

Learning Lifted Action Models from Unsupervised Visual Traces

Kai Xi¹, Stephen Gould¹, Sylvie Thiébaux^{1, 2}

¹School of Computing, The Australian National University

²LAAS-CNRS, Université de Toulouse

oliver.xi@anu.edu.au, stephen.gould@anu.edu.au, sylvie.thiebaux@anu.edu.au

Abstract

Efficient construction of models capturing the preconditions and effects of actions is essential for applying AI planning in real-world domains. Extensive prior work has explored learning such models from high-level descriptions of state and/or action sequences. In this paper, we tackle a more challenging setting: learning lifted action models from sequences of state images, without action observation. We propose a deep learning framework that jointly learns state prediction, action prediction, and a lifted action model. We also introduce a mixed-integer linear program (MILP) to prevent prediction collapse and self-reinforcing errors among predictions. The MILP takes the predicted states, actions, and action model over a subset of traces and solves for logically consistent states, actions, and action model that are as close as possible to the original predictions. Pseudo-labels extracted from the MILP solution are then used to guide further training. Experiments across multiple domains show that integrating MILP-based correction helps the model escape local optima and converge toward globally consistent solutions.

Code — <https://github.com/xikaioliver/ROSAME>

1 Introduction

AI planning enables intelligent agents to generate sequences of actions that achieve specified goals in complex environments. A core requirement for planning is an action model, specifying the preconditions and effects of actions, that allows a planner to reason about how actions change the state of the world. However, acquiring such models manually is often labor-intensive, error-prone, and expensive. This has motivated a growing body of research on learning action models from data, particularly from plan execution traces.

Much of the existing work focuses on learning symbolic action models from example traces composed of high-level, symbolic states and/or actions (Yang, Wu, and Jiang 2007; Cresswell and Gregory 2011; Zhuo and Kambhampati 2013; Aineto, Celorrio, and Onaindia 2019; Juba, Le, and Stern 2021; Lamanna and Serafini 2024; Gösgens, Jansen, and Geffner 2025). While these approaches assume structured symbolic inputs, real-world data often comes in raw perceptual forms such as images or videos. With the advancement

of deep learning, it has become feasible to make state and action predictions directly from such visual inputs, opening the door to learning action models from visual observations.

Two representative works in this direction are Latplan (Asai et al. 2022) and ROSAME (Xi, Gould, and Thiébaux 2024). Latplan is an unsupervised framework that learns action models from pairs of state images representing valid transitions in the environment. However, the resulting models are propositional and uninterpretable, with both states and actions encoded in an opaque latent space. To solve a new problem, Latplan requires images of the initial and goal states. In addition, its latent propositional representation is tied to a fixed object set and therefore cannot generalize to planning instances involving different objects. In contrast, ROSAME is a neuro-symbolic action model learner that computes probabilistic preconditions and effects for action schemas. When combined with deep learning-based state predictors, ROSAME learns lifted, human-readable action models from visual traces. However, it assumes that the executed actions are fully observed.

Here, we address some of the limitations of Latplan and ROSAME by learning lifted, human-readable action models from visual traces consisting of state images, without action supervision. To this end, we develop a deep learning framework that jointly learns to predict states, executed actions, and a lifted action model. While this framework can operate independently, we observe that it is often prone to prediction collapse and self-reinforcing errors among predictions, which prevent convergence to globally consistent solutions.

To overcome this, we introduce a mixed-integer linear program (MILP) as an external source of logical consistency. The MILP corrects a small subset of predicted traces and the action model so that they satisfy the logical constraints of planning while remaining close to the neural model’s original predictions. We extract pseudo-labels from the MILP solution and use them to supervise further training of the deep learning framework. This feedback loop enables the model to escape local optima and correct its own errors.

We evaluate our approach across several classical planning domains using two different types of visual representations. The experimental results show that integrating MILP-based correction substantially improves convergence and logical consistency in most cases, enabling recovery of the ground-truth action models without error.

