships not only aid humans in understanding and verifying the automatically learned world models but also facilitate easier specification and communication of tasks to robots.

6 Related Work

The presented approach directly relates to various concepts in task and motion planning, model learning, and abstraction learning. However, to the best of our knowledge, this is the first work that automatically invents generalizable symbolic predicates and high-level actions simultaneously using a set of low-level trajectories.

Task and motion planning approaches (Srivastava et al. 2014; Dantam et al. 2018; Garrett, Lozano-Pérez, and Kaelbling 2020; Shah et al. 2020) develop approaches for autonomously solving long-horizon robot planning problems. These approaches are complementary to the presented approach as they focus on using provided abstractions for efficiently solving the robot planning problems. Shah and Srivastava (2022, 2024) learn state and action abstractions for long-horizon motion planning problems. Orthogonal research (Mishra et al. 2023; Cheng et al. 2023; Fang et al. 2023) learn implicit abstractions (action interpreters or abstract actions) for TAMP in the form of generative models. However, these approaches do not learn generalizable relational representations as well as complex high-level relations and actions which is the focus of our work.

Several approaches invent symbolic vocabularies given a set of high-level actions (or skills) (Konidaris, Kaelbling, and Lozano-Pérez 2014; Ugur and Piater 2015; Konidaris, Kaelbling, and Lozano-Pérez 2015; Andersen and Konidaris 2017; Konidaris, Kaelbling, and Lozano-Pérez 2018; Bonet and Geffner 2019; James, Rosman, and Konidaris 2020). Ahmetoglu et al. (2022); Asai et al. (2022); Liang and Boularias (2023) learn symbolic predicates in the form of latent spaces of deep neural networks and use them for high-level symbolic planning. However, these approaches assume high-level actions to be provided as input. On the other hand, the approach presented in this paper automatically learns high-level actions along with symbolic predicates.

Numerous approaches (Yang, Wu, and Jiang 2007; Cresswell, McCluskey, and West 2009; Zhuo and Kambhampati 2013; Aineto, Celorrio, and Onaindia 2019; Verma, Marpally, and Srivastava 2021) have focused on learning preconditions and effects for high-level actions, i.e., action model. A few approaches (Čertický 2014; Lamanna et al. 2021) have also focused on continually learning action models while collecting experience in the environment. Bryce, Benton, and Boldt (2016) and Nayyar, Verma, and Srivastava (2022) focus on updating a known model using inconsistent observations. However, these approaches require a set of symbolic predicates and/or high-level action signatures as input whereas our approach automatically invents these predicates and actions. Several approaches (Silver et al. 2021; Verma, Marpally, and Srivastava 2022; Chitnis et al. 2022; Silver et al. 2022; Kumar et al. 2023; Silver et al. 2023) have been able to automatically invent high-level actions that are induced by state abstraction akin to the presented approach. However, unlike our approach, these approaches do not automatically learn symbolic predicates and/or low-level samplers and require them as input.

LLMs for robot planning Recent years have also seen significantly increased interest in using foundational models such as LLM (large language model), VLM (visual language model), and transformers for robot planning and control owing to their success in other fields such as NLP, text generation, and vision. Several approaches (Brohan et al. 2022; Goyal et al. 2023; Shridhar, Manuelli, and Fox 2023; Vuong et al. 2023) use transformer architecture for learning reactive policies for short-horizon robot control problems. Problems tackled by these approaches are analogous to individual actions learned by our approach.

Several directions of research explore the use of LLMs for utilize LLMs as high-level planners to generate sequences comprising of high-level, expert crafted actions (Yu et al. 2023; Liang et al. 2023; Huang et al. 2022; Rana et al. 2023; Lin et al. 2023; Huang et al. 2023b; Ahn et al. 2023). These methods make progress on the problem of near-natural language communication with robots and are complementary to the proposed work. However, there is a strong evidence against the soundness of LLMs as planners. Valmeekam et al. (2023a) show that LLMs are only $\sim 36\%$ accurate as planners even in simple block stacking settings not involving more than 5 object.

On the other hand, approaches that utilize LLMs to translate user requirements to formal specifications (Yu et al. 2023; Ding et al. 2023; Liu et al. 2023b,a; Kwon et al. 2023; Huang et al. 2023a) are complimentary to our approach. These approaches input a set of symbolic predicates and use LLMs for automatically generating symbolic goals from natural language specifications. These goals can be further used by existing planners.

