

000 001 002 003 004 005 EXOPREDICATOR: LEARNING ABSTRACT MODELS OF 006 DYNAMIC WORLDS FOR ROBOT PLANNING 007 008 009

010 **Anonymous authors**
011 Paper under double-blind review
012
013
014
015
016
017
018
019
020

021 ABSTRACT 022

023 Long-horizon embodied planning is challenging because the world does not only
024 change through an agent’s actions: exogenous processes (e.g., water heating,
025 dominoes cascading) unfold concurrently with the agent’s actions. We propose
026 a framework for abstract world models that jointly learns (i) symbolic state rep-
027 resentations and (ii) causal processes for both endogenous actions and exogenous
028 mechanisms. Each causal process models the time course of a stochastic cause-
029 effect relation. We learn these world models from limited data via variational
030 Bayesian inference combined with LLM proposals. Across five simulated tabletop
031 robotics environments, the learned models enable fast planning that generalizes to
032 held-out tasks with more objects and more complex goals, outperforming a range
033 of baselines.
034

035 1 INTRODUCTION 036

037 For an agent to think about the future consequences of its actions, does it need to simulate the world
038 pixel-by-pixel, frame-by-frame, or can it reason more abstractly? Consider planning a flight to an-
039 other country: we can reason about buying tickets, changing airplanes, and crossing borders without
040 committing to the color of the airplane or the milliseconds before takeoff. Absent abstraction, plan-
041 ning over long time horizons would be intractable, because every minute detail of the world would
042 need to be simulated. This intuition is captured by *abstract world models*, (Konidaris, 2019; Wong
043 et al., 2025) which retain information essential for decision-making, while hiding irrelevant details.
044

045 Recent work on learning abstract world models (e.g., (Liang et al., 2024; Athalye et al., 2024))
046 assumes that the world changes only by direct, instantaneous actions. But in the real world, our
047 actions are not instantaneous, and are only half the story: the external world has its own causal
048 mechanisms, which unfold continuously in time concurrent with our own actions. For instance,
049 consider boiling water (Figure 1). After switching on a kettle, the water’s temperature continuously
050 rises independently of the agent’s subsequent actions until it finally boils. A *good abstract world*
051 *model must therefore abstract not just the states, but also the temporal granularity*: decision-makers
052 should know that switching on a kettle triggers another causal mechanism, which eventually results
053 in boiling, without reasoning about the exact timecourse of the water’s temperature (i.e., a robot
054 could chop vegetables while waiting for the water to boil). Such abstraction is conceptually separate
055 from options/skills/high-level actions, which abstract the timecourse of one’s own actions, but do
056 not abstract the timecourse of external causal processes in the outside world.
057

058 The standard planning representation used for decades, PDDL, also fails to capture this: it only mod-
059 els the effects of one’s own actions, and treats each action as instantaneous. Learning a symbolic
060 PDDL planning model (Silver et al., 2023; Liang et al., 2024) combinatorially explodes with respect
061 to temporal granularity. Vision-language models (VLMs) and vision-language-action models (Team
062 et al., 2023; Black et al., 2025) could in principle reason about external causal mechanisms, but gen-
063 eralize poorly to novel situations, particularly when reasoning about temporal physical constraints.
064

065 To address these challenges, we introduce a framework for learning world models that abstract both
066 the state space, and the timecourse of causal processes. We contribute the following: (1) A symbolic
067 yet learnable representation of abstract world models for environments with temporal dynamics and
068 external causal processes. (2) A state abstraction learner that leverages the commonsense knowledge
069 of foundation models. (3) An efficient Bayesian inference method for learning the parameters and
070 structures of these causal models. (4) A fast planner for reasoning with the proposed representation.
071

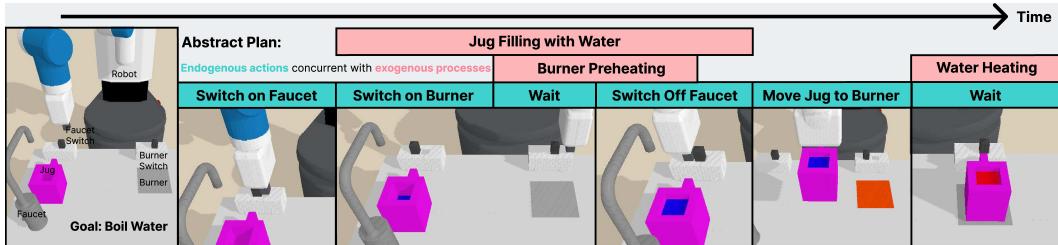


Figure 1: *Dynamic* environments include both **endogenous processes** (actions under the agent’s direct control, such as **Switch On Faucet**) and **exogenous processes** (e.g., **Jug Filling with Water**) that evolve on their own. Planning requires reasoning about both kinds of processes.

2 BACKGROUND AND PROBLEM FORMULATION

We consider learning abstract world models for robot planning in environments whose causal mechanisms include both the agent’s own action space, and external mechanisms not directly under the agent’s control. The actual environment operates frame-by-frame (high temporal granularity), and exposes a state space with object tracking features and pixel-level visual appearance (high-resolution perception). We assume built-in motor skills, such as `Pick/Place`, a common assumption (Kumar et al., 2024; Silver et al., 2021). The goal is to learn a world model abstractly describing the timecourse of causal processes, and to generalize to held-out decision-making tasks.

Environments. An environment \mathcal{E} is a tuple $\langle \mathcal{X}, \mathcal{U}, \mathcal{C}, f, \Lambda \rangle$ where \mathcal{X} is a state space, $\mathcal{U} \subseteq \mathbb{R}^m$ is a low-level action space (e.g. motor torques), \mathcal{C} is a set of controllers for skills (e.g. `Pick/Place`), $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$ is a transition function, and Λ is a set of *object types* (object classifier outputs).

Tasks. Within an environment, a task T is a tuple $\langle \mathcal{O}, x_0, g \rangle$ of objects \mathcal{O} , initial state x_0 , and goal g . The allowed states depend on the objects \mathcal{O} , so we write the state space as $\mathcal{X}_{\mathcal{O}}$ (or sometimes just \mathcal{X} when the objects are clear from context). Each state $x \in \mathcal{X}_{\mathcal{O}}$ includes associated object features, such as 3D object position. The environment is shared across tasks.

3 ABSTRACTING STATES, TIME, AND CAUSAL PROCESSES

Environments present a high-dimensional observation space that evolves frame-by-frame. Abstract world models hide this complexity behind a *state abstraction*, which distills a small set of features from observations, together with *causal processes*, which describe temporal dynamics (Figure 2).

State Abstraction with Predicates. A predicate ψ – after being parameterized by specific objects – is a Boolean feature of states. A predicate is represented as a Python function that queries a VLM to inspect the robot’s visual input. We treat this as function $\psi : \mathcal{O}^m \rightarrow (\mathcal{X} \rightarrow \mathbb{B})$ that given m objects predicts whether a predicate holds in a state. A set of predicates Ψ induces an *abstract state* s corresponding to all the predicate/object combinations (*ground atoms*) that hold in that state:

$$s := \text{ABSTRACT}_{\Psi}(x) = \{(\psi, o_1, \dots, o_m) : \psi(o_1, \dots, o_m) \text{ holds in state } x, \text{ for } \psi \in \Psi \text{ and } o_j \in \mathcal{O}\}$$

Thus an abstract state is the collection of symbolic facts the agent believes about the world. This style of state abstraction is standard in robot task planning (Garrett et al., 2021). We write \mathcal{S} for the set of possible abstract states.

Causal Processes: Informal intuition. A causal process coarsely models a cause-effect relation. For example, opening a faucet above a bucket triggers a chain of cause-effect relations: a stopper opens, hidden pipes fill with water, water rises in the bucket, and eventually the bucket is filled. For decision making, many such details are irrelevant, or omitted from the state abstraction. A causal process therefore abstracts away such details by saying that certain conditions (the “cause”) later lead to other conditions (the “effect”). We further distinguish two kinds of causal processes. **Endogenous processes** correspond to the agent’s high-level actions or skills. They represent operations that are under the agent’s direct control, such as switching a faucet on. **Exogenous processes** describe the background dynamics of the environment. They represent processes that unfold autonomously once certain conditions hold, independent of whether the conditions were created by the agent or by some

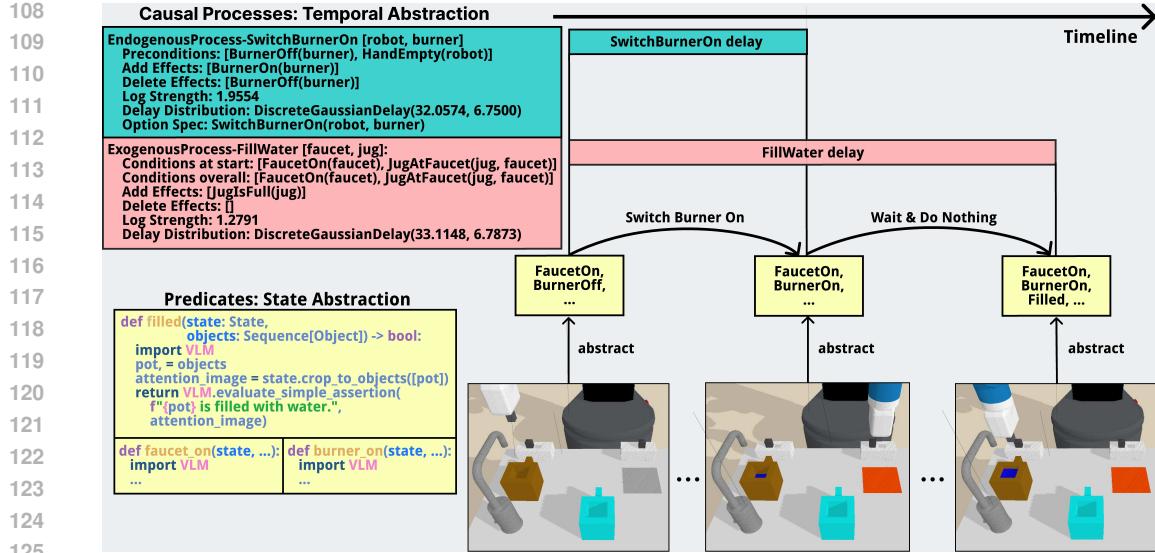


Figure 2: Raw input maps to a state abstraction via *predicates*: short Python programs detecting binary features. Learning the state abstraction means synthesizing these programs. Temporal dynamics of abstract states are governed by *causal processes*: either *endogenous processes* (actions), or *exogenous processes* in the outside world. Causes realize their effects only after a delay, and can be interleaved. Learning causal processes allows planning by breaking frame-by-frame dynamics into discrete jumps between abstract states. (Illustration simplified; see text.)

external mechanism, such as a kettle filling with water after being placed under a running tap. This separation allows the agent to reason about the consequences of its actions while also anticipating changes initiated by the environment itself. Figure 2 (top right) shows concrete examples of such causal processes (e.g., switching a burner on, filling a jug), with their conditions, delayed effects, and whether they are endogenous or exogenous.¹

Causal Processes: Formalization. Abstract world models are equipped with a set of causal processes \mathcal{L} . Each causal process $L \in \mathcal{L}$ is defined by a **schema** tuple, $\langle \text{PAR}, C, O, E, W, p^{\text{delay}} \rangle$ (see fig. 2, top right, for concrete schema examples), where:

- **PAR (parameters)** is a list of typed variables present in the condition or effect of the process.
- C is the *condition at start*, a set of atoms that must be true for the process to be activated.
- O is the *condition overall*, a set of atoms that must remain true throughout the process’s duration.
- E is the *effect*, an add and a delete set of atoms describing the state change upon completion.
- $W \in \mathbb{R}$ (*log strength*) quantifies how likely the effect will happen when the conditions are satisfied.
- p^{delay} is a probability distribution over the delay between the process’s activation and effect (Appendix C.2).

Endogenous processes further include a skill $c \in \mathcal{C}$. A schema is instantiated into a **ground causal process** $\underline{L} = \langle \text{PAR}, \underline{C}, \underline{O}, \underline{E}, W, p^{\text{delay}} \rangle$ (and optionally c) by substituting its parameters with specific objects. For example, the endogenous process *SwitchBurnerOn* has parameters ?robot and ?burner . Its condition C requires that the burner is off and the robot’s hand is free. Once executed, its effect E (the burner being on) occurs after a delay sampled from p^{delay} .

¹Our terminology differs from standard definitions in causality (Pearl, 2009) and recent RL literature (e.g., Efroni et al. (2021)), where “exogenous” typically refers to variables that are determined outside the system or are irrelevant to the task. In contrast, our exogenous processes are task-relevant and can be triggered by the agent (and are thus causally downstream), but differ in that they evolve autonomously without requiring continuous agent actuation. This aligns with the definition of “processes” in temporal planning frameworks like PDDL+ (Fox & Long, 2002)

Interdependence of state abstraction and causal processes. The right abstractions depend on downstream tasks, as our goal is generalization to unseen planning problems. Tasks demanding a more detailed state abstraction will generally require modeling more temporal dynamics. This couples the causal processes to the state abstraction.

Probabilistic Semantics. The causal processes define how abstract states evolve over time. Mathematically, \mathcal{L} defines a probabilistic generative model over sequences of abstract states $\{s_t\}$ at fine-grained timesteps $t \in \mathbb{N}$. At each step, active causal processes contribute pressure for their effect atoms to change, while a separate frame term prefers all atoms to remain as they are. To write this compactly, we use indicator functions for the *conditions at start*: $C_{\underline{L}}(s_{1:t})$ is 1 iff all atoms in the condition-at-start set C of ground process \underline{L} are satisfied at state s_t but not satisfied at state s_{t-1} , and 0 otherwise; similarly, $O_{\underline{L}}(s_{t'+1:t-1})$ is 1 iff all atoms in the overall-condition set O hold at every step between $t' + 1$ and $t - 1$. We probabilistically model the effect of ground process \underline{L} as a potential $E_{\underline{L}}^j$ upon each feature j of the abstract state:

$$\log E_{\underline{L}}^j(s^j) = \begin{cases} W_{\underline{L}} \times s^j & \text{if } j \in E_{\underline{L}}.\text{Add} \\ W_{\underline{L}} \times (1 - s^j) & \text{if } j \in E_{\underline{L}}.\text{Del} \\ 0 & \text{otherwise.} \end{cases}$$

We similarly define a *frame axiom potential* $\log F(s_t^j | s_{t-1}^j) = W_F \times 1 \left[s_t^j = s_{t-1}^j \right]$, which encourages states to stay constant over time; W_F is a learnable parameter. Because causal processes have stochastic delays, we associate each ground causal process \underline{L} with random variables $\{\Delta_t^{\underline{L}}\}$ for the delay should the process trigger at time t , i.e. $\Delta_t^{\underline{L}} \sim p_{\underline{L}}^{\text{delay}}$. With these definitions in hand, the joint distribution over delays and abstract states is

$$p(s_{1:t}, \{\Delta_{1:t}^L\} \mid s_0) = \prod_t p(s_t \mid s_{<t}, \{\Delta_{<t}^L\}) \prod_L p(\Delta_t^L \mid s_{\leq t}) \quad (\text{autoregressive})$$

$$p\left(s_t \mid s_{<t}, \{\Delta_{<t}^L\}\right) = \prod_j p\left(s_t^j \mid s_{<t}, \{\Delta_{<t}^L\}\right) \quad (\text{next-state factorizes over features})$$

$$p\left(s_t^j \mid s_{<t}, \{\Delta_{<t}^L\}\right) \propto F(s_t^j | s_{t-1}^j) \prod_{\substack{L \in \mathcal{L} \\ t' < t}} E_L^j(s_t^j)^{C_L(s_{1:t'}) O_L(s_{t'+1:t-1}) 1[t=t'+\Delta_{t'}^L]} \quad (\text{cause-effect})$$

$$p(\Delta_t^L | s_{<t}) = p_L^{\text{delay}}(\Delta_t^L | C_{\underline{L}}(s_{1:t})) \quad (\text{delay distribution for when condition at start holds})$$

But this formalization simulates every fine-grained timestep: reasoners should abstract away temporal details like the milliseconds before a domino falls. We describe next how to do that.

4 PLANNING WITH CAUSAL PROCESSES

We model the world as changing abruptly in discrete “jumps” between abstract states. Planning can clump together stretches of time where the abstract state remains constant. We define a **big-step transition function**, \mathcal{T}_{big} , which runs the world model until the abstract state changes, and optionally takes as input an action to initiate. The agent doesn’t always need to manipulate an object directly; it can also choose to wait for an exogenous process to unfold on its own (such as waiting for water to boil). The agent achieves this by initiating a special `NoOp` (no operation) action, which terminates as soon as the abstract state changes. Formalizing \mathcal{T}_{big} is relatively technical; see Appendix A.1.

This big-step function allows a planner to perform forward search in the space of high-level actions, simulating the concurrent and delayed effects of both its own actions and the environment's exogenous dynamics. Given causal processes \mathcal{L} and a task, the agent performs an A* search over sequences of ground endogenous processes. The search uses \mathcal{T}_{big} to determine successor states and we design a version of the fast-forward heuristic (Hoffmann, 2001) to guide the search (Appendix A.2).