2 Preliminaries

A typed **planning domain** $D = \langle \mathcal{T}, \mathcal{P}, \mathcal{A}, M \rangle$ consists of: a set \mathcal{T} of types, a set \mathcal{P} of predicates, a set \mathcal{A} of action schemas, and an action model M specifying the preconditions, add and delete effects of each action schema.

The **types** in \mathcal{T} are organized into a tree (or hierarchy). We say that a type t' **subsumes** type t iff t' is either t or an ancestor of t in the tree. A **predicate** $\rho \in \mathcal{P}$ is a tuple $\langle \text{name}, \text{params} \rangle$, where name is a symbol and params is a list of types. We write $\text{name}(\rho)$ and $\text{params}(\rho)$ for the name and parameters of ρ , respectively. Similarly, an **action schema** $\alpha \in \mathcal{A}$ is a tuple $\langle \text{name}, \text{params} \rangle$ whose name and parameters are denoted by $\text{name}(\alpha)$ and $\text{params}(\alpha)$, respectively. We borrow from SAM (Juba, Le, and Stern 2021) and define a **parameter binding** between a predicate ρ and an action schema α as an *injective* function $b_{\rho, \alpha} : \text{params}(\rho) \rightarrow \text{params}(\alpha)$ that maps every parameter $\text{params}(\rho)_i$ of ρ to a parameter of α subsumed by $\text{params}(\rho)_i$. A **parameter-bound predicate** for an action schema α is a pair of the form $\langle \rho, b_{\rho, \alpha} \rangle$, where $b_{\rho, \alpha}$ is a valid parameter binding between ρ and α .

Given \mathcal{P} and \mathcal{A} , our goal is to learn an **action model** M mapping each action schema α , to a triple $M(\alpha) = \langle \text{Pre}(\alpha), \text{Add}(\alpha), \text{Del}(\alpha) \rangle$ of sets of parameter-bound predicates representing its (positive) preconditions, add and delete effects. We denote by $\mathcal{M}(\mathcal{P}, \mathcal{A})$ the set of candidate action models over the given predicates and action schemas.

A **planning instance** $I = \langle O, D \rangle$ consists of a set of objects O and a planning domain D . Each **object** $o \in O$ is associated with a leaf type $\text{type}(o) \in \mathcal{T}$ of the type hierarchy. A **binding** of a predicate ρ is an *injective* function $b_\rho : \text{params}(\rho) \rightarrow O$ mapping every parameter $\text{params}(\rho)_i$ of ρ to an object $o \in O$ such that $\text{params}(\rho)_i$ subsumes $\text{type}(o)$. A **proposition** p is a pair $\langle \rho, b_\rho \rangle$ where ρ is a predicate and b_ρ is a binding for ρ . We say that proposition $p = \langle \rho, b_\rho \rangle$ is a ground instance of predicate ρ . Similarly, a binding of an action schema α is defined like a binding of a predicate, i.e., an *injective* function $b_\alpha : \text{params}(\alpha) \rightarrow O$ where $\forall i, \text{params}(\alpha)_i$ subsumes $\text{type}(b_\alpha(\text{params}(\alpha)_i))$. An **action** $a = \langle \alpha, b_\alpha \rangle$, where α is an action schema and b_α is a binding for α , is a ground instance of the action schema α . The preconditions of a grounded action $a = \langle \alpha, b_\alpha \rangle$, denoted by $\text{Pre}(a)$, is the set of propositions created by taking every parameter-bound predicate $\langle \rho, b_{\rho, \alpha} \rangle$ in the preconditions of the action schema $\text{Pre}(\alpha)$ and grounding ρ with the binding $b_\alpha \circ b_{\rho, \alpha}$. The effects of a grounded action a are defined in a similar manner, and the action model $M(a)$ for the grounded action a is $\langle \text{Pre}(a), \text{Add}(a), \text{Del}(a) \rangle$. Given a planning instance I , we write P_I for the set of propositions, A_I for the set of actions, and $S \subseteq 2^{P_I}$ for the set of **states**.