7 Conclusion

This paper presents the first known approach for using continuous low-level demonstration to invent symbolic state and action abstractions that generalize to different robots and unseen problem settings. Thorough evaluation in simulated and real-world settings shows that the learned abstractions are efficient and sound, as well as generate comprehensible abstractions. In the future, we aim to utilize these automatically learned abstractions to allow non-experts to operate robots. We also aim to extend our approach to account for stochasticity in the environment while providing strong theoretical guarantees on learned world models.

References

Ahmetoglu, A.; Seker, M. Y.; Piater, J.; Oztop, E.; and Ugur, E. 2022. DeepSym: Deep symbol generation and rule learning for planning from unsupervised robot interaction. *JAIR*, 75: 709–745.

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Ho, D.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jang, E.; Ruano, R. J.; Jeffrey, K.; Jesmonth, S.; Joshi, N. J.; Julian, R.; Kalashnikov, D.; Kuang, Y.; Lee, K.-H.; Levine, S.; Lu, Y.; Luu, L.; Parada, C.; Pastor, P.; Quiambao, J.; Rao, K.; Rettinghouse, J.; Reyes, D.; Sermanet, P.; Sievers, N.; Tan, C.; Toshev, A.; Vanhoucke, V.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; Yan, M.; and Zeng, A. 2023. Do as I can, not as I say: Grounding language in robotic affordances. In *Proc. CoRL*.

Aineto, D.; Celorrio, S. J.; and Onaindia, E. 2019. Learning Action Models With Minimal Observability. *AIJ*, 275: 104–137.

Andersen, G.; and Konidaris, G. 2017. Active exploration for learning symbolic representations. In *Proc. NeurIPS*.

Asai, M.; Kajino, H.; Fukunaga, A.; and Muise, C. 2022. Classical planning in deep latent space. *JAIR*, 74: 1599–1686.

Black, K.; Brown, N.; Driess, D.; Esmail, A.; Equi, M.; Finn, C.; Fusai, N.; Groom, L.; Hausman, K.; Ichter, B.; et al. 2024. pi_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*.

Bonet, B.; and Geffner, H. 2019. Learning first-order symbolic representations for planning from the structure of the state space. In *Proc. ECAI*.

Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Dabis, J.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jackson, T.; Jesmonth, S.; Joshi, N. J.; Julian, R.; Kalashnikov, D.; Kuang, Y.; Leal, I.; Lee, K.-H.; Levine, S.; Lu, Y.; Malla, U.; Manjunath, D.; Mordatch, I.; Nachum, O.; Parada, C.; Peralta, J.; Perez, E.; Pertsch, K.; Quiambao, J.; Rao, K.; Ryoo, M.; Salazar, G.; Sanketi, P.; Sayed, K.; Singh, J.; Sontakke, S.; Stone, A.; Tan, C.; Tran, H.; Vanhoucke, V.; Vega, S.; Vuong, Q.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; Yu, T.; and Zitkovich, B. 2022. RT-1: Robotics Transformer for Real-World Control at Scale. *arXiv:2212.06817*.

Bryce, D.; Benton, J.; and Boldt, M. W. 2016. Maintaining Evolving Domain Models. In *Proc. IJCAI*.

Čertický, M. 2014. Real-Time Action Model Learning with Online Algorithm 3SG. *Applied AI*, 28(7): 690–711.

Cheng, S.; Garrett, C.; Mandlekar, A.; and Xu, D. 2023. NOD-TAMP: Multi-Step Manipulation Planning with Neural Object Descriptors. In *CoRL 2023 LEAP Workshop*.

Chitnis, R.; Silver, T.; Tenenbaum, J. B.; Lozano-Pérez, T.; and Kaelbling, L. P. 2022. Learning neuro-symbolic relational transition models for bilevel planning. In *Proc. IROS*.

Cresswell, S.; McCluskey, T.; and West, M. 2009. Acquisition of Object-Centred Domain Models from Planning Examples. In *Proc. ICAPS*.

Dantam, N. T.; Kingston, Z. K.; Chaudhuri, S.; and Kavraki, L. E. 2018. An incremental constraint-based framework for task and motion planning. *IJRR*, 37(10): 1134–1151.

Ding, Y.; Zhang, X.; Paxton, C.; and Zhang, S. 2023. Task and motion planning with large language models for object rearrangement. In *Proc. IROS*.

Driess, D.; Xia, F.; Sajjadi, M. S. M.; Lynch, C.; Chowdhery, A.; Ichter, B.; Wahid, A.; Tompson, J.; Vuong, Q.; Yu, T.; Huang, W.; Chebotar, Y.; Sermanet, P.; Duckworth, D.; Levine, S.; Vanhoucke, V.; Hausman, K.; Toussaint, M.; Greff, K.; Zeng, A.; Mordatch, I.; and Florence, P. 2023. PaLM-E: An Embodied Multimodal Language Model. In *Proc. ICML*.