5 PROCESS LEARNING AND PREDICATE INVENTION

Our goal is to learn how the outside world works: we assume an unfamiliar environment, but not an unfamiliar body. We therefore equip the agent with some basic predicates (such as whether

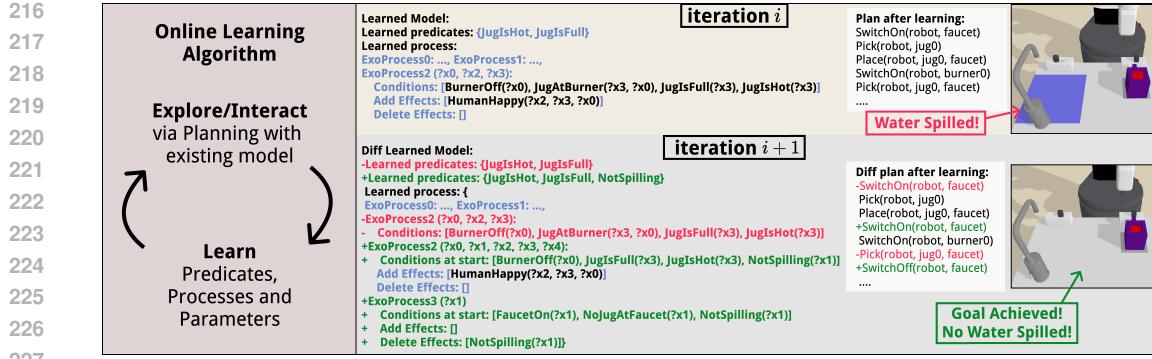


Figure 3: The online learning loop, where the agent repeatedly uses its current model to plan and interact with the world, then refines that model by learning new predicates and causal processes from the experience. The figure shows an example where the agent’s initial model in iteration i leads to a failed plan (Water Spilled!). After observing this failure and updating its knowledge (“Diff Learned Model”), the agent creates a successful plan in iteration $i + 1$ (“Diff plan after learning”).

it is holding an object) and endogenous causal processes defining its own action space, such as Pick/Place (Appendix B), and learn the remaining causal processes and state abstractions.

We initialize with 1-2 demonstration trajectories, and then perform online learning (Figure 3). Online learning involves planning to solve training tasks to collect further trajectories (when planning fails, random actions are taken). At each stage of online learning, we have a dataset of state-action trajectories $\mathcal{D}_{\text{low}} = \{(x_0, \mathcal{C}_0, \dots, \mathcal{C}_{T-1}, x_T)\}$. Given predicates Ψ , this generates a dataset of abstract state-action trajectories $\mathcal{D}_{\text{abs}} = \{(s_0, \mathcal{C}_0, \dots, \mathcal{C}_{T-1}, s_T)\}$. Given this dataset, we learn predicates (Section 5.3), exogenous processes (section 5.2), and continuous parameters (Section 5.1)—described in a reverse order, because each component is used in the previous component during learning.

5.1 PARAMETER LEARNING VIA VARIATIONAL INFERENCE

Each causal process has continuous parameters for the delay distribution p_L^{delay} and the probabilistic weight W_L , and we have a global parameter W_F for the frame axiom. Fixing all discrete structure—including the predicates—we seek parameters maximizing the marginal likelihood $p(\mathcal{D}_{\text{abs}}) = p(s_{1:t} | s_0) = \sum_{\{\Delta_{1:t}^L\}} p(s_{1:t}, \{\Delta_{1:t}^L\} | s_0)$. This is intractable because there are combinatorially many ways of timing the cause-effects.

We therefore approximate the marginal likelihood by introducing variational distributions over the time at which each process realizes its effect: categorical distribution $q_t^L(A_t^L)$ encodes belief about A_t^L , the “arrival time” of the effect coming from the process L due to its activation at time t . This is a change of basis from delays to absolute times, $A_t^L = \Delta_t^L + t$, which leads to a tidier ELBO decomposition. We provide the full expression and derivation of the following variational lower bound on the marginal likelihood in Appendix A.3, which we optimize using Adam (Kingma, 2014).

$$\log p(s_{1:t}) \geq \underbrace{\mathbb{E}_q [\log p(\{A_t^L\})]}_{\text{delay model}} + \underbrace{\mathbb{E}_q [\log p(\{s_t\} | \{A_t^L\})]}_{\text{abstract dynamics}} + \underbrace{\mathbb{H}_q [\{A_t^L\}]}_{\text{entropy regularizer}}$$

5.2 LEARNING EXOGENOUS PROCESSES: BAYESIAN MODEL SELECTION, LLM GUIDANCE

We learn causal processes by assuming a fixed set of predicates Ψ (which fixes the abstract state space \mathcal{S}_Ψ): we **segment** the trajectories into shorter clips, where each clip consists of a sequence of constant abstract states followed by a final state in which one or more atoms change value; **cluster** segments according to which features in the abstract state were changed; then **learn** one process per cluster by optimizing Bayesian criteria. To make optimization tractable, we use an LLM to propose different symbolic forms for processes and then score them with our Bayesian objective.

270 Given a set of trajectories \mathcal{D}_{abs} , we would ideally learn the causal processes \mathcal{L}^* maximizing
 271

$$272 \quad \mathcal{L}^* = \arg \max_{\mathcal{L}} p(\mathcal{L} | \mathcal{D}_{\text{abs}}) = \arg \max_{\mathcal{L}} p(\mathcal{D}_{\text{abs}} | \mathcal{L}) p(\mathcal{L}) \quad (\text{intractable})$$

273 where $p(\mathcal{L})$ is a minimum description length prior and $p(\mathcal{D}_{\text{abs}} | \mathcal{L})$ is approximated by Section 5.1.
 274 As this optimization is intractable, we learn a separate process L_C for each cluster \mathcal{C} :

$$275 \quad \mathcal{L}^* = \bigcup_{\mathcal{C} \in \text{CLUSTER}(\mathcal{D}_{\text{abs}})} \{L_C\} \quad \text{where } L_C = \arg \max_L p(\mathcal{C} | L) p(L) \quad (\text{still intractable})$$

276 But computing $\arg \max_L p(\mathcal{C} | L) p(L)$ means optimizing over combinatorially many discrete structures
 277 for L . To narrow down the discrete search, we prompt a language model with the cluster and
 278 ask it to propose a small number of candidate processes:

$$279 \quad \mathcal{L}^* = \bigcup_{\mathcal{C} \in \text{CLUSTER}(\mathcal{D}_{\text{abs}})} \{L_C\} \quad \text{where } L_C = \arg \max_{L \in \text{PROMPT}(\mathcal{C})} p(\mathcal{C} | L) p(L) \quad (\text{tractable})$$

280 Appendix A.4 fully specifies this algorithm, which builds on Chitnis et al. (2022).
 281

282 5.3 LEARNING STATE ABSTRACTIONS: PROGRAM SYNTHESIS AND LOCAL SEARCH

283 The abstract state space is defined by a collection of short Python programs (predicates) which check
 284 for an abstract feature within the raw perceptual input. Learning the state abstraction therefore means
 285 synthesizing that set of programs. One strategy for learning the predicates is to *propose* a large set
 286 of candidate predicates and then use discrete search to *select* a subset optimizing certain objectives
 287 (Silver et al., 2023). In our setting, we propose predicates by prompting an LLM with trajectories,
 288 and seek a subset of those predicates, Ψ^* , maximizing:

$$289 \quad \Psi^* = \arg \max_{\Psi \in \mathcal{P}(\text{PROMPT}(\mathcal{D}_{\text{low}}))} p(\mathcal{D}_{\text{low}} | \mathcal{L}^*(\Psi)) p(\mathcal{L}^*(\Psi)) p(\Psi)$$

290 where $\mathcal{P}(\cdot)$ is the powerset, $p(\Psi)$ is a prior favoring fewer predicates, and we have made explicit
 291 the dependence of \mathcal{L}^* upon the predicates Ψ (Section 5.2). But the above objective is intractable for
 292 two reasons, which we address as follows (Appendix A.5):
 293

- 300 • **Expensive outer loop:** The powerset is exponentially large. Rather than exhaustively enumerate
 301 every subset of predicates, we do a local *hill-climbing* search starting from $\Psi = \emptyset$ and greedily
 302 adding new predicates from the LLM (Silver et al., 2023).
- 303 • **Expensive inner loop:** Scoring a candidate subset of predicates is expensive, requiring variational
 304 inference and causal process structure learning (Sections 5.1 and 5.2). We therefore run
 305 structure learning and parameter estimation *only once*, *using all proposed predicates*, and cache
 306 the resulting processes and parameters for reuse when scoring different subsets of predicates.

307 6 EXPERIMENTS

308 We design our experiments to answer the following questions: **(Q1)** How does ExoPredicator per-
 309 form compared to state-of-the-art methods, including hierarchical reinforcement learning (HRL),
 310 VLM planning, and operator learning approaches, in terms of overall solve rate and sample effi-
 311 ciency? **(Q2)** How do the learned abstractions perform relative to manually engineered abstractions,
 312 and relative to the case where no learning is performed? **(Q3)** How useful are the Bayesian model
 313 selection and the LLM-guidance components in model learning?

314 **Experimental Setup.** We evaluate eight approaches across five simulated robotics environments,
 315 illustrated in Figure 4, using the PyBullet physics engine (Coumans & Bai, 2016). All results are
 316 averaged across three random seeds. For each seed, we train the agent with one or two training tasks,
 317 each of which includes one demonstration. We then evaluate their performance on 50 held-out test
 318 tasks, which include more objects and more complex goals. In each online learning iteration, the
 319 agent performs 8 rollouts in a training task with each rollout lasting a maximum of 300 timesteps.
 320

321 **Environments.** We describe the environments and their corresponding predefined closed-loop
 322 skills, which are shared across all approaches. All environments have a `NoOp` skill in addition
 323 to the ones listed for each environment. See Appendix B for more details.

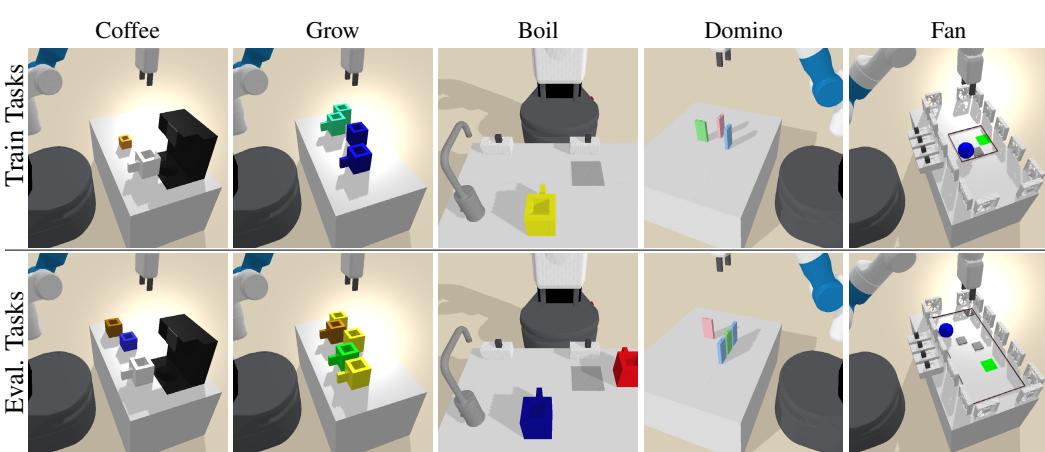


Figure 4: Environments. Top row: train task examples. Bottom row: evaluation task examples.

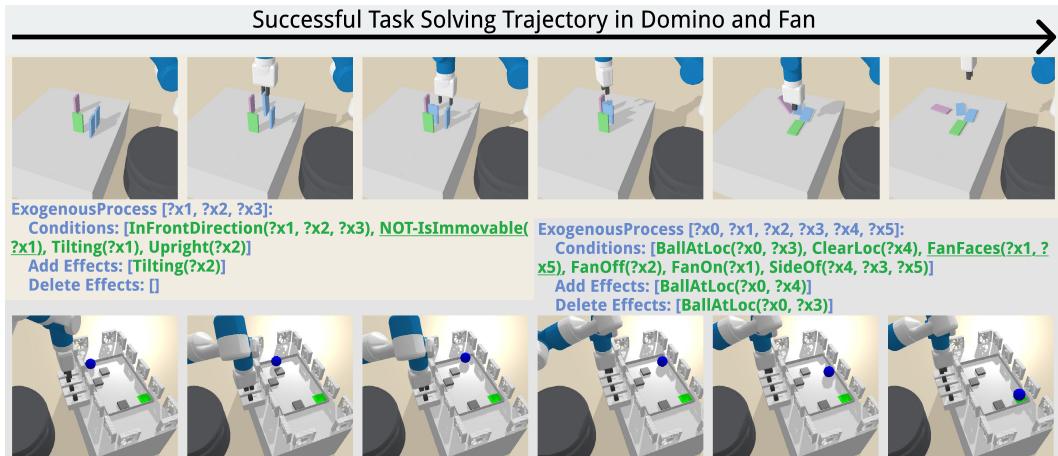


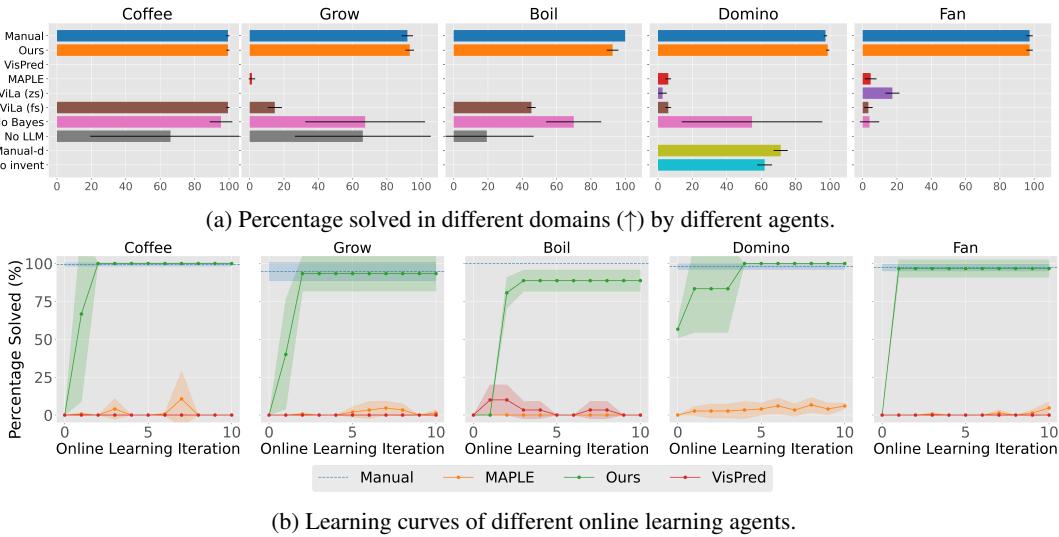
Figure 5: Successful ExoPredicator trajectories in the Domino (top) and Fan (bottom) environments. The code highlights the key learned exogenous processes, describing how dominoes cascade and how the fan’s wind moves the ball. These processes incorporate predicates invented by the agent, like NOT-IsImmovable and FanFaces, which enable efficient and effective planning.

1. **Coffee.** The agent is asked to fill the cups with coffee. To do so, the agent needs first to get coffee from the coffee machine, then pour it into cups, both of which are exogenous processes. The environment provides 4 skills: Pick, Place, Push, and Pour.
2. **Grow.** The agent is tasked with watering plants in pots. A plant will only grow when watered by a jug that has the same color as its pot. The provided skills include Pick, Place and Pour.
3. **Boil.** A cooking domain where the agent is asked to fill jugs with water using the faucet and boil it with the burner, without overspilling any water, which may happen when the faucet is on and no jug is under it or the jug underneath is full, which are all exogenous processes. Four skills are defined in this domain, including Pick, Place, and Switch On/Off.
4. **Domino.** A domino puzzle environment with two types of tasks. The agent is tasked to only move the blue dominoes and push the green dominoes such that all the purple target dominoes are toppled. Additionally, there are “impossible” tasks, where there are red target dominoes whose mass is too large to be toppled in a cascade. Impossible tasks are “solved” if the agent predicts that the goal is unachievable. Inter-domino dynamics are exogenous processes. The included skills are Pick, Place, and Push.