Let I be a planning instance, $s \in S$ be a state, $a \in A_I$ be an action. We say that a is **applicable** in s iff $\text{Pre}(a) \subseteq s$. The result of applying a in s is the **successor** state $\text{res}(s, a) = (s \setminus \text{Del}(a)) \cup \text{Add}(a)$. A **state trace** for planning instance I is a state sequence $e = (s_1, s_2, \dots, s_{|e|+1})$. The **length** $|e|$ of the trace is its number of transitions. We say that a trace is *k-step* if it has length k . We refer to s_1 as the **initial state** and $s_{|e|+1}$ as the **final state** of the trace.

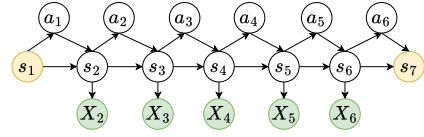


Figure 1: Observations in a visual trace τ compared to the state trace e . Symbol s denotes states and X denotes images. Shaded nodes are observed.

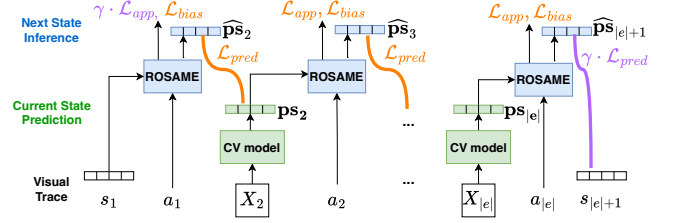


Figure 2: Integrating ROSAME with a state predictor to learn action models from visual traces with fully observed actions, adapted from Xi, Gould, and Thiébaux (2024).

State trace e is **consistent** with an action model M if and only if, according to M , for all $t \in 1, \dots, |e|$, there exists an action a_t that is applicable in s_t and $\text{res}(s_t, a_t) = s_{t+1}$. In the following, we write \mathcal{E}_M^k for the set of state traces of length k that are consistent with action model M .

3 Problem Formulation

We now formalize the problem considered in this paper. For an execution trace e , we observe only a **visual trace**,

$$\tau = (s_1, X_2, X_3, \dots, X_{|e|}, s_{|e|+1}),$$

where s_1 and $s_{|e|+1}$ are states, and each X_t is an image of the corresponding intermediate state. We are given a dataset $\{\tau_j\}_{j=1}^n$ of such visual traces generated under a domain $D^M = \langle \mathcal{T}, \mathcal{P}, \mathcal{A}, M \rangle$ for an unknown action model $M \in \mathcal{M}(\mathcal{P}, \mathcal{A})$. Our assumptions are as follows.

- The initial and final states are fully observed. Having known initial and final states helps prevent degenerate solutions and ensures that the learned state representation remains human-readable. Importantly, we do not include images for these states.
- Each intermediate observation X_t is a fully observed image of its underlying state, i.e., it fully determines the state. Note that the converse is not true: several different images can represent the same state.
- The predicate set \mathcal{P} and action schema set \mathcal{A} (names and parameter signatures) are given.
- For each trace, the object set O is given. For simplicity, our notations assume that all traces share the same object set O , but in principle, traces may come from instances with different object sets.
- Our implementation uses ROSAME (Xi, Gould, and Thiébaux 2024) as a component, where add effects and preconditions cannot intersect, and only preconditions may be deleted. This assumption can easily be relaxed.

Fig. 1 illustrates our observations in a 6-step visual trace compared to the ground-truth state trace.

Given a visual trace τ_j , we wish to use a neural network with parameters θ to predict the underlying state trace e . Ideally, we aim to jointly choose an action model M and neural network parameters θ to maximize the log-likelihood of the visual traces being observations of state traces consistent with the action model M :

$$\ell(M, \theta) = \sum_{j=1}^n \log \left(\sum_{e \in \mathcal{E}_M^{|\tau_j|}} \Pr(e \mid \tau_j; \theta) \right).$$

In practice, this formulation is intractable: evaluating joint distributions over all propositions across all time steps and summing over $e \in \mathcal{E}_M^{|\tau_j|}$ is computationally prohibitive. Furthermore, optimizing over discrete action models is hard.