Fang, X.; Garrett, C. R.; Eppner, C.; Lozano-Pérez, T.; Kaelbling, L. P.; and Fox, D. 2023. DiMSam: Diffusion Models as Samplers for Task and Motion Planning under Partial Observability. In *CoRL 2023 LEAP Workshop*.

Fu, Z.; Zhao, T. Z.; and Finn, C. 2024. Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation. In *Conference on Robot Learning (CoRL)*.

Garrett, C. R.; Lozano-Pérez, T.; and Kaelbling, L. P. 2020. PDDLStream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning. In *Proc. ICAPS*.

Goyal, A.; Xu, J.; Guo, Y.; Blukis, V.; Chao, Y.-W.; and Fox, D. 2023. RVT: Robotic View Transformer for 3D Object Manipulation. In *Proc. CoRL*.

Hafner, D.; Pasukonis, J.; Ba, J.; and Lillicrap, T. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Han, M.; Zhu, Y.; Zhu, S.-C.; Wu, Y. N.; and Zhu, Y. 2024. InterPreT: Interactive Predicate Learning from Language Feedback for Generalizable Task Planning. In *Proc. R:SS*.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proc. ICML*.

Huang, W.; Wang, C.; Zhang, R.; Li, Y.; Wu, J.; and Fei-Fei, L. 2023a. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *Proc. CoRL*.

Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y.; Sermanet, P.; Brown, N.; Jackson, T.; Luu, L.; Levine, S.; Hausman, K.; and Ichter, B. 2023b. Inner Monologue: Embodied Reasoning through Planning with Language Models. In *Proc. CoRL*.

James, S.; Rosman, B.; and Konidaris, G. 2020. Learning portable representations for high-level planning. In *Proc. ICML*.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L. P.; and Murthy, A. B. 2024. Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In *Proc. ICML*.

Konidaris, G.; Kaelbling, L. P.; and Lozano-Pérez, T. 2014. Constructing symbolic representations for high-level planning. In *Proc. AAAI*.

Konidaris, G.; Kaelbling, L. P.; and Lozano-Pérez, T. 2015. Symbol acquisition for probabilistic high-level planning. In *Proc. IJCAI*.

Konidaris, G.; Kaelbling, L. P.; and Lozano-Pérez, T. 2018. From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *JAIR*, 61: 215–289.

Kumar, N.; McClinton, W.; Chitnis, R.; Silver, T.; Lozano-Pérez, T.; and Kaelbling, L. P. 2023. Learning Efficient Abstract Planning Models that Choose What to Predict. In *Proc. CoRL*.

Kwon, M.; Xie, S. M.; Bullard, K.; and Sadigh, D. 2023. Reward design with language models. In *Proc. ICLR*.

Lamanna, L.; Saetti, A.; Serafini, L.; Gerevini, A.; and Traverso, P. 2021. Online Learning of Action Models for PDDL Planning. In *Proc. IJCAI*.

LaValle, S. M. 2006. *Planning Algorithms*. USA: Cambridge University Press. ISBN 0521862051.

Liang, J.; and Boularias, A. 2023. Learning category-level manipulation tasks from point clouds with dynamic graph CNNs. In *Proc. ICRA*.

Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; and Zeng, A. 2023. Code as policies: Language model programs for embodied control. In *Proc. ICRA*.

Lin, K.; Agia, C.; Migimatsu, T.; Pavone, M.; and Bohg, J. 2023. Text2Motion: From natural language instructions to feasible plans. *Autonomous Robots*.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023a. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv:2304.11477*.

Liu, W.; Du, Y.; Hermans, T.; Chernova, S.; and Paxton, C. 2023b. StructDiffusion: Language-guided creation of physically-valid structures using unseen objects. In *Proc. RSS*.

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

Mishra, U. A.; Xue, S.; Chen, Y.; and Xu, D. 2023. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Proc. CoRL*.

Molina, D.; Kumar, K.; and Srivastava, S. 2020. Learn and Link: Learning Critical Regions for Efficient Planning. In *Proc. ICRA*.

Nayyar, R. K.; Verma, P.; and Srivastava, S. 2022. Differential Assessment of Black-Box AI Agents. In *Proc. AAAI*.

OpenAI. 2024. ChatGPT.