378 5. **Fan**. A maze environment where a ball is blown by fans. The agent must control the fans in each
 379 cardinal direction to move the ball to the green target location while avoiding obstacles. The
 380 provided skills are turning the `Switch On/Off`.
 381

382 Approaches.

383 1. **Manual**. A planning agent with manually engineered predicates and processes for each domain.
 384 2. **Ours**. Our ExoPredicator approach.
 385 3. **MAPLE** (Nasiriany et al., 2022). An HRL baseline that learns to select ground controllers by
 386 learning an action-value function, but does not explicitly learn abstract world models and perform
 387 lookahead planning. This approach is provided with 1000 training tasks and given a budget of
 388 10000 interaction rollouts per online learning iteration.
 389 4. **ViLA** (Hu et al., 2023). A VLM planning baseline that prompts a VLM (Gemini-2.5-Pro) to plan
 390 a sequence of ground skills. We experiment with two variants, which either exclude (zero-shot;
 391 zs) or include (few-shot; fs) the demonstrations. Both this and MAPLE are provided with the full
 392 set of predicates in **Manual**, which they can use as termination conditions for the `NoOp` action.
 393 5. **VisPred** (VisualPredicator) (Liang et al., 2024). An online STRIPS-style operator learning and
 394 planning agent. We provide it with all the necessary predicates used in **Manual**, which sidestep
 395 the challenge of predicate learning, to highlight the difference between our *causal processes*
 396 representation with traditional STRIPS-style representations.
 397 6. **No Bayes**. An ablation that uses an LLM to learn processes without Bayesian model selection.
 398 7. **No LLM**. An ablation that replaces the LLM condition proposer with a fast-to-compute heuristic.
 399 Note that we still use an LLM for predicate proposal.
 400 8. **Manual-d (Manual minus tuned delay parameters)**. A planning agent that uses the **Manual**
 401 abstraction, but has the same delay parameter (e.g., 1) across all processes.
 402 9. **No invent**. An ablation that uses the initial abstractions and does not perform any learning.
 403



412 Figure 6: Performance metrics for various agents across different domains. The error bars/shaded
 413 regions show ± 1 standard deviation.
 414

415 **Results and Discussion.** Figure 6 shows the evaluation solve rate for all approaches and the learning
 416 curve of our online learning agents.
 417

418 **(Q1).** Our approach consistently outperforms the VLM planning (**ViLA**), HRL (**MAPLE**) and
 419 STRIPS-style operator learning and planning (**VisPred**) approaches, achieving a near-perfect score
 420 across all domains. For each environment, ExoPredicator learns 1-4 exogenous processes and
 421 converges after at most three online interaction iterations (fig. 6b). Once trained, it can solve nearly all
 422 tasks in `Coffee`, `Domino`, and `Fan`, and over 80% in `Boil` and `Grow`. In comparison, we find
 423

432 that **MAPLE** is unable to achieve a high level of success even with 1000 times more interactions
 433 and evaluated only on the training distribution. We hypothesize that this is largely due to the chal-
 434 lenge of exploration with sparse rewards. **ViLa** (zero-shot) was not able to do well in any domains.
 435 The few-shot variant achieves good performance in simple domains where satisfying plans share
 436 significant similarity with the demonstration plan (e.g., in `Coffee`, one simply needs to perform
 437 `Pour` and `NoOp` more times than in the demonstration). We observe that its performance degrades
 438 significantly in other domains where it must identify additional rules through trial and error (e.g.,
 439 the requirement of matching colors in `Grow`) or domains that require compositional generalization
 440 (sequencing skills in potentially new ways). **VisPred** also struggles in these tasks because it learns
 441 in a highly constrained model space. It attempted to learn the exogenous processes as different op-
 442 erators for the `NoOp` skill, but fell short due to an overly strong inductive bias. This bias restricts
 443 preconditions to only include atoms with variables already present in an operator’s effects or option,
 444 which is especially limiting for the `NoOp` skill, where “robot” is the only variable. Moreover, it
 445 does not learn about the varying delays for different processes, a feature crucial for effective and
 446 efficient planning (e.g., in `Boil`). We note that **ExoPredicator** is unable to solve all tasks in `Boil`,
 447 partly because it failed to recognize the full **disjunctive** condition under which a spill can happen:
 448 it learned a process for water spilling when there is nothing under the faucet, but not when the jug
 449 underneath is full.

450 **(Q2).** **Ours** achieves the same (in `Coffee` and `Fan`) or better (in `Grow` and `Domino`) performance
 451 as **Manual** which uses manually engineered abstractions. We attribute this to the parameters learned
 452 via variational inference instead of being manually tuned. Furthermore, the near-zero performance
 453 of **No invent** and **Manual-d** underscores the importance of model learning and parameter learning.
 454 For example, in `Domino` their incomplete knowledge of the environment’s dynamics and delay
 455 caused them to classify most tasks as unsolvable.

456 **(Q3).** Both the Bayesian model learning and the LLM guidance play a critical role in efficient,
 457 effective, and robust model structure learning. Without LLM guidance, the size of the search space
 458 for each process becomes astronomically large (sometimes reaching 2^{50}), making it intractable to
 459 score all possible conditions. Without computing the Bayesian posterior of the data and model, the
 460 selection is based entirely on the prior in the LLM, which is not always reliable, especially with
 461 uncommon or unseen environment dynamics.

462 7 RELATED WORKS

463 **Temporal Planning.** Classical planners handle effects after a delay with durative actions (PDDL
 464 2.1) (Fox & Long, 2003) and with autonomous processes and events (PDDL+) (Fox & Long, 2002);
 465 (RDDL) (Sanner et al., 2010), while heuristic search-based temporal planners such as COLIN
 466 (Coles et al., 2012) or OPTIC (Benton et al., 2012) support numeric fluents and deadlines. These
 467 works assume the full domain description is given. Our contribution is complementary: we *learn*
 468 models—including conditional stochastic delays—directly from a small number of trajectories.

469 **Hierarchical Reinforcement Learning.** HRL uses temporally extended actions to address long-
 470 horizon decision-making (Barto & Mahadevan, 2003). While many skill-learning approaches exist,
 471 they typically adopt the (semi-)Markov assumption at the option level: the distribution over out-
 472 comes depends only on the initiation state and the chosen option (Masson et al., 2016; Nasiriany
 473 et al., 2022; Mishra et al., 2023). This fails to explicitly model exogenous dynamics or variable
 474 delays, attributing all changes to the agent’s actions. We relax this assumption by learning a model
 475 for these external dynamics and stochastic delays, allowing a fixed set of skills to be flexibly used
 476 as the environment evolves.

477 **Hierarchical Reinforcement Learning.** HRL uses temporally extended actions to address long-
 478 horizon decision-making (Barto & Mahadevan, 2003). While many skill-learning approaches exist,
 479 they typically adopt the (semi-)Markov assumption at the option level: the distribution over out-
 480 comes depends only on the initiation state and the chosen option (Masson et al., 2016; Nasiriany
 481 et al., 2022; Mishra et al., 2023). **This fails to explicitly model exogenous dynamics or variable**
 482 **delays, attributing all changes to the agent’s actions.** In such formulations, environment-driven or
 483 exogenous dynamics can be absorbed into the option’s transition model, but are not usually rep-
 484 resented as separate causal processes with their own activation conditions and delay distributions.
 485 **We relax this assumption by learning a model for these external dynamics and stochastic delays,**

486 ~~allowing a fixed set of skills to be flexibly used as the environment evolves.~~ Our approach is com-
 487plementary: we learn an explicit model for these external dynamics and stochastic delays, allowing
 488 a fixed set of skills to be used more flexibly as the environment evolves and enabling more general-
 489 izable planning.

490 **Large Foundation Models for Robotics.** Approaches such as SayCan (Ahn et al., 2022), RT-2
 491 (Brohan et al., 2023), Inner Monologue (Huang et al., 2022), Code-as-Policy (Liang et al., 2023),
 492 ViLA (Hu et al., 2023), and π_0 (Black et al., 2025) treat planning as prompting: a pretrained
 493 LLM/VLM selects or synthesizes the next action at each step. These approaches inherit strong
 494 general-language priors, but—because they do not learn a world model—struggle to reason about
 495 concurrent processes (e.g. water keeps heating) or about actions whose effects materialize only if
 496 certain conditions persist. Our method calls foundation models for predicate invention (as in Visu-
 497 alPredicator) and model learning, yet it grounds their suggestions in experience and learns symbolic
 498 world models that supports look-ahead search.

499 **Causal Reasoning and Causal RL.** Structural causal models (SCMs) (Pearl, 2009) and their dy-
 500 namic extensions form the foundation of various recent causal-RL algorithms (Buesing et al., 2019;
 501 Hammond et al., 2023; Zeng et al., 2025). These approaches assume that the underlying causal graph
 502 is either known or learnable at the feature level, but they do not tackle the challenges of symbolic
 503 abstraction or planning over durative processes. In contrast, ExoPredicator learns a causally consist-
 504 ent SCM (Rubenstein et al., 2017) whose variables are invented predicates and whose mechanisms
 505 are the learned causal processes, which enables reasoning at a higher level of abstraction.

506 **Learning Abstractions for Planning.** Early work learned STRIPS or NDR transition rules from
 507 demonstrations given a fixed predicate set (Pasula et al., 2007; Silver et al., 2021; 2022; Chitnis
 508 et al., 2022). Recent methods invent new predicates to improve generalisation (Silver et al., 2023;
 509 Liang et al., 2024); however, ~~all assume instantaneous deterministic effects~~ they typically abstract
 510 dynamics through agent-centric actions, without explicitly modeling exogenous causal processes
 511 that unfold over time in the background. ExoPredicator extends predicate-invention to environments
 512 with exogenous dynamics and delayed causal effect.

514 8 CONCLUSION

516 We presented ExoPredicator, an integrated approach for learning and planning with causal processes
 517 in environments with exogenous dynamics and delayed effects. Our method demonstrates the ability
 518 to learn abstract world models from limited data, generalizing to new tasks with unseen objects and
 519 goals across various simulated environments, and outperforming key baselines. Future work will
 520 scale the framework to more complex, ~~noisier~~, larger-scale environments, enhance learning with
 521 foundation models, and explore the interplay between skill and world modeling.

523 REFERENCES

525 Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea
 526 Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say:
 527 Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

528 Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Jiuguang Wang, Tomás Lozano-Pérez,
 529 and Leslie Pack Kaelbling. From pixels to predicates: Learning symbolic world models via
 530 pretrained vision-language models. *arXiv preprint arXiv:2501.00296*, 2024.

531 Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning.
 532 *Discrete event dynamic systems*, 13:341–379, 2003.

533 J Benton, Amanda Coles, and Andrew Coles. Temporal planning with preferences and time-
 534 dependent continuous costs. In *Proceedings of the International Conference on Automated Plan-
 535 ning and Scheduling*, volume 22, pp. 2–10, 2012.

536 Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo
 537 Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0 : A vision-language-action flow
 538 model for general robot control. *corr*, abs/2410.24164, 2024. doi: 10.48550. *RSS 2025*, 2025.

540 Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action
541 models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

542

543 Lars Buesing, Theophane Weber, Yori Zwols, Nicolas Heess, Sebastien Racaniere, Arthur Guez, and Jean-Baptiste Lespiau. Woulda, coulda, shoulda: Counterfactually-guided policy search. In *International Conference on Learning Representations*, 2019.

544

545 Rohan Chitnis, Tom Silver, Joshua B Tenenbaum, Tomas Lozano-Perez, and Leslie Pack Kaelbling.
546 Learning neuro-symbolic relational transition models for bilevel planning. In *2022 IEEE/RSJ
547 International Conference on Intelligent Robots and Systems (IROS)*, pp. 4166–4173. IEEE, 2022.

548

549

550 Amanda Jane Coles, Andrew I Coles, Maria Fox, and Derek Long. Colin: Planning with continuous
551 linear numeric change. *Journal of Artificial Intelligence Research*, 44:1–96, 2012.

552

553 Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games,
554 robotics and machine learning, 2016.

555

556 Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford.
557 Provably filtering exogenous distractors using multistep inverse dynamics. In *International Con-
558 ference on Learning Representations*, 2021.

559

560 Maria Fox and Derek Long. Pddl+: Modeling continuous time dependent effects. In *Proceedings of
561 the 3rd International NASA Workshop on Planning and Scheduling for Space*, volume 4, pp. 34,
562 2002.

563

564 Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning
565 domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

566

567 Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack
568 Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of
control, robotics, and autonomous systems*, 4(1):265–293, 2021.

569

570 Lewis Hammond, James Fox, Tom Everitt, Ryan Carey, Alessandro Abate, and Michael Wooldridge.
571 Reasoning about causality in games. *Artificial Intelligence*, 2023.

572

573 Jörg Hoffmann. Ff: The fast-forward planning system. *AI magazine*, 22(3):57–57, 2001.

574

575 Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the
576 power of gpt-4v in robotic vision-language planning. *arXiv preprint arXiv:2311.17842*, 2023.

577

578 Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan
579 Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through
580 planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

581

582 Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*,
583 2014.

584

585 George Konidaris. On the necessity of abstraction. *Current opinion in behavioral sciences*, 29:1–7,
586 2019.

587

588 Nishanth Kumar, Tom Silver, Willie McClinton, Linfeng Zhao, Stephen Proulx, Tomás Lozano-
589 Pérez, Leslie Pack Kaelbling, and Jennifer Barry. Practice makes perfect: Planning to learn skill
590 parameter policies, 2024.

591

592 Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and
593 Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE
594 International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

595

596 Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B Tenenbaum, Tom Silver,
597 João F Henriques, and Kevin Ellis. Visualpredicator: Learning abstract world models with neuro-
598 symbolic predicates for robot planning. *arXiv preprint arXiv:2410.23156*, 2024.

594 Warwick Masson, Pravesh Ranchod, and George Konidaris. Reinforcement learning with param-
 595 eterized actions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30,
 596 2016.

597 Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining:
 598 Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pp. 2905–
 599 2925. PMLR, 2023.

600 Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior
 601 primitives for diverse manipulation tasks. In *2022 International Conference on Robotics and*
 602 *Automation (ICRA)*, pp. 7477–7484. IEEE, 2022.

603 Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling. Learning symbolic models of
 604 stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

605 Judea Pearl. *Causality*. Cambridge university press, 2009.

606 Paul K Rubenstein, Sebastian Weichwald, Stephan Bongers, Joris M Mooij, Dominik Janzing,
 607 Moritz Grosse-Wentrup, and Bernhard Schölkopf. Causal consistency of structural equation mod-
 608 els. *arXiv preprint arXiv:1707.00819*, 2017.

609 Scott Sanner et al. Relational dynamic influence diagram language (rddl): Language description.
 610 *Unpublished ms. Australian National University*, 32:27, 2010.

611 Tom Silver, Rohan Chitnis, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez.
 612 Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International*
 613 *Conference on Intelligent Robots and Systems (IROS)*, pp. 3182–3189. IEEE, 2021.

614 Tom Silver, Ashay Athalye, Joshua B Tenenbaum, Tomas Lozano-Perez, and Leslie Pack Kaelbling.
 615 Learning neuro-symbolic skills for bilevel planning. *arXiv preprint arXiv:2206.10680*, 2022.

616 Tom Silver, Rohan Chitnis, Nishanth Kumar, Willie McClinton, Tomás Lozano-Pérez, Leslie Kael-
 617 bling, and Joshua B Tenenbaum. Predicate invention for bilevel planning. In *Proceedings of the*
 618 *AAAI Conference on Artificial Intelligence*, volume 37, pp. 12120–12129, 2023.

619 Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu,
 620 Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly
 621 capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

622 Lionel Wong, Katherine M Collins, Lance Ying, Cedegao E Zhang, Adrian Weller, Tobias Ger-
 623 stenberg, Timothy O’Donnell, Alexander K Lew, Jacob D Andreas, Joshua B Tenenbaum, et al.
 624 Modeling open-world cognition as on-demand synthesis of probabilistic models. *arXiv preprint*
 625 *arXiv:2507.12547*, 2025.

626 Yan Zeng, Ruichu Cai, Fuchun Sun, Libo Huang, and Zhifeng Hao. A survey on causal reinforce-
 627 ment learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648	CONTENTS	
649		
650		
651	A Additional Approach details	14
652	A.1 Causal Process Semantics	14
653	A.2 Fast Forward Heuristic for Causal Processes	15
654	A.3 Probabilistic Model and ELBO Derivation	16
655	A.4 Process Learning details	18
656	A.5 Predicate Learning Details	19
657		
658		
659		
660	B Additional Environment details	24
661	B.1 Coffee	24
662	B.2 Grow	25
663	B.3 Boil	27
664	B.4 Domino	29
665	B.5 Fan	31
666		
667		
668		
669	C Additional Experiment Details	31
670	C.1 Learned Causal Processes	31
671	C.2 Further Learning and Planning Statistics	41
672	C.3 Priors and Distributions	41
673		
674		
675		
676		
677		
678		
679		
680		
681		
682		
683		
684		
685		
686		
687		
688		
689		
690		
691		
692		
693		
694		
695		
696		
697		
698		
699		
700		
701		

702 **A ADDITIONAL APPROACH DETAILS**
703704 **A.1 CAUSAL PROCESS SEMANTICS**
705

706 We formalize the semantics of our causal process model, which underpins the planner described in
707 section 4. We begin by defining a **small-step transition function** that describes the world’s evo-
708 lution at the finest temporal granularity, advancing one discrete timestep at a time. This detailed
709 model allows us to precisely specify how and when processes are activated and their effects are ap-
710 plied. We then build upon this to define the **big-step transition function**, T_{big} , which abstracts away
711 these fine-grained details. This function enables the planner to efficiently jump between significant
712 changes in the abstract state, which is crucial for tractable long-horizon planning.

713 **Small-Step Semantics.** We model the world’s evolution in discrete timesteps $t \in \mathbb{N}$. A complete
714 snapshot of the world, or the **world state**, is a tuple $w_t = \langle s_t, Q_t, H_t \rangle$, where:

- 715 • s_t is the set of ground atoms that are currently true.
- 716 • Q_t is the *event dictionary*, a dictionary of scheduled effects of the form $\langle \underline{L}, t_{\text{start}} \rangle$, keyed by their
717 end time t_{end} .
- 718 • H_t is the history of all past atomic states, $[s_0, s_1, \dots, s_{t-1}]$.