To make learning feasible, we introduce the following three relaxations:

1. We relax the trace prediction to be the product of marginal state probabilities at each time step.
2. Following ROSAME, we further relax each state prediction to a **probabilistic state vector** $\mathbf{ps}_t \in [0, 1]^{|P_t|}$ listing the marginal probabilities of each proposition.
3. Also following ROSAME, instead of optimizing directly over the discrete space of action models, we compute a probability distribution over action models and modify the successor operator res to be probabilistic so that $res(\mathbf{ps}_t, a)$ gives the expected next probabilistic state.

Under these relaxations, the probability of a trace e given a visual trace τ factorizes as

$$\begin{aligned} \Pr(e \mid \tau) &= \prod_t \Pr(s_{t+1} \mid s_t, X_{t+1}) \\ &= \prod_t \sum_{a \in A_t} \Pr(s_{t+1}, a \mid s_t, X_{t+1}) \\ &= \prod_t \sum_{a \in A_t} \Pr(a \mid s_t, s_{t+1}) \Pr(s_{t+1} \mid s_t, X_{t+1}), \end{aligned}$$

where X_t denotes the image observation at time t . Since the state images are fully observed, $\Pr(s_{t+1} \mid s_t, X_{t+1}) = \Pr(s_{t+1} \mid X_{t+1})$, giving

$$\Pr(e \mid \tau) = \prod_t \sum_{a \in A_t} \Pr(a \mid s_t, s_{t+1}) \Pr(s_{t+1} \mid X_{t+1}).$$

We predict these distributions with two neural networks: a state predictor with parameters θ_s that estimates $\Pr(s_t \mid X_t; \theta_s)$ and outputs \mathbf{ps}_t , and an action predictor with parameters θ_a that estimates $\Pr(a \mid s_t, s_{t+1}; \theta_a)$. The initial and final states come directly from the input; intermediate \mathbf{ps}_t are predicted from images.

We compute a probability distribution over action models with parameters θ_M , which can be obtained using ROSAME (see Sec. 4). We then minimize

$$\mathbb{E}_{M \sim \Pr(M; \theta_M)} \sum_{j,t} \mathcal{L}_{j,t}(M).$$

where for each trace j and step t , the loss $\mathcal{L}_{j,t}(M)$ is

$$\mathbb{E}_{a \sim \Pr(a \mid \mathbf{ps}_t, \mathbf{ps}_{t+1}; \theta_a)} \left[\|\mathit{res}(\mathbf{ps}_t, a) - \mathbf{ps}_{t+1}\|_2^2 + \mathcal{L}(a, \mathbf{ps}_t) \right],$$

and $\mathcal{L}(a, \mathbf{ps}_t)$ is a relaxed loss measuring applicability of action a in state \mathbf{ps}_t .

Note that the loss decomposes over traces and time steps. For clarity in what follows, we define the loss for a single time step t , omitting the trace index j ; the final objective is the sum of these losses across all traces and time steps.

4 ROSAME

Xi, Gould, and Thiébaux (2024) presented a neuro-symbolic action model learner—ROSAME—and integrated it with a deep learning state predictor to enable action model learning from sequences of interleaved state images and observed actions. We build on this work by extending ROSAME to enable action model learning without observing the actions.