Rana, K.; Haviland, J.; Garg, S.; Abou-Chakra, J.; Reid, I.; and Suenderhauf, N. 2023. SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning. In *Proc. CoRL*.

Shah, N.; and Srivastava, S. 2022. Using Deep Learning to Bootstrap Abstractions for Hierarchical Robot Planning. In *Proc. AAMAS*.

Shah, N.; and Srivastava, S. 2024. Hierarchical Planning and Learning for Robots in Stochastic Settings Using Zero-Shot Option Invention. In *Proc. AAAI*.

Shah, N.; Vasudevan, D. K.; Kumar, K.; Kamojjhala, P.; and Srivastava, S. 2020. Anytime Integrated Task and Motion Policies for Stochastic Environments. In *Proc. ICRA*.

Shridhar, M.; Manuelli, L.; and Fox, D. 2023. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proc. CoRL*.

Silver, T.; Athalye, A.; Tenenbaum, J. B.; Lozano-Pérez, T.; and Kaelbling, L. P. 2022. Learning Neuro-Symbolic Skills for Bilevel Planning. In *Proc. CoRL*.

Silver, T.; Chitnis, R.; Kumar, N.; McClinton, W.; Lozano-Pérez, T.; Kaelbling, L. P.; and Tenenbaum, J. 2023. Predicate Invention for Bilevel Planning. In *Proc. AAAI*.

Silver, T.; Chitnis, R.; Tenenbaum, J.; Kaelbling, L. P.; and Lozano-Pérez, T. 2021. Learning Symbolic Operators for Task and Motion Planning. In *Proc. IROS*.

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S.; and Abbeel, P. 2014. Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer. In *Proc. ICRA*.

Stern, R.; and Juba, B. 2017. Efficient, Safe, and Probably Approximately Complete Learning of Action Models. In *Proc. IJCAI*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Couairon, R.; Moreau, T.; Pino, J.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. https://ai.facebook.com/research/large-language-models.

Ugur, E.; and Piater, J. 2015. Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In *Proc. ICRA*.

Valmeekam, K.; Marquez, M.; Sreedharan, S.; and Kambhampati, S. 2023a. On the Planning Abilities of Large Language Models–A Critical Investigation. In *Proc. NeurIPS*.

Valmeekam, K.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2023b. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). In *Proc. NeurIPS*.

Verma, P.; Marpally, S. R.; and Srivastava, S. 2021. Asking the Right Questions: Learning Interpretable Action Models Through Query Answering. In *Proc. AAAI*.

Verma, P.; Marpally, S. R.; and Srivastava, S. 2022. Discovering User-Interpretable Capabilities of Black-Box Planning Agents. In *Proc. KR*.

Vuong, Q.; Levine, S.; Walke, H. R.; Pertsch, K.; Singh, A.; Doshi, R.; Xu, C.; Luo, J.; Tan, L.; Shah, D.; Finn, C.; Du, M.; Kim, M. J.; Khazatsky, A.; Yang, J. H.; Zhao, T. Z.; Goldberg, K.; et al. 2023. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. In *CoRL 2023 TGR Workshop*.

Wang, X. 1994. Learning Planning Operators by Observation and Practice. In *Proc. AIPS*.

Wu, J.; Antonova, R.; Kan, A.; Lepert, M.; Zeng, A.; Song, S.; Bohg, J.; Rusinkiewicz, S.; and Funkhouser, T. 2023. TidyBot: personalized robot assistance with large language models. *Autonomous Robots*, 47(8): 1087–1102.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning Action Models from Plan Examples Using Weighted MAX-SAT. *AIJ*, 171(2-3): 107–143.

Yu, W.; Gileadi, N.; Fu, C.; Kirmani, S.; Lee, K.-H.; Arenas, M. G.; Chiang, H.-T. L.; Erez, T.; Hasenclever, L.; Humplik, J.; Ichter, B.; Xiao, T.; Xu, P.; Zeng, A.; Zhang, T.; Heess, N.; Sadigh, D.; Tan, J.; Tassa, Y.; and Xia, F. 2023. Language to Rewards for Robotic Skill Synthesis. In *Proc. CoRL*.

Zhao, T. Z.; Tompson, J.; Driess, D.; Florence, P.; Ghasemipour, S. K. S.; Finn, C.; and Wahid, A. 2024. ALOHA Unleashed: A Simple Recipe for Robot Dexterity. In *Proc. CoRL*.

Zhuo, H. H.; and Kambhampati, S. 2013. Action-model acquisition from noisy plan traces. In *Proc. IJCAI*.