720 The world’s fundamental dynamics are defined by a **small-step transition function**,
721 $\mathcal{T}_{\text{small}}(w_t, \alpha_t) \mapsto w_{t+1}$, which advances the world by a single timestep. The transition given a
722 potential agent command α_t (which may be an ground endogenous process or `None`) occurs in
723 three stages:

- 724 1. **Event Execution:** Effects from events due at time t are applied. We initialize $s_{t+1} \leftarrow s_t$. For
725 every event $\langle \underline{L}, t_{\text{start}} \rangle$ in Q_t scheduled for time t , if its overall condition $O_{\underline{L}}$ held from the step
726 after activation up to the previous step, (i.e., for all $s_i \in H_t$ where $i > t_{\text{start}}$), its effects are
727 applied: $s_{t+1} \leftarrow (s_{t+1} \setminus E_{\underline{L}}.\text{Del}) \cup E_{\underline{L}}.\text{Add}$.
- 728 2. **Process Activation:** New events are scheduled based on the state s_{t+1} and the agent’s command.
 - 729 • *Endogenous Activation:* If the agent issues a command $\alpha_t = \underline{L}_{\text{en}}$ and the process’s start con-
730 dition $C_{\text{start}, \underline{L}_{\text{en}}}$ is satisfied in s_{t+1} , a delay $d \sim p_{\underline{L}_{\text{en}}}^{\text{delay}}$ is sampled (with $d \geq 1$). A new event
731 $\langle \underline{L}_{\text{en}}, t \rangle$ is added to the queue for time $t + d$.
 - 732 • *Exogenous Activation:* For every exogenous process $\underline{L}_{\text{ex}}$, if its start condition $C_{\text{start}, \underline{L}_{\text{ex}}}$ is satis-
733 fied in s_{t+1} but was *not* satisfied in the previous state s_{t-1} (i.e., it is edge-triggered), a delay
734 $d \sim p_{\underline{L}_{\text{ex}}}^{\text{delay}}$ is sampled. A new event $\langle \underline{L}_{\text{ex}}, t \rangle$ is added to the queue for time $t + d$.
- 736 3. **State Finalization:** The next world state is $w_{t+1} = \langle s_{t+1}, Q'_t, H_t \cup \{s_t\} \rangle$, where Q'_t is the
737 updated event dictionary.

739 **Big-Step Semantics.** We define a **big-step transition function**, $\mathcal{T}_{\text{big}}(w_t, \underline{L}_{\text{en}})$, which computes
740 the resulting world state after executing a single ground endogenous process $\underline{L}_{\text{en}}$ starting from world
741 state w_t , or after simply waiting for the world to change ($\underline{L}_{\text{en}} = \text{NoOp}$).

742 This function simulates the environment forward by applying the small-step transition function $\mathcal{T}_{\text{small}}$
743 iteratively. The simulation proceeds until the chosen action $\underline{L}_{\text{en}}$ has completed or a maximum hori-
744 zon K_{max} is reached.

746 The transition $\mathcal{T}_{\text{big}}(w_t, \underline{L}_{\text{en}}) \mapsto w_{t+k}$ is computed as follows:

- 747 1. **Initialization:**
 - 748 • Initialize a step counter: $k \leftarrow 0$.
 - 749 • Set the initial world state for the simulation: $w'_k \leftarrow w_t$.
 - 750 • The endogenous process $\underline{L}_{\text{en}}$ is set as the command for the first step. Let the command for
751 step i be denoted α_i . So, $\alpha_k \leftarrow \underline{L}_{\text{en}}$. For all subsequent steps $i > 0$, the command is null:
752 $\alpha_i \leftarrow \text{None}$.
- 753 2. **Simulation Loop:** While the action $\underline{L}_{\text{en}}$ is still considered active and $k < K_{\text{max}}$:
 - 754 • Apply the small-step transition: $w'_{k+1} \leftarrow \mathcal{T}_{\text{small}}(w'_k, \alpha_k)$.
 - 755 • Increment the step counter: $k \leftarrow k + 1$.

756 • The action $\underline{L}_{\text{en}}$ is considered complete if its corresponding event has been executed within the
 757 simulation. This is tracked implicitly by the simulator state. A special case is the NoOp action,
 758 which is considered complete if any atom changes in the state, allowing the agent to wait for
 759 exogenous events.

760 3. **Final State:** The resulting world state is the state at the end of the simulation loop: $w_{t+k} \leftarrow w'_k$.

762 A.2 FAST FORWARD HEURISTIC FOR CAUSAL PROCESSES

764 The Fast-Forward (FF) heuristic (Hoffmann, 2001) is a domain-independent planning heuristic that
 765 estimates the distance to goal by solving a relaxed version of the planning problem. We adapt
 766 this heuristic to our causal process framework, accounting for both endogenous and exogenous
 767 processes, as well as derived predicates.

769 **Relaxed Planning Graph Construction** The FF heuristic constructs a Relaxed Planning Graph
 770 (RPG) by iteratively applying all applicable processes without considering delete effects. Given a
 771 state with atoms s , the heuristic proceeds as follows:

- 772 1. **Initialization:** Start with the current atoms s , augmented with any derived predicates that hold
 773 given those atoms.
- 774 2. **Forward Propagation:** For each layer i :
 - 776 • Find all processes whose C (condition at start) is satisfied by facts in layer $i - 1$
 - 777 • Add all add effects $E.\text{Add}$ from these processes to create layer i (ignoring delete effects $E.\text{Del}$)
 - 778 • Incrementally compute new derived predicates based on newly added primitive facts
- 779 3. **Termination:** Stop when the goal atoms $g \subseteq$ layer i , or when a fixed point is reached (no new
 780 facts can be added).

782 **Incremental Derived Predicate Computation** To efficiently handle derived predicates, we main-
 783 tain a dependency map from auxiliary predicates to derived predicates. When new primitive facts
 784 are added to a layer, we:

- 785 1. Identify which derived predicates might be affected based on their auxiliary predicate dependen-
 786 cies
- 788 2. Incrementally evaluate only those derived predicates on the updated state
- 789 3. Propagate newly derived facts through the dependency chain until a fixed point is reached

791 This avoids redundant recomputation of derived predicates that cannot be affected by the new facts.

792 **Relaxed Plan Extraction** Once the RPG is built, we extract a relaxed plan via backward search:

- 794 1. Start with the goal atoms as subgoals to achieve
- 795 2. For each layer i from n to 1:
 - 796 • For each subgoal appearing for the first time in layer i :
 - 797 – If it's a derived predicate, replace it with its supporting auxiliary predicates
 - 798 – If it's a primitive predicate, find a process from layer $i - 1$ that achieves it
 - 800 • Add the preconditions of selected processes as new subgoals
 - 801 • Count only endogenous processes toward the heuristic value

802 **Heuristic Value** The heuristic value $h_{\text{FF}}(s)$ is the number of endogenous processes in the ex-
 803 tracted relaxed plan. Exogenous processes are treated as having zero cost, reflecting that they occur
 804 automatically when their conditions are met. Formally:

$$807 \quad h_{\text{FF}}(s) = |\{L \in \text{RelaxedPlan} : L \text{ is endogenous}\}| \quad (1)$$

808 This provides an admissible estimate when all action costs are uniform, and guides the search toward
 809 states that require fewer agent interventions to reach the goal.

810 **Implementation Notes** The implementation uses several optimizations:
 811

812 • **Add-effect indexing:** We maintain a map from atoms to processes that add them, enabling effi-
 813 cient backward search during plan extraction
 814 • **Early termination:** If the RPG reaches a fixed point without achieving the goal, we return $h = \infty$
 815 • **Zero-cost exogenous processes:** These are included in the RPG construction but not counted in
 816 the final heuristic value, allowing the planner to leverage environmental dynamics
 817

818 **A.3 PROBABILISTIC MODEL AND ELBO DERIVATION**
 819

820 The derivation for the probabilistic model is:
 821

$$\begin{aligned}
 & p(\{s_t\}, \{\Delta_t^L\}) \\
 &= (\text{the law of conditional probability}) \\
 & \prod_{t=1}^T p\left(s_t, \{\Delta_t^L\} | s_{1:t-1}, \{\Delta_{1:t-1}^L\}\right) \\
 &= (\text{the law of conditional probability}) \\
 & \prod_{t=1}^T p\left(\{\Delta_t^L\} | s_{1:t}, \{\Delta_{1:t-1}^L\}\right) \times p\left(s_t | s_{1:t-1}, \{\Delta_{1:t-1}^L\}\right) \\
 &= (\text{conditional independence of the delay vars.}) \\
 & \left(\prod_{t=1}^T \prod_{\underline{L}} p\left(\Delta_t^L | s_{1:t}\right) \right) \times \left(\prod_{t=1}^T p\left(s_t | s_{1:t-1}, \{\Delta_{1:t-1}^L\}\right) \right) \\
 &= (\text{by the structure of our model}) \\
 & \left(\prod_{t=1}^T \prod_{\underline{L}} p_{\underline{L}}^{\text{delay}}(\Delta_t^L)^{C_{\underline{L}}(s_{1:t})} \right) \times \left(\prod_{t=1}^T \frac{1}{Z_t} F(s_t | s_{t-1}) \prod_{\underline{L}} \prod_{t'=1}^{t-1} E_{\underline{L}}(s_t)^{C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[t=t'+\Delta_{t'}^L]} \right) \\
 & \text{where } Z_t = \sum_{s_t \in \mathcal{S}} F(s_t | s_{t-1}) \prod_{\underline{L}} \prod_{t'=1}^{t-1} E_{\underline{L}}(s_t)^{C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[t=t'+\Delta_{t'}^L]} \\
 &= (\text{assume factored state } s_t = (s_t^1, s_t^2, \dots)) \\
 & \left(\prod_{t=1}^T \prod_{\underline{L}} p_{\underline{L}}^{\text{delay}}(\Delta_t^L)^{C_{\underline{L}}(s_{1:t})} \right) \times \left(\prod_{t=1}^T \prod_{j=1}^J \frac{1}{Z_t^j} F(s_t^j | s_{t-1}^j) \prod_{\underline{L}} \prod_{t'=1}^{t-1} E_{\underline{L}}^j(s_t^j)^{C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[t=t'+\Delta_{t'}^L]} \right) \\
 & \text{where } Z_t^j = \sum_{s_t^j \in \mathcal{S}^j} F(s_t^j | s_{t-1}^j) \prod_{\underline{L}} \prod_{t'=1}^{t-1} E_{\underline{L}}^j(s_t^j)^{C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[t=t'+\Delta_{t'}^L]}
 \end{aligned}$$

854 With the change of basis, our model becomes:
 855

$$\begin{aligned}
 & p(\{s_t\}, \{A_t^L\}) = \\
 & \left(\prod_{t=1}^T \prod_{\underline{L}} p_{\underline{L}}^{\text{delay}}(A_t^L - t)^{C_{\underline{L}}(s_{1:t})} \right) \times \left(\prod_{t=1}^T \prod_{j=1}^J \frac{1}{Z_t^j} F(s_t^j | s_{t-1}^j) \prod_{\underline{L}} \prod_{t'=1}^{t-1} E_{\underline{L}}^j(s_t^j)^{C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[A_{t'}^L = t]} \right) \\
 & \text{where } Z_t^j = \sum_{s_t^j \in \mathcal{S}^j} F(s_t^j | s_{t-1}^j) \prod_{\underline{L}} \prod_{t'=1}^{t-1} E_{\underline{L}}^j(s_t^j)^{C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[A_{t'}^L = t]}
 \end{aligned}$$

864 The derivation for the ELBO is:

865

866

867

$$868 \log p(\{s_t\})$$

$$869 = \log \int \frac{q(\{A_t^L\})}{q(\{A_t^L\})} p(\{s_t\}, \{A_t^L\}) d\{A_t^L\}$$

$$870 = \log \mathbb{E}_q \left[\frac{p(\{s_t\}, \{A_t^L\})}{q(\{A_t^L\})} \right]$$

$$871 \geq \mathbb{E}_q \left[\log \frac{p(\{s_t\}, \{A_t^L\})}{q(\{A_t^L\})} \right]$$

$$872 = \mathbb{E}_q \left[\log p(\{s_t\}, \{A_t^L\}) \right] + \mathbb{H}_q \left[\{A_t^L\} \right]$$

$$873 = \sum_{t=1}^T \sum_{\underline{L}} \underbrace{\mathbb{E}_q \left[C_{\underline{L}}(s_{1:t}) \log \left(p_{\underline{L}}^{\text{delay}}(A_t^L - t) \right) \right]}_{\text{expected log delay probability}} +$$

$$874 \sum_{t=1}^T \sum_{j=1}^J \log \left(F(s_t^j | s_{t-1}^j) \right) + \sum_{\underline{L}} \sum_{t'=1}^{t-1} \underbrace{\mathbb{E}_q \left[C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[A_{t'}^L = t] \log \left(E_{\underline{L}}^j(s_t^j) \right) \right]}_{\text{unnormalized expected log state probability}} -$$

875

$$876 \sum_{t=1}^T \sum_{j=1}^J \mathbb{E}_q \left[\log \left(\sum_{\hat{s}_t^j \in \mathcal{S}^j} F(\hat{s}_t^j | s_{t-1}^j) \prod_{\underline{L}} \prod_{t'=1}^{t-1} E_{\underline{L}}^j(\hat{s}_t^j) C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[A_{t'}^L = t] \right) \right] + \mathbb{H}_q \left[\{A_t^L\} \right]$$

877

$$878 \geq (\text{reverse bound the expected normalization constant; independence of } A_t^L)$$

$$879 \sum_{t=1}^T \sum_{\underline{L}} \mathbb{E}_q \left[C_{\underline{L}}(s_{1:t}) \log \left(p_{\underline{L}}^{\text{delay}}(A_t^L - t) \right) \right] +$$

$$880 \sum_{t=1}^T \sum_{j=1}^J \log \left(F(s_t^j | s_{t-1}^j) \right) + \sum_{\underline{L}} \sum_{t'=1}^{t-1} \mathbb{E}_q \left[C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[A_{t'}^L = t] \log \left(E_{\underline{L}}^j(s_t^j) \right) \right] -$$

$$881 \sum_{t=1}^T \sum_{j=1}^J \log \sum_{\hat{s}_t^j \in \mathcal{S}^j} F(\hat{s}_t^j | s_{t-1}^j) \prod_{\underline{L}} \prod_{t'=1}^{t-1} \mathbb{E}_q \left[E_{\underline{L}}^j(\hat{s}_t^j) C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) 1[A_{t'}^L = t] \right] + \mathbb{H}_q \left[\{A_t^L\} \right]$$

882

$$883 = (\text{expand the expectation out independently})$$

$$884 \sum_{t=1}^T \sum_{\underline{L}} C_{\underline{L}}(s_{1:t}) \sum_{\substack{t=1 \\ A_t^L = t+1}}^T q_t^L(A_t^L) \log \left(p_{\underline{L}}^{\text{delay}}(A_t^L - t) \right) +$$

$$885 \sum_{t=1}^T \sum_{j=1}^J \log \left(F(s_t^j | s_{t-1}^j) \right) + \sum_{\underline{L}} \sum_{t'=1}^{t-1} q_{t'}^L(A_{t'}^L = t) C_{\underline{L}}(s_{1:t'}) O_{\underline{L}}(s_{t'+1:t-1}) \log \left(E_{\underline{L}}^j(s_t^j) \right) -$$

$$886 \sum_{t=1}^T \sum_{j=1}^J \log \sum_{\hat{s}_t^j \in \mathcal{S}^j} F(\hat{s}_t^j | s_{t-1}^j) \prod_{\underline{L}} \prod_{t'=1}^{t-1} \left(q_{t'}^L(A_{t'}^L = t) \underbrace{E_{\underline{L}}^j(\hat{s}_t^j) O_{\underline{L}}(s_{t'+1:t-1})}_{\exp(O_{\underline{L}}(s_{t'+1:t-1}) \log E_{\underline{L}}^j(\hat{s}_t^j))} + (1 - q_{t'}^L(A_{t'}^L = t)) \right)^{C_{\underline{L}}(s_{1:t'})} +$$

$$887 \mathbb{H}_q \left[\{A_t^L\} \right]$$

Evaluating this objective is computationally intensive due to the nested loops, but can be simplified by some algebraic refactoring. In the first term, the second sum only needs to loop through the non-zero terms, which are laws whose conditions are satisfied at time t .

In the second term, the sum over j and L can be reduced to a sum over s_t^j 's that are in the effects of some laws, and thus have non-zero $\log(E_L^j(s_t^j))$ plus the log frame strength from unchanged atoms, and the sum over t' can be reduced to just steps where the law is activated, similar to the first term above.

For experiments, we use a categorical variational distribution over the discrete support of the delays. The variational parameters are initialized to be uniform over this support.