ROSAME is a learnable model that takes as input a grounded action and outputs the grounded preconditions, add effects, and delete effects of that action. To achieve this, ROSAME learns a neural network for each action schema, whose outputs are the lifted preconditions and effects for the schema. Given an action a_t performed at step t , ROSAME first looks up its action schema α and computes three vectors: pre_α , add_α , and del_α , where each component represents the probability of a parameter-bound predicate being a precondition, an add effect, or a delete effect, respectively. These lifted conditions and effects are then mapped to their grounded counterparts for that action, i.e., pre_{a_t} , add_{a_t} , and del_{a_t} . Each value in these grounded vectors represents the probability that a proposition is a precondition or effect of the action. This probability is copied from the corresponding lifted vector if a parameter binding exists that unifies the proposition with the action; otherwise, it is set to zero.

Given a state prediction \mathbf{ps}_t and an observed action a_t , ROSAME computes the preconditions and effects of a_t and infers the next state as

$$\widehat{\mathbf{ps}}_{a_t} = \mathbf{ps}_t \odot (1 - \mathit{del}_{a_t}) + (1 - \mathbf{ps}_t) \odot \mathit{add}_{a_t}, \quad (1)$$

where \odot denotes elementwise product.

This predicted state should be close to the next state prediction \mathbf{ps}_{t+1} derived from the next state image. Accordingly, we define the prediction loss as:

$$\mathcal{L}_{\text{pred}} = \text{MSE}(\widehat{\mathbf{ps}}_{a_t}, \mathbf{ps}_{t+1}) \quad (2)$$

In addition, all preconditions of a_t must hold in \mathbf{ps}_t . That is, a proposition cannot be a precondition of a_t and simultaneously be false in \mathbf{ps}_t . This gives rise to an action applicability loss:

$$\mathcal{L}_{\text{app}} = \text{MSE}(\mathit{pre}_{a_t} \odot (1 - \mathbf{ps}_t), \mathbf{0}) \quad (3)$$

ROSAME also introduces a prior bias loss that encourages learning parameter-bound predicates as preconditions, in order to recover prevail conditions—preconditions that are not deleted by actions (Bäckström and Nebel 1995):

$$\mathcal{L}_{\text{bias}} = \text{MSE}(\mathit{pre}_{a_t}, \mathbf{1}) \quad (4)$$

If a proposition remains true both before and after an action, it could be a prevail condition, an add effect, or irrelevant to the action. In such cases, the prior bias loss encourages including the proposition as a precondition.

The total loss used to train ROSAME for the observed action a_t is:

$$\mathcal{L}_{a_t} = \mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{app}} + \lambda \cdot \mathcal{L}_{\text{bias}} \quad (5)$$

where λ is a hyperparameter that controls the strength of the bias term. Gradients are directly backpropagated through the neural networks associated with the action schemas, allowing the model to learn a consistent lifted action model.

For different actions, ROSAME operates like a lookup table: it selects and runs the neural network corresponding to the action schema. ROSAME provides a differentiable and logically consistent way to compute preconditions and effects. In principle, it can be integrated with any deep learning framework that expects such structured inputs and outputs. Fig. 2 illustrates how ROSAME is integrated with a deep learning-based state predictor to enable action model learning from sequences of state images and fully observable actions. A hyperparameter $\gamma \geq 1$ is used to upweight the prediction loss at the final step, since the final state is fully observed.

We treat ROSAME as a black-box action model learner. In particular, it provides the probability distribution over action models $\Pr(M; \theta_M)$ used in our framework. Conceptually, our method does not rely on any ROSAME-specific architecture; it is compatible with any action model learner that produces probabilistic preconditions and effects in a similar form.

5 Deep Learning Framework

Action Prediction and Regularization. In this work, we assume that actions are not observed in the visual traces. As such, we introduce an additional neural network that predicts the action from two consecutive states. Its output is a probability distribution over actions, $\Pr(a_t | \mathbf{ps}_t, \mathbf{ps}_{t+1})$.

To compute the loss in the absence of observed actions, we take the expected value over the ROSAME loss in Eq. 5 for all possible actions to get:

$$\bar{\mathcal{L}}_{a_t} = \sum_{a_t \in A_I} \Pr(a_t | \mathbf{ps}_t, \mathbf{ps}_{t+1}) \mathcal{L}_{a_t} \quad (6)$$

This formulation allows the preconditions and effects for different actions to remain separate during next-state inference and applicability checking (Eq. 2 and 3), while also providing a training signal that encourages the neural network to predict the most consistent action for each step.