A.4 PROCESS LEARNING DETAILS

Given a set of trajectories \mathcal{D}_{abs} , a set of predicates Ψ , and the agent's known endogenous processes \mathcal{L}_{en} , we learn the set of exogenous processes \mathcal{L}_{ex} . Our method follows Chitnis et al. (2022) in assuming that for any given effect (a unique pair of add/delete atoms), there is at most one exogenous process that causes it. While this prevents learning multiple distinct causes for the same outcome, it significantly simplifies the search problem. Any lost expressivity can be recovered by inventing more nuanced predicates. The learning algorithm proceeds in five steps:

1. **Segment.** First, we split each raw trajectory into shorter segments based on changes in the abstract state. Specifically, a new segment begins whenever the set of true predicates changes. Each segment therefore contains a sequence of constant abstract states followed by a single timestamp where the state changes.
2. **Filter.** Next, we filter out any segments where the observed state change can be explained by one of the agent's known endogenous processes. For example, if the agent executes the `Pick` action and the `Holding` predicate becomes true, that segment is attributed to an endogenous process and removed from consideration. This ensures we only attempt to learn models for effects caused by the environment's own dynamics.
3. **Cluster.** We then cluster the remaining segments based on their effects. We assume that each exogenous process has a single, atomic effect (e.g., one predicate changing from false to true).² If a segment involves multiple predicate changes, we duplicate it into multiple clusters—one for each change—allowing us to learn a separate process for each atomic effect. **This induces a deterministic partition of segments by effect, so the step introduces no clustering hyperparameters.**
4. **Intersect.** For each cluster, we identify a set of potential preconditions for the associated effect. This is done by taking the set intersection of all predicates that were true at the start of every segment in the cluster. This step produces a superset of candidate atoms for the process's conditions.
5. **Select.** The intersection from the previous step often contains many irrelevant atoms. To find the true preconditions, we first use an LLM to propose a small number of plausible condition sets from this large superset (the prompt is detailed in the end of this section). We then use Bayesian model selection to score each candidate condition set C_i and select the one that maximizes the posterior probability:

$$L_{C^*, E} = \arg \max_{L_{C_i, E}} \log p(L_{C_i, E} | \mathcal{D}_{\text{abs}}) = \arg \max_{L_{C_i, E}} (\log p(\mathcal{D}_{\text{abs}} | L_{C_i, E}) + \log p(L_{C_i, E}))$$

where $\log p(\mathcal{D}_{\text{abs}} | L_{C_i, E})$ is the approximate marginal likelihood from the previous section and $\log p(L_{C_i, E})$ is a minimum description length prior that penalizes overly complex conditions.

Process condition proposal prompt

²This imposes no loss of generality, because separate processes can be learned for each changed predicate.

```

972
973 You are an expert in automated planning and causal reasoning. Your
974 task is to propose the most likely sets of conditions for
975 specific process effects to occur.
976
977 Given a process with specific add effects and delete effects, you
978 need to propose multiple coherent sets of candidate atoms that
979 could serve as necessary conditions for the process to
980 successfully achieve its effects.
981
982 Key principles for proposing condition sets:
983 1. **Causal Relevance**: Each set should contain atoms that are
984 causally necessary for the effects to occur
985 2. **Physical Constraints**: Include atoms representing physical
986 constraints or requirements
987 3. **Domain Knowledge**: Use common sense about how processes work
988 in the real world
989 4. **State Dependencies**: Include atoms that represent
990 prerequisite states for the effects
991 5. Terminal-Progress Exclusion: If an add effect is a
992 terminal/complete state within a progression family, do not
993 include any intermediate/progress predicates from the same
994 family as preconditions (e.g., Partially*, Started, InProgress,
995 HasSome).
996
997 Available predicates in the candidate atoms are:
998 {PREDICATE_LISTING}
999
1000 For each process, I will provide:
1001 - Add effects: What the process makes true
1002 - Delete effects: What the process makes false
1003 - Candidate atoms: Potential precondition atoms to choose from
1004
1005 {PROCESS_EFFECTS_AND_CANDIDATES}
1006
1007 Please propose as many likely condition sets as you deem suitable
1008 for each process. Each condition set should be a coherent
1009 combination of atom indices that together form a plausible set
1010 of preconditions. It's possible that there is a large number of
1011 atoms in a condition set in some cases.
1012
1013 Think step by step if it's helpful before outputting your final
1014 response, formatted strictly as:
1015 <answer>
1016 Process 0:
1017 Set 1: [2, 0, 4]
1018 Set 2: [1, 3, 0, 5]
1019 Set 3: [2, 4]
1020 Process 1:
1021 Set 1: [1, 3]
1022 Set 2: [0, 1, 3]
1023 ...
1024 </answer>
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
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
316
```

1026 Our proposal process involves two stages. First, we prompt a VLM with a trajectory from the
 1027 environment to propose a set of high-level, symbolic concepts that could be useful for planning. We
 1028 use different prompts depending on whether the provided trajectory was successful or resulted in
 1029 failure. In the second stage, these concepts are translated into executable Python code that matches
 1030 our predicate object API.

1031 Since the primary focus of this work is on learning abstract models in a more expressive and com-
 1032 plex model space, we simplify the perception problem. We assume the agent has access to a state
 1033 representation containing all the necessary object features to evaluate any relevant predicate, without
 1034 needing to ground them directly in image data.

1035 In our experiments, we found it sufficient to generate a pool of candidate predicates only once,
 1036 based on the initial demonstration trajectories. This set of candidates is then retained and made
 1037 available for the agent to select from during all subsequent online learning iterations. The full
 1038 prompt templates used for predicate invention are provided below.

1039

1040

1041 Predicate invention from successful trajectory prompt template

1042

1043

1044

1045 Context: You are an expert AI planning researcher. Your task is to
 1046 design task-specific predicates that can be used in a PDDL-like
 1047 model to facilitate effective and efficient robot planning.

1048

1049

1049 ### Types and Features

1050 The environment has the following types, each with some features:

1051

1051 {TYPES_IN_ENV}

1052

1052 ### Existing Predicates

1053 You should consider the following existing predicates:

1054

1054 {PREDICATES_IN_ENV}

1055

1056 ### Robot's Goal

1057 The robot's ultimate goal in this environment is to make the
 1058 predicate {GOAL_PREDICATE} true.

1059

1060 ### Demonstration Trajectory

1061 The demonstrator performs a sequence of actions, ending with the
 1062 goal being achieved. The state-feature-action trajectory is
 1063 provided below.

1064

1064 {EXPERIENCE_IN_ENV}

1065

1066 ### Your Task

1067 Invent a small set of the **most essential** new predicates. These
 1068 should be simple, primitive concepts that represent critical
 1069 ***subgoals***, ***conditions for subgoals***, or **any conditions*
 1070 *that must be maintained to prevent failure, even if failure is*
*not shown in the demonstration**.

1071

1071 Note:

- If a continuous feature represents progress toward a subgoal,
 define exactly one terminal predicate for its end state (with a
 high threshold near the value observed when the goal is
 achieved) and ignore intermediate progress states.
- Geometry/affordance guardrail: Do not invent predicates that
 rely on pose or derived geometry (distance, proximity,
 alignment, path, line-of-sight, "within theta", $\text{abs}() < \tau$),
 or on capabilities (Can*, AbleTo*, Near*), unless such
 relations are already provided as explicit non-pose features.

```

1080
1081 - There maybe some risk or violation condition that must remain
1082   safe for the goal to succeed, even if failure is not shown. For
1083   such features include a maintenance predicate that keeps it
1084   within a safe bound (e.g., T_low <= 0.1 for normalized
1085   features).
1086
1087   #### Constraints
1088   - Do *not* propose any new predicates that are purely pose-based
1089     (i.e., based on raw x, y, z coordinates or 'rot', 'tilt'
1090     angles).
1091   - Avoid composite predicates (no negation/AND/OR of other
1092     predicates). Each proposal should state one primitive property
1093     or relation that can be verified from the given features.
1094   - Assertions must be clear and unambiguous, describing the
1095     relationship or properties of variables ?<var1>, ?<var2>, etc.,
1096     so an external observer could label truth values from the
1097     provided object features alone.
1098   - Replace placeholders like <predicate1_name>, <var1>, etc., with
1099     actual names; <type1> and <obj1> with actual names and types
1100     from the state dictionary provided. *Do not* use types that are
1101     not present in the states (e.g., int or float).
1102   - Do not use bold or italic fonts in your response.
1103   - Respond only with the output section outlined below.
1104
1105   #### Output Format
1106   Provide your predicate proposals in the following format:
1107
1108   ````plaintext
1109   # Predicate Proposals
1110   * <predicate1_name>(?<var1>:<type1>, ?<var2>:<type2>, ...): <The
1111     assertion this predicate is making>.
1112   * ...
1113   ````
```

Predicate invention from failed trajectory prompt template

```

1111
1112
1113 Context: You are an expert AI planning researcher. Your task is to
1114   design task-specific predicates that can be used in a PDDL-like
1115   model to (a) detect unreachable goals early and (b) avoid
1116   futile plans.
1117
1118   #### Types and Features
1119   The environment has the following types, each with some features:
1120
1121   {TYPES_IN_ENV}
1122
1123   #### Existing Predicates
1124   You should consider the following existing predicates:
1125
1126   {PREDICATES_IN_ENV}
1127
1128   #### Robot's Goal
1129   The robot's goal in this environment is to make the predicate
1130   {GOAL_PREDICATE} true.
1131
1132   #### Demonstration Trajectory
1133   The demonstrator performs a sequence of actions, which they
1134   thought would achieve the goal but did not. The
1135   state-feature-action trajectory is provided below.
```

```

1134
1135 {EXPERIENCE_IN_ENV}
1136
1137     ### Your Task
1138     Invent a small set of the most essential new predicates whose
1139     primary purpose is to expose **blocking conditions**:
1140     primitive, easily-checkable properties that must hold somewhere
1141     in the environment for the goal to be achievable. If a blocking
1142     condition is true (or a required enabling condition is false),
1143     a rational planner can conclude that the goal is unreachable
1144     under the available actions.
1145
1146     Prioritize:
1147     - Necessary preconditions for success that the demonstrator
1148       overlooked.
1149     - Irreversible or static properties (e.g., object class, material,
1150       color category) that make certain transitions impossible.
1151     - Local checks that can be evaluated from the provided object
1152       features without simulating dynamics.
1153
1154     Note: Features such as colors may encode latent material or
1155     affordance classes. You may want to propose predicates that
1156     identify such classes when they impose hard constraints on
1157     feasible state transitions or interactions (e.g.,
1158     non-deformable, too-heavy, low-friction, brittle, sealed,
1159     non-activatable).
1160     For every new predicate, explicitly state how its truth value is
1161     decided from the listed non-pose features. Use a deterministic
1162     rule with concrete feature names and numeric thresholds or
1163     categorical equalities (no vague phrases like "such as mass").
1164     If invoking a latent property, tie it to an explicit feature
1165     pattern.
1166
1167     ### Constraints
1168     - Geometry/pose prohibition (hard): Do not use or derive from pose
1169       fields (x, y, z, yaw, roll, tilt, wrist) or any geometric
1170       constructs (distance, proximity, alignment, vector, path,
1171       line-of-sight, "within theta", abs()<tau). Predicates
1172       mentioning these are invalid.
1173     - Soft blacklist (names/definitions to avoid): aligned, path,
1174       near, distance, airstream, obstructed, adjacency (unless given
1175       as a non-pose feature).0
1176     - Avoid composite predicates (no negation/AND/OR of other
1177       predicates). Each proposal should state one primitive property
1178       or relation that can be verified from the given features.
1179     - Assertions must be clear and unambiguous, describing the
1180       relationship or properties of variables ?<var1>, ?<var2>, etc.,
1181       so an external observer could label truth values from the
1182       provided object features alone.
1183     - Replace placeholders like <predicate1_name>, <var1>, etc., with
1184       actual names; use only types present in the states.
1185     - Do not use bold or italic fonts.
1186     - Respond only with the Output section below.
1187
1188     ### Output Format
1189     Provide your predicate proposals in the following format:
1190
1191     '''plaintext
1192     # Predicate Proposals
1193     * <predicate1_name>(?<var1>:<type1>, ?<var2>:<type2>, ...): <The
1194       assertion this predicate is making>.
1195     * ...
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
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
3
```

```

1188 Predicate implementation prompt template
1189
1190
1191 Context: You are an expert AI researcher tasked with inventing
1192 task-specific state abstraction predicates for effective and
1193 efficient robotic planning.
1194
1195 I will describe the API you should use for writing predicates and
1196 the environment the robot is in.
1197 # The API for 'Predicate' and 'State' is:
1198 {STRUCT_DEFINITION}
1199
1200 The environment includes the following object-type variables with
1201 features:
1202 {TYPES_IN_ENV}
1203 where 'bbox_left', 'bbox_lower', ..., corresponds to the pixel
1204 index of the left, lower boundary of the object bounding box in
1205 the image starting from (0, 0) at the bottom left corner of the
1206 image.
1207 'pose_x', 'pose_y', and 'pose_z' correspond to the 3d object
1208 position in the world frame, so these are not comparable to the
1209 bbox values.
1210
1211 The existing predicates are:
1212 {PREDICATES_IN_ENV}
1213
1214 The states the predicates have been evaluated on are:
1215 {LISTED_STATES}
1216
1217 Please implement the following predicates which would have
1218 evaluation values that matches the following specification:
1219
1220 {PREDICATE_SPECS}
1221
1222 Implement each predicate in a seperate Python block as follows:
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
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
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
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
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3
```

1242 **B ADDITIONAL ENVIRONMENT DETAILS**
12431244 We describe the predicates and endogenous processes that we provide to ExoPredicator at the be-
1245 ginning of learning. In contrast, the baselines (**Manual**, **ViLa**, **MAPLE** and **VisualPredicator**) are
1246 provided with an expanded set of predicates that we intend our approach to discover autonomously.
12471248 **B.1 COFFEE**
12491250 **Train/Test split** The training tasks for this environment involve filling a single cup with coffee.
1251 The held-out test tasks require the agent to fill two or three cups. In both distributions, the size and
1252 color of the cups may vary.
12531254 **Goal predicates.** {CupFilled}1255 **Initial predicates and endogenous processes.** {JugAboveCup, OnTable, NotAboveCup,
1256 CupFilled, Holding, MachineOn, JugInMachine, HandEmpty}1258 EndogenousProcess-NoOp:
1259 Parameters: [?robot:robot]
1260 Conditions at start: []
1261 Conditions overall: []
1262 Conditions at end: []
1263 Add Effects: []
1264 Delete Effects: []
1265 Ignore Effects: [JugAboveCup, NotAboveCup]
1266 Log Strength: 1.0000
1267 Delay Distribution: ConstantDelay(1.0000)
1268 Option Spec: NoOp(?robot:robot),1269 EndogenousProcess-PickJugFromMachine:
1270 Parameters: [?robot:robot, ?jug:jug, ?machine:coffee_machine]
1271 Conditions at start: [HandEmpty(?robot:robot), JugInMachine(?jug:jug,
1272 ?machine:coffee_machine)]
1273 Conditions overall: []
1274 Conditions at end: []
1275 Add Effects: [Holding(?robot:robot, ?jug:jug)]
1276 Delete Effects: [HandEmpty(?robot:robot), JugInMachine(?jug:jug, ?
1277 machine:coffee_machine)]
1278 Ignore Effects: []
1279 Log Strength: 1.0000
1280 Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1281 Option Spec: PickJug(?robot:robot, ?jug:jug),1281 EndogenousProcess-PickJugFromTable:
1282 Parameters: [?robot:robot, ?jug:jug]
1283 Conditions at start: [HandEmpty(?robot:robot), OnTable(?jug:jug)]
1284 Conditions overall: []
1285 Conditions at end: []
1286 Add Effects: [Holding(?robot:robot, ?jug:jug)]
1287 Delete Effects: [HandEmpty(?robot:robot), OnTable(?jug:jug)]
1288 Ignore Effects: []
1289 Log Strength: 1.0000
1290 Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1291 Option Spec: PickJug(?robot:robot, ?jug:jug),1292 EndogenousProcess-PlaceJugInMachine:
1293 Parameters: [?robot:robot, ?jug:jug, ?machine:coffee_machine]
1294 Conditions at start: [Holding(?robot:robot, ?jug:jug), NotAboveCup(?
1295 robot:robot, ?jug:jug)]
1296 Conditions overall: []
1297 Conditions at end: []

```

1296 Add Effects: [HandEmpty(?robot:robot), JugInMachine(?jug:jug, ?
1297 machine:coffee_machine)]
1298 Delete Effects: [Holding(?robot:robot, ?jug:jug)]
1299 Ignore Effects: []
1300 Log Strength: 1.0000
1301 Delay Distribution: DiscreteGaussianDelay(4.0000, 0.1000)
1302 Option Spec: PlaceJugInMachine(?robot:robot, ?jug:jug, ?machine:
1303 coffee_machine),
1304

```

```

1304 EndogenousProcess-PourFromCup:
1305 Parameters: [?robot:robot, ?jug:jug, ?to_cup:cup, ?from_cup:cup]
1306 Conditions at start: [Holding(?robot:robot, ?jug:jug), JugAboveCup(??
1307 jug:jug, ?from_cup:cup)]
1308 Conditions overall: []
1309 Conditions at end: []
1310 Add Effects: [JugAboveCup(?jug:jug, ?to_cup:cup)]
1311 Delete Effects: [JugAboveCup(?jug:jug, ?from_cup:cup), NotAboveCup(??
1312 robot:robot, ?jug:jug)]
1313 Ignore Effects: [JugAboveCup, NotAboveCup]
1314 Log Strength: 1.0000
1315 Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
1316 Option Spec: Pour(?robot:robot, ?jug:jug, ?to_cup:cup),
1317

```

```

1317 EndogenousProcess-PourFromNotAboveCup:
1318 Parameters: [?robot:robot, ?jug:jug, ?cup:cup]
1319 Conditions at start: [Holding(?robot:robot, ?jug:jug), NotAboveCup(??
1320 robot:robot, ?jug:jug)]
1321 Conditions overall: []
1322 Conditions at end: []
1323 Add Effects: [JugAboveCup(?jug:jug, ?cup:cup)]
1324 Delete Effects: [NotAboveCup(?robot:robot, ?jug:jug)]
1325 Ignore Effects: []
1326 Log Strength: 1.0000
1327 Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
1328 Option Spec: Pour(?robot:robot, ?jug:jug, ?cup:cup),
1329

```

```

1329 EndogenousProcess-TurnMachineOn:
1330 Parameters: [?robot:robot, ?jug:jug, ?machine:coffee_machine]
1331 Conditions at start: [HandEmpty(?robot:robot), JugInMachine(?jug:jug,
1332 ?machine:coffee_machine)]
1333 Conditions overall: []
1334 Conditions at end: []
1335 Add Effects: [MachineOn(?machine:coffee_machine)]
1336 Delete Effects: []
1337 Ignore Effects: []
1338 Log Strength: 1.0000
1339 Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
1340 Option Spec: TurnMachineOn(?robot:robot, ?machine:coffee_machine)
1341

```

Additional predicates. {JugFilled}

B.2 GROW

Train/Test split. In the training tasks, the agent must grow plants in two pots. For each pot, at least one jug of a matching color is available, with a maximum of two jugs present in the environment overall. The test tasks increase in complexity, requiring the agent to grow plants in three pots, again with at least one matching jug available for each and a maximum of two jugs in total.

Goal predicates. {Grown}

```

1350
1351 Initial predicates and endogenous processes. {NotAboveCup, JugOnTable, Holding}
1352
1353 EndogenousProcess-NoOp:
1354     Parameters: [?robot:robot]
1355     Conditions at start: []
1356     Conditions overall: []
1357     Conditions at end: []
1358     Add Effects: []
1359     Delete Effects: []
1360     Ignore Effects: []
1361     Log Strength: 1.0000
1362     Delay Distribution: DiscreteGaussianDelay(1.0000, 0.1000)
1363     Option Spec: NoOp(?robot:robot),
1364
1365 EndogenousProcess-PickJugFromTable:
1366     Parameters: [?robot:robot, ?jug:jug]
1367     Conditions at start: [HandEmpty(?robot:robot), JugOnTable(?jug:jug)]
1368     Conditions overall: []
1369     Conditions at end: []
1370     Add Effects: [Holding(?robot:robot, ?jug:jug)]
1371     Delete Effects: [HandEmpty(?robot:robot), JugOnTable(?jug:jug)]
1372     Ignore Effects: []
1373     Log Strength: 1.0000
1374     Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1375     Option Spec: PickJug(?robot:robot, ?jug:jug),
1376
1377 EndogenousProcess-PlaceJugOnTable:
1378     Parameters: [?robot:robot, ?jug:jug, ?cup:cup]
1379     Conditions at start: [Holding(?robot:robot, ?jug:jug), JugAboveCup(?jug:jug, ?cup:cup)]
1380     Conditions overall: []
1381     Conditions at end: []
1382     Add Effects: [HandEmpty(?robot:robot), JugOnTable(?jug:jug),
1383     NotAboveCup(?robot:robot, ?jug:jug)]
1384     Delete Effects: [Holding(?robot:robot, ?jug:jug), JugAboveCup(?jug:jug, ?cup:cup)]
1385     Ignore Effects: [HandEmpty, Holding, JugAboveCup, JugOnTable,
1386     NotAboveCup]
1387     Log Strength: 1.0000
1388     Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1389     Option Spec: Place(?robot:robot, ?jug:jug),
1390
1391 EndogenousProcess-PourFromAboveCup:
1392     Parameters: [?robot:robot, ?jug:jug, ?from_cup:cup, ?to_cup:cup]
1393     Conditions at start: [Holding(?robot:robot, ?jug:jug), JugAboveCup(?jug:jug, ?from_cup:cup)]
1394     Conditions overall: []
1395     Conditions at end: []
1396     Add Effects: [JugAboveCup(?jug:jug, ?to_cup:cup)]
1397     Delete Effects: [JugAboveCup(?jug:jug, ?from_cup:cup)]
1398     Ignore Effects: [JugAboveCup, NotAboveCup]
1399     Log Strength: 1.0000
1400     Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
1401     Option Spec: Pour(?robot:robot, ?jug:jug, ?to_cup:cup),
1402
1403 EndogenousProcess-PourFromNotAboveCup:
1404     Parameters: [?robot:robot, ?jug:jug, ?cup:cup]
1405     Conditions at start: [Holding(?robot:robot, ?jug:jug), NotAboveCup(?robot:robot, ?jug:jug)]
1406     Conditions overall: []
1407     Conditions at end: []
1408     Add Effects: [JugAboveCup(?jug:jug, ?cup:cup)]
1409     Delete Effects: [NotAboveCup(?robot:robot, ?jug:jug)]

```

```

1404     Ignore Effects: [JugAboveCup, NotAboveCup]
1405     Log Strength: 1.0000
1406     Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
1407     Option Spec: Pour(?robot:robot, ?jug:jug, ?cup:cup)
1408

```

1409 **Additional predicates.** {SameColor}

1410 **B.3 BOIL**

1411 **Train/Test split.** Training tasks require the agent to boil a single jug of water. The evaluation
 1412 includes tasks that involve boiling either one or two jugs.

1413 **Goal predicates.** {HumanHappy}

1414 **Initial predicates and endogenous processes.** { FaucetOn, FaucetOff, HumanHappy,
 1415 JugAtBurner, Holding, JugAtFaucet, NoJugAtBurner, BurnerOff, HandEmpty,
 1416 BurnerOn, NoJugAtFaucet, JugNotAtBurnerOrFaucet}

```

1417 EndogenousProcess-NoOp:
1418     Parameters: [?robot:robot]
1419     Conditions at start: []
1420     Conditions overall: []
1421     Conditions at end: []
1422     Add Effects: []
1423     Delete Effects: []
1424     Ignore Effects: []
1425     Log Strength: 1.0000
1426     Delay Distribution: ConstantDelay(1.0000)
1427     Option Spec: NoOp(?robot:robot),
1428

```

```

1429 EndogenousProcess-PickJugFromBurner:
1430     Parameters: [?robot:robot, ?jug:jug, ?burner:burner]
1431     Conditions at start: [HandEmpty(?robot:robot), JugAtBurner(?jug:jug,
1432     ?burner:burner)]
1433     Conditions overall: []
1434     Conditions at end: []
1435     Add Effects: [Holding(?robot:robot, ?jug:jug), NoJugAtBurner(?burner:
1436     ?burner)]
1437     Delete Effects: [HandEmpty(?robot:robot), JugAtBurner(?jug:jug, ?
1438     ?burner:burner)]
1439     Ignore Effects: []
1440     Log Strength: 1.0000
1441     Delay Distribution: DiscreteGaussianDelay(4.0000, 0.1000)
1442     Option Spec: PickJug(?robot:robot, ?jug:jug),
1443

```

```

1444 EndogenousProcess-PickJugFromFaucet:
1445     Parameters: [?robot:robot, ?jug:jug, ?faucet:faucet]
1446     Conditions at start: [HandEmpty(?robot:robot), JugAtFaucet(?jug:jug,
1447     ?faucet:faucet)]
1448     Conditions overall: []
1449     Conditions at end: []
1450     Add Effects: [Holding(?robot:robot, ?jug:jug), NoJugAtFaucet(?faucet:
1451     ?faucet)]
1452     Delete Effects: [HandEmpty(?robot:robot), JugAtFaucet(?jug:jug, ?
1453     ?faucet:faucet)]
1454     Ignore Effects: []
1455     Log Strength: 1.0000
1456     Delay Distribution: DiscreteGaussianDelay(4.0000, 0.1000)
1457     Option Spec: PickJug(?robot:robot, ?jug:jug),
1458

```

```

1458
1459 EndogenousProcess-PickJugFromOutsideFaucetAndBurner:
1460     Parameters: [?robot:robot, ?jug:jug]
1461     Conditions at start: [HandEmpty(?robot:robot), JugNotAtBurnerOrFaucet
1462     (?jug:jug)]
1463     Conditions overall: []
1464     Conditions at end: []
1465     Add Effects: [Holding(?robot:robot, ?jug:jug)]
1466     Delete Effects: [HandEmpty(?robot:robot), JugNotAtBurnerOrFaucet (?jug
1467     :jug)]
1468     Ignore Effects: []
1469     Log Strength: 1.0000
1470     Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1471     Option Spec: PickJug(?robot:robot, ?jug:jug),
1472
1473 EndogenousProcess-PlaceOnBurner:
1474     Parameters: [?robot:robot, ?jug:jug, ?burner:burner]
1475     Conditions at start: [Holding(?robot:robot, ?jug:jug), NoJugAtBurner
1476     (?burner:burner)]
1477     Conditions overall: []
1478     Conditions at end: []
1479     Add Effects: [HandEmpty(?robot:robot), JugAtBurner(?jug:jug, ?burner:
1480     burner)]
1481     Delete Effects: [Holding(?robot:robot, ?jug:jug), NoJugAtBurner(??
1482     burner:burner)]
1483     Ignore Effects: []
1484     Log Strength: 1.0000
1485     Delay Distribution: DiscreteGaussianDelay(5.0000, 0.1000)
1486     Option Spec: PlaceOnBurner(?robot:robot, ?burner:burner),
1487
1488 EndogenousProcess-PlaceOutsideFaucetAndBurner:
1489     Parameters: [?robot:robot, ?jug:jug]
1490     Conditions at start: [Holding(?robot:robot, ?jug:jug)]
1491     Conditions overall: []
1492     Conditions at end: []
1493     Add Effects: [HandEmpty(?robot:robot), JugNotAtBurnerOrFaucet (?jug:
1494     jug)]
1495     Delete Effects: [Holding(?robot:robot, ?jug:jug)]
1496     Ignore Effects: []
1497     Log Strength: 1.0000
1498     Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1499     Option Spec: PlaceOutsideBurnerAndFaucet(?robot:robot),
1500
1501 EndogenousProcess-PlaceUnderFaucet:
1502     Parameters: [?robot:robot, ?jug:jug, ?faucet:faucet]
1503     Conditions at start: [Holding(?robot:robot, ?jug:jug), NoJugAtFaucet
1504     (?faucet:faucet)]
1505     Conditions overall: []
1506     Conditions at end: []
1507     Add Effects: [HandEmpty(?robot:robot), JugAtFaucet (?jug:jug, ?faucet:
1508     faucet)]
1509     Delete Effects: [Holding(?robot:robot, ?jug:jug), NoJugAtFaucet (??
1510     faucet:faucet)]
1511     Ignore Effects: []
1512     Log Strength: 1.0000
1513     Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1514     Option Spec: PlaceUnderFaucet(?robot:robot, ?faucet:faucet),
1515
1516 EndogenousProcess-SwitchBurnerOff:
1517     Parameters: [?robot:robot, ?burner:burner]
1518     Conditions at start: [BurnerOn(?burner:burner), HandEmpty(?robot:
1519     robot)]
1520     Conditions overall: []

```

```

1512     Conditions at end: []
1513     Add Effects: [BurnerOff(?burner:burner)]
1514     Delete Effects: [BurnerOn(?burner:burner)]
1515     Ignore Effects: []
1516     Log Strength: 1.0000
1517     Delay Distribution: DiscreteGaussianDelay(1.0000, 0.1000)
1518     Option Spec: SwitchBurnerOff(?robot:robot, ?burner:burner),
1519

```

```

1520 EndogenousProcess-SwitchBurnerOn:
1521     Parameters: [?robot:robot, ?burner:burner]
1522     Conditions at start: [BurnerOff(?burner:burner), HandEmpty(?robot:
1523     robot)]
1524     Conditions overall: []
1525     Conditions at end: []
1526     Add Effects: [BurnerOn(?burner:burner)]
1527     Delete Effects: [BurnerOff(?burner:burner)]
1528     Ignore Effects: []
1529     Log Strength: 1.0000
1530     Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1531     Option Spec: SwitchBurnerOn(?robot:robot, ?burner:burner),
1532

```

```

1533 EndogenousProcess-SwitchFaucetOff:
1534     Parameters: [?robot:robot, ?faucet:faucet]
1535     Conditions at start: [FaucetOn(?faucet:faucet), HandEmpty(?robot:
1536     robot)]
1537     Conditions overall: []
1538     Conditions at end: []
1539     Add Effects: [FaucetOff(?faucet:faucet)]
1540     Delete Effects: [FaucetOn(?faucet:faucet)]
1541     Ignore Effects: []
1542     Log Strength: 1.0000
1543     Delay Distribution: DiscreteGaussianDelay(1.0000, 0.1000)
1544     Option Spec: SwitchFaucetOff(?robot:robot, ?faucet:faucet),
1545

```

```

1546 EndogenousProcess-SwitchFaucetOn:
1547     Parameters: [?robot:robot, ?faucet:faucet]
1548     Conditions at start: [FaucetOff(?faucet:faucet), HandEmpty(?robot:
1549     robot)]
1550     Conditions overall: []
1551     Conditions at end: []
1552     Add Effects: [FaucetOn(?faucet:faucet)]
1553     Delete Effects: [FaucetOff(?faucet:faucet)]
1554     Ignore Effects: []
1555     Log Strength: 1.0000
1556     Delay Distribution: DiscreteGaussianDelay(1.0000, 0.1000)
1557     Option Spec: SwitchFaucetOn(?robot:robot, ?faucet:faucet)
1558

```

1559 **Additional predicates.** NoWaterSpilled, WaterBoiled, JugFilled,
1560 NoJugAtFaucetOrAtFaucetAndFilled

1559 B.4 DOMINO

1560 **Train/Test split.** The training tasks takes place in a compact 3×2 grid, where the agent must
1561 arrange one movable domino to successfully topple a single target domino. The test tasks are more
1562 complex in three ways: the workspace is enlarged to a 4×3 grid, the number of movable dominoes
1563 is increased to two, and the goals may require toppling either one or two target dominoes.

1564
1565 **Goal predicates.** Toppled

```

1566 Initial predicates and endogenous processes. Upright, InFrontDirection,
1567 InitialBlock, MovableBlock, Toppled, AdjacentTo, DominoAtPos, Holding,
1568 DominoAtRot, HandEmpty, Tilting, PosClear
1569
1570 EndogenousProcess-NoOp:
1571   Parameters: [?robot:robot]
1572   Conditions at start: []
1573   Conditions overall: []
1574   Conditions at end: []
1575   Add Effects: []
1576   Delete Effects: []
1577   Ignore Effects: [AdjacentTo, DominoAtPos, DominoAtRot, PosClear]
1578   Log Strength: 1.0000
1579   Delay Distribution: ConstantDelay(1.0000)
1580   Option Spec: NoOp(?robot:robot),
1581
1582 EndogenousProcess-PickDomino:
1583   Parameters: [?robot:robot, ?domino:domino, ?pos:loc, ?rot:angle]
1584   Conditions at start: [DominoAtPos(?domino:domino, ?pos:loc),
1585   DominoAtRot(?domino:domino, ?rot:angle), HandEmpty(?robot:robot),
1586   MovableBlock(?domino:domino), Upright(?domino:domino)]
1587   Conditions overall: []
1588   Conditions at end: []
1589   Add Effects: [Holding(?robot:robot, ?domino:domino), PosClear(?pos:
1590   loc)]
1591   Delete Effects: [DominoAtPos(?domino:domino, ?pos:loc), DominoAtRot(??
1592   domino:domino, ?rot:angle), HandEmpty(?robot:robot)]
1593   Ignore Effects: [DominoAtPos, DominoAtRot, PosClear, Tilting, Toppled
1594   , Upright]
1595   Log Strength: 1.0000
1596   Delay Distribution: DiscreteGaussianDelay(4.0000, 0.1000)
1597   Option Spec: Pick(?robot:robot, ?domino:domino),
1598
1599 EndogenousProcess-PlaceDomino:
1600   Parameters: [?robot:robot, ?domino1:domino, ?domino2:domino, ?pos1:
1601   loc, ?rot:angle]
1602   Conditions at start: [AdjacentTo(?pos1:loc, ?domino2:domino), Holding
1603   (?robot:robot, ?domino1:domino), PosClear(?pos1:loc), Upright(??
1604   domino2:domino)]
1605   Conditions overall: []
1606   Conditions at end: []
1607   Add Effects: [DominoAtPos(?domino1:domino, ?pos1:loc), DominoAtRot(??
1608   domino1:domino, ?rot:angle), HandEmpty(?robot:robot)]
1609   Delete Effects: [Holding(?robot:robot, ?domino1:domino), PosClear(??
1610   pos1:loc)]
1611   Ignore Effects: [AdjacentTo, DominoAtPos, DominoAtRot, PosClear,
1612   Tilting]
1613   Log Strength: 1.0000
1614   Delay Distribution: DiscreteGaussianDelay(3.0000, 0.1000)
1615   Option Spec: Place(?robot:robot, ?domino1:domino, ?domino2:domino, ??
1616   pos1:loc, ?rot:angle),
1617
1618 EndogenousProcess-PushStartBlock:
1619   Parameters: [?robot:robot, ?domino:domino]
1620   Conditions at start: [HandEmpty(?robot:robot), InitialBlock(?domino:
1621   domino), Upright(?domino:domino)]
1622   Conditions overall: []
1623   Conditions at end: []
1624   Add Effects: [Tilting(?domino:domino)]
1625   Delete Effects: [Upright(?domino:domino)]
1626   Ignore Effects: [AdjacentTo, DominoAtPos, DominoAtRot, PosClear]
1627   Log Strength: 1.0000
1628   Delay Distribution: DiscreteGaussianDelay(1.0000, 0.1000)
1629   Option Spec: Push(?robot:robot, ?domino:domino)

```

1620 **Additional predicates.** {NotHeavy}

1621
1622 **B.5 FAN**

1624 **Train/Test split.** Training tasks are conducted on a small 3×3 grid containing a single wall obstacle. In contrast, test tasks feature a larger 6×4 grid and more intricate mazes constructed with either
1625 two or three walls.