However, in practice, action prediction often gets stuck in local minima under the above loss. We found that this arises because some actions are much more similar to each other—both visually and semantically—than to others. For example, in blocksworld, the only difference between `pickup(b1)` and `unstack(b1, b2)` lies in the state they apply to. Confusing them incurs less loss than predicting an unrelated action like `putdown(b1)`. These ambiguities lead to a loss landscape with many local minima, making it difficult for gradient-based optimization to reach the correct solution.

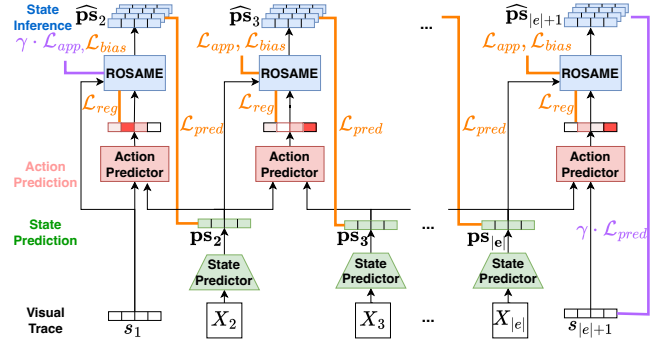


Figure 3: Deep learning framework that predicts states, actions and the action model; a scaling factor $\gamma \geq 1$ emphasizes losses at the fully observed initial and final steps.

To mitigate this issue, we introduce a simple self-regularization loss that helps the learning process escape such local minima by using the locally best action as an auxiliary target. For an action a_t , the log-probability that its inferred next state $\widehat{\mathbf{ps}}_{a_t}$ matches the state prediction \mathbf{ps}_{t+1} is:

$$\log \Pr_{\text{pred}} = \sum_{i=1}^{|P_I|} \log [\widehat{\mathbf{ps}}_{a_t} \odot \mathbf{ps}_{t+1} + (1 - \widehat{\mathbf{ps}}_{a_t}) \odot (1 - \mathbf{ps}_{t+1})]_i \quad (7)$$

Similarly, the log-probability of action a_t being applicable in state \mathbf{ps}_t is:

$$\log \Pr_{\text{app}} = \sum_{i=1}^{|P_I|} \log [1 - \mathbf{pre}_{a_t} \odot (1 - \mathbf{ps}_t)]_i \quad (8)$$

We define the locally best action as:

$$a_t^* = \operatorname{argmax} [\log \Pr_{\text{pred}} + \log \Pr_{\text{app}}] \quad (9)$$

Finally, we augment the loss from Eq. 6 with an additional cross-entropy term that encourages the action prediction to match a_t^* :

$$\mathcal{L}_t = \bar{\mathcal{L}}_{a_t} + \text{CE}(\Pr(a_t | \mathbf{ps}_t, \mathbf{ps}_{t+1}), a_t^*) \quad (10)$$

Fig. 3 presents our framework for learning a lifted action model from visual traces. Since we assume that the initial and final states are fully observed, we introduce a hyperparameter $\gamma \geq 1$ to scale the first-step applicability loss and the last-step prediction loss. This encourages consistency with the available observations.

6 MILP Solution Fixer

The framework described above can, in principle, solve the relaxed task defined in Sec. 3. A globally optimal solution exists in which the predicted states, actions, and action model align with the ground truth. However, in practice, without any action supervision, the system is susceptible to collapse to degenerate solutions—internally consistent predictions of states, actions, and an action model that do not reflect the true underlying dynamics. Moreover, small systematic mistakes early in training can lead to self-reinforcing

local optima, where errors in the state predictions, action predictions, and the learned action model repeatedly support one another rather than being corrected. In such cases, the model can become stuck in configurations where significant inconsistencies with the visual traces persist, and gradient-based training alone struggles to repair these errors.