1627

1628 **Goal predicates.** {BallAtLoc}

1629

1630 **Initial predicates and endogenous processes.** {SideOf, BallAtLoc, ClearLoc, FanOn,
1631 FanOff }

1632 EndogenousProcess-NoOp:
1633 Parameters: [?robot:robot]
1634 Conditions at start: []
1635 Conditions overall: []
1636 Conditions at end: []
1637 Add Effects: []
1638 Delete Effects: []
1639 Ignore Effects: []
1640 Log Strength: 1.0000
1641 Delay Distribution: ConstantDelay(1.0000)
1642 Option Spec: NoOp(?robot:robot),

1643 EndogenousProcess-TurnFanOff:
1644 Parameters: [?robot:robot, ?fan:fan]
1645 Conditions at start: [FanOn(?fan:fan)]
1646 Conditions overall: []
1647 Conditions at end: []
1648 Add Effects: [FanOff(?fan:fan)]
1649 Delete Effects: [FanOn(?fan:fan)]
1650 Ignore Effects: []
1651 Log Strength: 1.0000
1652 Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
1653 Option Spec: SwitchOff(?robot:robot, ?fan:fan),

1653 EndogenousProcess-TurnFanOn:
1654 Parameters: [?robot:robot, ?fan:fan]
1655 Conditions at start: [FanOff(?fan:fan)]
1656 Conditions overall: []
1657 Conditions at end: []
1658 Add Effects: [FanOn(?fan:fan)]
1659 Delete Effects: [FanOff(?fan:fan)]
1660 Ignore Effects: []
1661 Log Strength: 1.0000
1662 Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
1663 Option Spec: SwitchOn(?robot:robot, ?fan:fan)

1663

1664 **Additional predicates.** {FanFacingSide, OppositeFan}

1665

1666 **C ADDITIONAL EXPERIMENT DETAILS**

1667

1668 **C.1 LEARNED CAUSAL PROCESSES**

1669

1670 We show example learned predicates and causal processes in each domain.

1671

1672 **C.1.1 COFFEE**

1673

Learned predicates and processes. {JugFilled}

```

1674
1675 EndogenousProcess-NoOp:
1676     Parameters: [?robot:robot]
1677     Conditions at start: []
1678     Conditions overall: []
1679     Conditions at end: []
1680     Add Effects: []
1681     Delete Effects: []
1682     Ignore Effects: [JugAboveCup, NotAboveCup]
1683     Log Strength: -0.0113
1684     Delay Distribution: ConstantDelay(-0.0115)
1685     Option Spec: NoOp(?robot:robot)
1686
1687 EndogenousProcess-PickJugFromMachine:
1688     Parameters: [?robot:robot, ?jug:jug, ?machine:coffee_machine]
1689     Conditions at start: [HandEmpty(?robot:robot), JugInMachine(?jug:jug,
1690     ?machine:coffee_machine)]
1691     Conditions overall: []
1692     Conditions at end: []
1693     Add Effects: [Holding(?robot:robot, ?jug:jug)]
1694     Delete Effects: [HandEmpty(?robot:robot), JugInMachine(?jug:jug, ?
1695     machine:coffee_machine)]
1696     Ignore Effects: []
1697     Log Strength: 4.8335
1698     Delay Distribution: DiscreteGaussianDelay(13.8455, 5.3512)
1699     Option Spec: PickJug(?robot:robot, ?jug:jug)
1700
1701 EndogenousProcess-PickJugFromTable:
1702     Parameters: [?robot:robot, ?jug:jug]
1703     Conditions at start: [HandEmpty(?robot:robot), OnTable(?jug:jug)]
1704     Conditions overall: []
1705     Conditions at end: []
1706     Add Effects: [Holding(?robot:robot, ?jug:jug)]
1707     Delete Effects: [HandEmpty(?robot:robot), OnTable(?jug:jug)]
1708     Ignore Effects: []
1709     Log Strength: 1.5300
1710     Delay Distribution: DiscreteGaussianDelay(23.8392, 6.6450)
1711     Option Spec: PickJug(?robot:robot, ?jug:jug)
1712
1713 EndogenousProcess-PlaceJugInMachine:
1714     Parameters: [?robot:robot, ?jug:jug, ?machine:coffee_machine]
1715     Conditions at start: [Holding(?robot:robot, ?jug:jug), NotAboveCup(??
1716     robot:robot, ?jug:jug)]
1717     Conditions overall: []
1718     Conditions at end: []
1719     Add Effects: [HandEmpty(?robot:robot), JugInMachine(?jug:jug, ?
1720     machine:coffee_machine)]
1721     Delete Effects: [Holding(?robot:robot, ?jug:jug)]
1722     Ignore Effects: []
1723     Log Strength: 1.6979
1724     Delay Distribution: DiscreteGaussianDelay(20.0003, 6.5394)
1725     Option Spec: PlaceJugInMachine(?robot:robot, ?jug:jug, ?machine:?
1726     coffee_machine)
1727
1728 EndogenousProcess-PourFromCup:
1729     Parameters: [?robot:robot, ?jug:jug, ?to_cup:cup, ?from_cup:cup]
1730     Conditions at start: [Holding(?robot:robot, ?jug:jug), JugAboveCup(??
1731     jug:jug, ?from_cup:cup)]
1732     Conditions overall: []
1733     Conditions at end: []
1734     Add Effects: [JugAboveCup(?jug:jug, ?to_cup:cup)]
1735     Delete Effects: [JugAboveCup(?jug:jug, ?from_cup:cup), NotAboveCup(??
1736     robot:robot, ?jug:jug)]

```

```

1728     Ignore Effects: [JugAboveCup, NotAboveCup]
1729     Log Strength: 0.0012
1730     Delay Distribution: DiscreteGaussianDelay(1.0125, 1.0112)
1731     Option Spec: Pour(?robot:robot, ?jug:jug, ?to_cup:cup)
1732
1733 EndogenousProcess-PourFromNotAboveCup:
1734     Parameters: [?robot:robot, ?jug:jug, ?cup:cup]
1735     Conditions at start: [Holding(?robot:robot, ?jug:jug), NotAboveCup(?robot:robot, ?jug:jug)]
1736     Conditions overall: []
1737     Conditions at end: []
1738     Add Effects: [JugAboveCup(?jug:jug, ?cup:cup)]
1739     Delete Effects: [NotAboveCup(?robot:robot, ?jug:jug)]
1740     Ignore Effects: []
1741     Log Strength: 1.7079
1742     Delay Distribution: DiscreteGaussianDelay(7.4837, 5.0596)
1743     Option Spec: Pour(?robot:robot, ?jug:jug, ?cup:cup)
1744
1745 EndogenousProcess-TurnMachineOn:
1746     Parameters: [?robot:robot, ?jug:jug, ?machine:coffee_machine]
1747     Conditions at start: [HandEmpty(?robot:robot), JugInMachine(?jug:jug, ?machine:coffee_machine)]
1748     Conditions overall: []
1749     Conditions at end: []
1750     Add Effects: [MachineOn(?machine:coffee_machine)]
1751     Delete Effects: []
1752     Ignore Effects: []
1753     Log Strength: 1.7298
1754     Delay Distribution: DiscreteGaussianDelay(18.3430, 6.4795)
1755     Option Spec: TurnMachineOn(?robot:robot, ?machine:coffee_machine)
1756
1757 ExogenousProcess-Op3:
1758     Parameters: [?x0:coffee_machine, ?x2:jug]
1759     Conditions at start: [JugInMachine(?x2:jug, ?x0:coffee_machine),
1760     MachineOn(?x0:coffee_machine)]
1761     Conditions overall: [JugInMachine(?x2:jug, ?x0:coffee_machine),
1762     MachineOn(?x0:coffee_machine)]
1763     Conditions at end: []
1764     Add Effects: [JugFilled(?x2:jug)]
1765     Delete Effects: []
1766     Log Strength: 1.7168
1767     Delay Distribution: DiscreteGaussianDelay(17.3098, 6.4991)
1768
1769 ExogenousProcess-Op5:
1770     Parameters: [?x1:cup, ?x2:jug, ?x3:robot]
1771     Conditions at start: [Holding(?x3:robot, ?x2:jug), JugAboveCup(?x2:jug, ?x1:cup),
1772     JugFilled(?x2:jug)]
1773     Conditions overall: [Holding(?x3:robot, ?x2:jug), JugAboveCup(?x2:jug, ?x1:cup),
1774     JugFilled(?x2:jug)]
1775     Conditions at end: []
1776     Add Effects: [CupFilled(?x1:cup)]
1777     Delete Effects: []
1778     Log Strength: 1.7250
1779     Delay Distribution: DiscreteGaussianDelay(4.5577, 1.8173)

```

C.1.2 GROW

Learned predicates and processes. {ColorMatches}

```

1780 EndogenousProcess-NoOp:
1781     Parameters: [?robot:robot]
1782     Conditions at start: []

```

```

1782     Conditions overall: []
1783     Conditions at end: []
1784     Add Effects: []
1785     Delete Effects: []
1786     Ignore Effects: []
1787     Log Strength: -0.0113
1788     Delay Distribution: DiscreteGaussianDelay(25.6750, 6.9284)
1789     Option Spec: NoOp(?robot:robot)
1790
1791 EndogenousProcess-PickJugFromTable:
1792     Parameters: [?robot:robot, ?jug:jug]
1793     Conditions at start: [HandEmpty(?robot:robot), JugOnTable(?jug:jug)]
1794     Conditions overall: []
1795     Conditions at end: []
1796     Add Effects: [Holding(?robot:robot, ?jug:jug)]
1797     Delete Effects: [HandEmpty(?robot:robot), JugOnTable(?jug:jug)]
1798     Ignore Effects: []
1799     Log Strength: 2.3222
1800     Delay Distribution: DiscreteGaussianDelay(32.9836, 6.9427)
1801     Option Spec: PickJug(?robot:robot, ?jug:jug)
1802
1803 EndogenousProcess-PlaceJugOnTable:
1804     Parameters: [?robot:robot, ?jug:jug, ?cup:cup]
1805     Conditions at start: [Holding(?robot:robot, ?jug:jug), JugAboveCup(?jug:jug, ?cup:cup)]
1806     Conditions overall: []
1807     Conditions at end: []
1808     Add Effects: [HandEmpty(?robot:robot), JugOnTable(?jug:jug),
1809     NotAboveCup(?robot:robot, ?jug:jug)]
1810     Delete Effects: [Holding(?robot:robot, ?jug:jug), JugAboveCup(?jug:jug, ?cup:cup)]
1811     Ignore Effects: [HandEmpty, Holding, JugAboveCup, JugOnTable,
1812     NotAboveCup]
1813     Log Strength: 1.9316
1814     Delay Distribution: DiscreteGaussianDelay(24.9979, 6.6815)
1815     Option Spec: Place(?robot:robot, ?jug:jug)
1816
1817 EndogenousProcess-PourFromAboveCup:
1818     Parameters: [?robot:robot, ?jug:jug, ?from_cup:cup, ?to_cup:cup]
1819     Conditions at start: [Holding(?robot:robot, ?jug:jug), JugAboveCup(?jug:jug, ?from_cup:cup)]
1820     Conditions overall: []
1821     Conditions at end: []
1822     Add Effects: [JugAboveCup(?jug:jug, ?to_cup:cup)]
1823     Delete Effects: [JugAboveCup(?jug:jug, ?from_cup:cup)]
1824     Ignore Effects: [JugAboveCup, NotAboveCup]
1825     Log Strength: -0.0126
1826     Delay Distribution: DiscreteGaussianDelay(1.0035, 1.0031)
1827     Option Spec: Pour(?robot:robot, ?jug:jug, ?to_cup:cup)
1828
1829 EndogenousProcess-PourFromNotAboveCup:
1830     Parameters: [?robot:robot, ?jug:jug, ?cup:cup]
1831     Conditions at start: [Holding(?robot:robot, ?jug:jug), NotAboveCup(?robot:robot, ?jug:jug)]
1832     Conditions overall: []
1833     Conditions at end: []
1834     Add Effects: [JugAboveCup(?jug:jug, ?cup:cup)]
1835     Delete Effects: [NotAboveCup(?robot:robot, ?jug:jug)]
1836     Ignore Effects: [JugAboveCup, NotAboveCup]
1837     Log Strength: 2.2282
1838     Delay Distribution: DiscreteGaussianDelay(28.9585, 6.9970)
1839     Option Spec: Pour(?robot:robot, ?jug:jug, ?cup:cup)

```

```

1836
1837 ExogenousProcess-Op0:
1838     Parameters: [?x1:cup, ?x3:jug, ?x4:robot]
1839     Conditions at start: [ColorMatches(?x3:jug, ?x1:cup), CupOnTable(?x1:
1840     cup), Holding(?x4:robot, ?x3:jug), JugAboveCup(?x3:jug, ?x1:cup)]
1841     Conditions overall: [ColorMatches(?x3:jug, ?x1:cup), CupOnTable(?x1:
1842     cup), Holding(?x4:robot, ?x3:jug), JugAboveCup(?x3:jug, ?x1:cup)]
1843     Conditions at end: []
1844     Add Effects: [Grown(?x1:cup)]
1845     Delete Effects: []
1846     Log Strength: 1.2238
1847     Delay Distribution: DiscreteGaussianDelay(30.6220, 6.7903)
1848
1849
1850 C.1.3 BOIL
1851
1852 Learned predicates and processes. {JugIsHot, JugIsFull, NotSpilling}
1853
1854 EndogenousProcess-NoOp:
1855     Parameters: [?robot:robot]
1856     Conditions at start: []
1857     Conditions overall: []
1858     Conditions at end: []
1859     Add Effects: []
1860     Delete Effects: []
1861     Ignore Effects: []
1862     Log Strength: -0.0113
1863     Delay Distribution: ConstantDelay(-0.0115)
1864     Option Spec: NoOp(?robot:robot)
1865
1866
1867 EndogenousProcess-PickJugFromBurner:
1868     Parameters: [?robot:robot, ?jug:jug, ?burner:burner]
1869     Conditions at start: [HandEmpty(?robot:robot), JugAtBurner(?jug:jug,
1870     ?burner:burner)]
1871     Conditions overall: []
1872     Conditions at end: []
1873     Add Effects: [Holding(?robot:robot, ?jug:jug), NoJugAtBurner(?burner:
1874     burner)]
1875     Delete Effects: [HandEmpty(?robot:robot), JugAtBurner(?jug:jug, ?
1876     burner:burner)]
1877     Ignore Effects: []
1878     Log Strength: -0.0043
1879     Delay Distribution: DiscreteGaussianDelay(1.0085, 1.0069)
1880     Option Spec: PickJug(?robot:robot, ?jug:jug)
1881
1882
1883 EndogenousProcess-PickJugFromFaucet:
1884     Parameters: [?robot:robot, ?jug:jug, ?faucet:faucet]
1885     Conditions at start: [HandEmpty(?robot:robot), JugAtFaucet(?jug:jug,
1886     ?faucet:faucet)]
1887     Conditions overall: []
1888     Conditions at end: []
1889     Add Effects: [Holding(?robot:robot, ?jug:jug), NoJugAtFaucet(?faucet:
1890     faucet)]
1891     Delete Effects: [HandEmpty(?robot:robot), JugAtFaucet(?jug:jug, ?
1892     faucet:faucet)]
1893     Ignore Effects: []
1894     Log Strength: 1.9823
1895     Delay Distribution: DiscreteGaussianDelay(23.5668, 6.6476)
1896     Option Spec: PickJug(?robot:robot, ?jug:jug)
1897
1898
1899 EndogenousProcess-PickJugFromOutsideFaucetAndBurner:
1900     Parameters: [?robot:robot, ?jug:jug]
1901     Conditions at start: [HandEmpty(?robot:robot), JugNotAtBurnerOrFaucet
1902     (?jug:jug)]

```

```

1890     Conditions overall: []
1891     Conditions at end: []
1892     Add Effects: [Holding(?robot:robot, ?jug:jug)]
1893     Delete Effects: [HandEmpty(?robot:robot), JugNotAtBurnerOrFaucet(?jug
1894     :jug)]
1895     Ignore Effects: []
1896     Log Strength: 1.1899
1897     Delay Distribution: DiscreteGaussianDelay(43.4278, 6.8760)
1898     Option Spec: PickJug(?robot:robot, ?jug:jug)
1899
1900 EndogenousProcess-PlaceOnBurner:
1901     Parameters: [?robot:robot, ?jug:jug, ?burner:burner]
1902     Conditions at start: [Holding(?robot:robot, ?jug:jug), NoJugAtBurner
1903     (?burner:burner)]
1904     Conditions overall: []
1905     Conditions at end: []
1906     Add Effects: [HandEmpty(?robot:robot), JugAtBurner(?jug:jug, ?burner:
1907     :burner)]
1908     Delete Effects: [Holding(?robot:robot, ?jug:jug), NoJugAtBurner(??
1909     :burner:burner)]
1910     Ignore Effects: []
1911     Log Strength: 2.1507
1912     Delay Distribution: DiscreteGaussianDelay(21.9568, 6.6072)
1913     Option Spec: PlaceOnBurner(?robot:robot, ?burner:burner)
1914
1915 EndogenousProcess-PlaceOutsideFaucetAndBurner:
1916     Parameters: [?robot:robot, ?jug:jug]
1917     Conditions at start: [Holding(?robot:robot, ?jug:jug)]
1918     Conditions overall: []
1919     Conditions at end: []
1920     Add Effects: [HandEmpty(?robot:robot), JugNotAtBurnerOrFaucet(?jug:
1921     :jug)]
1922     Delete Effects: [Holding(?robot:robot, ?jug:jug)]
1923     Ignore Effects: []
1924     Log Strength: -0.0025
1925     Delay Distribution: DiscreteGaussianDelay(0.9866, 0.9832)
1926     Option Spec: PlaceOutsideBurnerAndFaucet(?robot:robot)
1927
1928 EndogenousProcess-PlaceUnderFaucet:
1929     Parameters: [?robot:robot, ?jug:jug, ?faucet:faucet]
1930     Conditions at start: [Holding(?robot:robot, ?jug:jug), NoJugAtFaucet
1931     (?faucet:faucet)]
1932     Conditions overall: []
1933     Conditions at end: []
1934     Add Effects: [HandEmpty(?robot:robot), JugAtFaucet(?jug:jug, ?faucet:
1935     :faucet)]
1936     Delete Effects: [Holding(?robot:robot, ?jug:jug), NoJugAtFaucet(??
1937     :faucet:faucet)]
1938     Ignore Effects: []
1939     Log Strength: 1.9426
1940     Delay Distribution: DiscreteGaussianDelay(41.1660, 6.8798)
1941     Option Spec: PlaceUnderFaucet(?robot:robot, ?faucet:faucet)
1942
1943 EndogenousProcess-SwitchBurnerOff:
1944     Parameters: [?robot:robot, ?burner:burner]
1945     Conditions at start: [BurnerOn(?burner:burner), HandEmpty(?robot:
1946     :robot)]
1947     Conditions overall: []
1948     Conditions at end: []
1949     Add Effects: [BurnerOff(?burner:burner)]
1950     Delete Effects: [BurnerOn(?burner:burner)]
1951     Ignore Effects: []
1952     Log Strength: 5.6200

```

```

1944     Delay Distribution: DiscreteGaussianDelay(9.8894, 4.0791)
1945     Option Spec: SwitchBurnerOff(?robot:robot, ?burner:burner)
1946

1947 EndogenousProcess-SwitchBurnerOn:
1948     Parameters: [?robot:robot, ?burner:burner]
1949     Conditions at start: [BurnerOff(?burner:burner), HandEmpty(?robot:
1950     robot)]
1951     Conditions overall: []
1952     Conditions at end: []
1953     Add Effects: [BurnerOn(?burner:burner)]
1954     Delete Effects: [BurnerOff(?burner:burner)]
1955     Ignore Effects: []
1956     Log Strength: 1.9554
1957     Delay Distribution: DiscreteGaussianDelay(32.0574, 6.7500)
1958     Option Spec: SwitchBurnerOn(?robot:robot, ?burner:burner)

1959 EndogenousProcess-SwitchFaucetOff:
1960     Parameters: [?robot:robot, ?faucet:faucet]
1961     Conditions at start: [FaucetOn(?faucet:faucet), HandEmpty(?robot:
1962     robot)]
1963     Conditions overall: []
1964     Conditions at end: []
1965     Add Effects: [FaucetOff(?faucet:faucet)]
1966     Delete Effects: [FaucetOn(?faucet:faucet)]
1967     Ignore Effects: []
1968     Log Strength: 1.4501
1969     Delay Distribution: DiscreteGaussianDelay(27.5714, 6.7946)
1970     Option Spec: SwitchFaucetOff(?robot:robot, ?faucet:faucet)

1971 EndogenousProcess-SwitchFaucetOn:
1972     Parameters: [?robot:robot, ?faucet:faucet]
1973     Conditions at start: [FaucetOff(?faucet:faucet), HandEmpty(?robot:
1974     robot)]
1975     Conditions overall: []
1976     Conditions at end: []
1977     Add Effects: [FaucetOn(?faucet:faucet)]
1978     Delete Effects: [FaucetOff(?faucet:faucet)]
1979     Ignore Effects: []
1980     Log Strength: 1.7156
1981     Delay Distribution: DiscreteGaussianDelay(35.2949, 6.7557)
1982     Option Spec: SwitchFaucetOn(?robot:robot, ?faucet:faucet)

1983 ExogenousProcess-Op0:
1984     Parameters: [?x1:faucet, ?x2:jug]
1985     Conditions at start: [FaucetOn(?x1:faucet), JugAtFaucet(?x2:jug, ?x1:
1986     faucet)]
1987     Conditions overall: [FaucetOn(?x1:faucet), JugAtFaucet(?x2:jug, ?x1:
1988     faucet)]
1989     Conditions at end: []
1990     Add Effects: [JugIsFull(?x2:jug)]
1991     Delete Effects: []
1992     Log Strength: 1.2791
1993     Delay Distribution: DiscreteGaussianDelay(33.1148, 6.7873)

1994 ExogenousProcess-Op1:
1995     Parameters: [?x0:burner, ?x2:jug]
1996     Conditions at start: [BurnerOn(?x0:burner), JugAtBurner(?x2:jug, ?x0:
1997     burner), JugIsFull(?x2:jug)]
1998     Conditions overall: [BurnerOn(?x0:burner), JugAtBurner(?x2:jug, ?x0:
1999     burner), JugIsFull(?x2:jug)]
2000     Conditions at end: []
2001     Add Effects: [JugIsHot(?x2:jug)]

```

```

1998     Delete Effects: []
1999     Log Strength: 1.9391
2000     Delay Distribution: DiscreteGaussianDelay(17.6401, 6.4452)
2001
2002     ExogenousProcess-Op2:
2003         Parameters: [?x0:burner, ?x1:faucet, ?x2:human, ?x3:jug, ?x4:robot]
2004         Conditions at start: [BurnerOff(?x0:burner), FaucetOff(?x1:faucet),
2005             HandEmpty(?x4:robot), JugIsFull(?x3:jug), JugIsHot(?x3:jug),
2006             NotSpilling(?x1:faucet)]
2007         Conditions overall: [BurnerOff(?x0:burner), FaucetOff(?x1:faucet),
2008             HandEmpty(?x4:robot), JugIsFull(?x3:jug), JugIsHot(?x3:jug),
2009             NotSpilling(?x1:faucet)]
2010         Conditions at end: []
2011         Add Effects: [HumanHappy(?x2:human, ?x3:jug, ?x0:burner)]
2012         Delete Effects: []
2013         Log Strength: 1.9474
2014         Delay Distribution: DiscreteGaussianDelay(5.5717, 4.1486)
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

```

```

ExogenousProcess-Op3:
Parameters: [?x1:faucet]
Conditions at start: [FaucetOn(?x1:faucet), NoJugAtFaucet(?x1:faucet),
, NotSpilling(?x1:faucet)]
Conditions overall: [FaucetOn(?x1:faucet), NoJugAtFaucet(?x1:faucet),
, NotSpilling(?x1:faucet)]
Conditions at end: []
Add Effects: []
Delete Effects: [NotSpilling(?x1:faucet)]
Log Strength: -0.0029
Delay Distribution: DiscreteGaussianDelay(1.0005, 1.0052)

```

C.1.4 DOMINO

Learned predicates and processes. {NOT-IsImmovable}

```

EndogenousProcess-NoOp:
Parameters: [?robot:robot]
Conditions at start: []
Conditions overall: []
Conditions at end: []
Add Effects: []
Delete Effects: []
Ignore Effects: [AdjacentTo, DominoAtPos, DominoAtRot, PosClear]
Log Strength: 1.0000
Delay Distribution: ConstantDelay(1.0000)
Option Spec: NoOp(?robot:robot)

```

```

EndogenousProcess-PickDomino:
Parameters: [?robot:robot, ?domino:domino, ?pos:loc, ?rot:angle]
Conditions at start: [DominoAtPos(?domino:domino, ?pos:loc),
DominoAtRot(?domino:domino, ?rot:angle), HandEmpty(?robot:robot),
MovableBlock(?domino:domino), Upright(?domino:domino)]
Conditions overall: []
Conditions at end: []
Add Effects: [Holding(?robot:robot, ?domino:domino), PosClear(?pos:
loc)]
Delete Effects: [DominoAtPos(?domino:domino, ?pos:loc), DominoAtRot(?domino:domino, ?rot:angle),
HandEmpty(?robot:robot)]
Ignore Effects: [DominoAtPos, DominoAtRot, PosClear, Tilting, Toppled,
, Upright]
Log Strength: 0.0000
Delay Distribution: DiscreteGaussianDelay(14.0000, 0.1000)
Option Spec: Pick(?robot:robot, ?domino:domino)

```

```

2052
2053 EndogenousProcess-PlaceDomino:
2054     Parameters: [?robot:robot, ?domino1:domino, ?domino2:domino, ?pos1:
2055         loc, ?rot:angle]
2056     Conditions at start: [AdjacentTo(?pos1:loc, ?domino2:domino), Holding
2057         (?robot:robot, ?domino1:domino), PosClear(?pos1:loc), Upright(??
2058         domino2:domino)]
2059     Conditions overall: []
2060     Conditions at end: []
2061     Add Effects: [DominoAtPos(?domino1:domino, ?pos1:loc), DominoAtRot(??
2062         domino1:domino, ?rot:angle), HandEmpty(?robot:robot)]
2063     Delete Effects: [Holding(?robot:robot, ?domino1:domino), PosClear(??
2064         pos1:loc)]
2065     Ignore Effects: [AdjacentTo, DominoAtPos, DominoAtRot, PosClear,
2066         Tilting]
2067     Log Strength: 0.0000
2068     Delay Distribution: DiscreteGaussianDelay(8.0000, 0.1000)
2069     Option Spec: Place(?robot:robot, ?domino1:domino, ?domino2:domino, ??
2070         pos1:loc, ?rot:angle)
2071
2072 EndogenousProcess-PushStartBlock:
2073     Parameters: [?robot:robot, ?domino:domino]
2074     Conditions at start: [HandEmpty(?robot:robot), InitialBlock(?domino:?
2075         domino), Upright(?domino:domino)]
2076     Conditions overall: []
2077     Conditions at end: []
2078     Add Effects: [Tilting(?domino:domino)]
2079     Delete Effects: [Upright(?domino:domino)]
2080     Ignore Effects: [AdjacentTo, DominoAtPos, DominoAtRot, PosClear]
2081     Log Strength: 0.0000
2082     Delay Distribution: DiscreteGaussianDelay(8.0000, 0.1000)
2083     Option Spec: Push(?robot:robot, ?domino:domino)
2084
2085 ExogenousProcess-Op0:
2086     Parameters: [?x1:domino, ?x2:domino, ?x11:direction]
2087     Conditions at start: [InFrontDirection(?x1:domino, ?x2:domino, ?x11:
2088         direction), NOT-IsImmovable(?x1:domino), NOT-IsImmovable(?x2:domino),
2089         Tilting(?x1:domino), Upright(?x2:domino)]
2090     Conditions overall: [InFrontDirection(?x1:domino, ?x2:domino, ?x11:
2091         direction), NOT-IsImmovable(?x1:domino), NOT-IsImmovable(?x2:domino),
2092         Tilting(?x1:domino), Upright(?x2:domino)]
2093     Conditions at end: []
2094     Add Effects: [Tilting(?x2:domino)]
2095     Delete Effects: []
2096     Log Strength: 0.0000
2097     Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
2098
2099 ExogenousProcess-Op1:
2100     Parameters: [?x1:domino, ?x2:domino, ?x11:direction]
2101     Conditions at start: [InFrontDirection(?x1:domino, ?x2:domino, ?x11:
2102         direction), NOT-IsImmovable(?x1:domino), NOT-IsImmovable(?x2:domino),
2103         Tilting(?x1:domino), Upright(?x2:domino)]
2104     Conditions overall: [InFrontDirection(?x1:domino, ?x2:domino, ?x11:
2105         direction), NOT-IsImmovable(?x1:domino), NOT-IsImmovable(?x2:domino),
2106         Tilting(?x1:domino), Upright(?x2:domino)]
2107     Conditions at end: []
2108     Add Effects: []
2109     Delete Effects: [Upright(?x2:domino)]
2110     Log Strength: 0.0000
2111     Delay Distribution: DiscreteGaussianDelay(2.0000, 0.1000)
2112
2113 ExogenousProcess-Op2:
2114     Parameters: [?x1:domino]

```

```

2106     Conditions at start: [Tilting(?x1:domino) ]
2107     Conditions overall: [Tilting(?x1:domino) ]
2108     Conditions at end: []
2109     Add Effects: []
2110     Delete Effects: [Tilting(?x1:domino) ]
2111     Log Strength: 0.0000
2112     Delay Distribution: DiscreteGaussianDelay(5.0000, 0.1000)

```

```

2113 ExogenousProcess-Op3:
2114     Parameters: [?x1:domino]
2115     Conditions at start: [Tilting(?x1:domino) ]
2116     Conditions overall: [Tilting(?x1:domino) ]
2117     Conditions at end: []
2118     Add Effects: [Toppled(?x1:domino) ]
2119     Delete Effects: []
2120     Log Strength: 0.0000
2121     Delay Distribution: DiscreteGaussianDelay(5.0000, 0.1000)

```

C.1.5 FAN

Learned predicates and processes. {FanFaces}

```

2126 EndogenousProcess-NoOp:
2127     Parameters: [?robot:robot]
2128     Conditions at start: []
2129     Conditions overall: []
2130     Conditions at end: []
2131     Add Effects: []
2132     Delete Effects: []
2133     Ignore Effects: []
2134     Log Strength: 0.0000
2135     Delay Distribution: ConstantDelay(0.0000)
2136     Option Spec: NoOp(?robot:robot)

```

```

2137 EndogenousProcess-TurnFanOff:
2138     Parameters: [?robot:robot, ?fan:fan]
2139     Conditions at start: [FanOn(?fan:fan) ]
2140     Conditions overall: []
2141     Conditions at end: []
2142     Add Effects: [FanOff(?fan:fan) ]
2143     Delete Effects: [FanOn(?fan:fan) ]
2144     Ignore Effects: []
2145     Log Strength: 0.0000
2146     Delay Distribution: DiscreteGaussianDelay(11.0000, 0.1000)
2147     Option Spec: SwitchOff(?robot:robot, ?fan:fan)

```

```

2148 EndogenousProcess-TurnFanOn:
2149     Parameters: [?robot:robot, ?fan:fan]
2150     Conditions at start: [FanOff(?fan:fan) ]
2151     Conditions overall: []
2152     Conditions at end: []
2153     Add Effects: [FanOn(?fan:fan) ]
2154     Delete Effects: [FanOff(?fan:fan) ]
2155     Ignore Effects: []
2156     Log Strength: 0.0000
2157     Delay Distribution: DiscreteGaussianDelay(13.6667, 2.0817)
2158     Option Spec: SwitchOn(?robot:robot, ?fan:fan)

```

```

2159 ExogenousProcess-Op1:
2160     Parameters: [?x0:ball, ?x2:fan, ?x3:fan, ?x14:loc, ?x15:loc, ?x16:
2161     side]

```

```

2160 Conditions at start: [BallAtLoc(?x0:ball, ?x14:loc), ClearLoc(?x15:
2161 loc), FanFaces(?x2:fan, ?x16:side), FanOff(?x3:fan), FanOn(?x2:fan),
2162 SideOf(?x15:loc, ?x14:loc, ?x16:side)]
2163 Conditions overall: [BallAtLoc(?x0:ball, ?x14:loc), ClearLoc(?x15:loc
2164 ), FanFaces(?x2:fan, ?x16:side), FanOff(?x3:fan), FanOn(?x2:fan),
2165 SideOf(?x15:loc, ?x14:loc, ?x16:side)]
2166 Conditions at end: []
2167 Add Effects: [BallAtLoc(?x0:ball, ?x15:loc)]
2168 Delete Effects: [BallAtLoc(?x0:ball, ?x14:loc)]
2169 Log Strength: 0.0000
2170 Delay Distribution: DiscreteGaussianDelay(21.5000, 8.2407)
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

```

C.2 FURTHER LEARNING AND PLANNING STATISTICS

Each online learning iteration of ExoPredicator takes approximately 30–360 seconds wall-clock time depending on the domain, using 32 CPU cores and no GPU. In comparison, the HRL baseline requires on average about 600 seconds per online iteration, and its runtime increases as it accumulates more data (taking longer to converge), reaching up to 3200 seconds in some cases. The VLM planning baseline does not perform online learning, but can incur substantial test-time cost because it issues computationally expensive VLM calls every time it generates a plan.

Monetarylly, assuming Gemini 2.5 Pro with standard pricing, a representative online iteration consuming a combined 10,000 input tokens and 1,000 output tokens would cost approximately \$0.0225 in total: \$0.0125 for inputs ($10,000 / 1,000,000 \times 1.25$) and \$0.01 for outputs ($1,000 / 1,000,000 \times 10$).

The following table lists the success rate and planning time statistics.

Environment	Manual		Ours		No invent	
	Succ	Time	Succ	Time	Succ	Time
Coffee	99.3	0.612	99.3	0.851	0.0	–
Grow	92.0	0.608	93.3	0.922	0.0	–
Boil	100.0	15.467	92.7	12.204	0.0	–
Domino	97.3	31.710	98.7	21.299	62.0	0.000
Fan	97.3	16.143	97.3	58.244	0.0	–

C.3 PRIORS AND DISTRIBUTIONS

In all experiments, we model process delays with a truncated discrete Gaussian distribution over positive integers $\{1, \dots, 300\}$. The distribution is parameterized by a log-mean and a log-standard-deviation.

The log-mean, log-standard-deviation, and log-process-weights are initialized from a normal distribution with mean 0 and standard deviation 0.01.