To prevent such collapse and correct persistent logical inconsistencies, we introduce an external source of regularization. Specifically, we formulate a mixed-integer linear programming (MILP) problem that extracts a logically consistent set of states, actions, and a lifted action model that remain as close as possible to the outputs of the deep learning framework. In the deep learning framework, logical constraints are imposed only through soft relaxations and can therefore be violated. The MILP repairs these errors by enforcing hard constraints while making minimal edits to the predicted states, actions, and action model. This corrective mechanism guides learning away from degenerate solutions and enables the system to escape self-reinforcing local optima. Training converges when neural predictions and MILP solutions agree globally across all training examples.

In the MILP, we define binary decision variables *pre*, *add*, and *del*, indexed by action schema α and parameter-bound predicate $\langle \rho, b_{\rho, \alpha} \rangle$, indicating whether $\langle \rho, b_{\rho, \alpha} \rangle$ is a lifted precondition, add effect, or delete effect of α . These correspond to the structure produced by ROSAME. We also define binary variables *act* and *hol*, where $\text{act}[a, t]$ indicates whether action a is executed at time t , and $\text{hol}[p, t]$ indicates whether proposition p holds at time t .

We further introduce auxiliary step-level indicators *stepadd*, *stepdel*, and *steppre*, where $\text{stepadd}[p, t]$, $\text{stepdel}[p, t]$, and $\text{steppre}[p, t]$ indicate whether proposition p is added, deleted or required as a precondition at time t , respectively.

For a proposition $p = \langle \rho, b_{\rho} \rangle$, we define its **actors** as the set of all actions whose parameters can be bound to instantiate p : $\text{actors}(p) = \{a = \langle \alpha, b_{\alpha} \rangle \mid \exists b_{\rho, \alpha} \text{ s.t. } b_{\rho} = b_{\alpha} \circ b_{\rho, \alpha}\}$.

The inputs to the MILP are the state and action predictions from the deep learning model, the fully observed initial and final states, and the lifted action model predicted by ROSAME. Specifically, $\text{OBS}(p, t)$ denotes the predicted probability that proposition p holds at time t , and $\text{OBS}(a, t)$ denotes the probability that action a is executed at time t . Similarly, ROS_{Pre} , ROS_{Add} , and ROS_{Del} denote the predicted probabilities that a parameter-bound predicate is a precondition, add effect, or delete effect of an action schema.

The objective of the MILP is to maximize the **agreement** between the binary decision variables and the soft predictions. For example, the agreement term for propositions is:

$$\text{OBS}(p, t) \cdot \text{hol}[p, t] + (1 - \text{OBS}(p, t)) \cdot (1 - \text{hol}[p, t]),$$

which is rewritten in linear form below (up to a constant).

We present the formulation for a single trace below; extending it to multiple traces simply replicates the *hol*, *act*, and step-level indicator variables for each trace, while sharing the lifted action model variables *pre*, *add*, and *del* across all traces. The objectives in Eq. 11–12 are then summed over all traces. Eq. 16 adds a prior bias term consistent with Eq. 5, using the same hyperparameter λ .

$$\text{maximize}_{\text{pre, add, del, act, hol}} \sum_{p, t} (2 \cdot \text{OBS}(p, t) - 1) \cdot \text{hol}[p, t] + \quad (11)$$

$$\sum_{a, t} (2 \cdot \text{OBS}(a, t) - 1) \cdot \text{act}[a, t] + \quad (12)$$

$$\sum_{\alpha, \langle \rho, b_{\rho, \alpha} \rangle} (2 \cdot \text{ROS}_{\text{Pre}}(\alpha, \langle \rho, b_{\rho, \alpha} \rangle) - 1) \cdot \text{pre}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + \quad (13)$$

$$\sum_{\alpha, \langle \rho, b_{\rho, \alpha} \rangle} (2 \cdot \text{ROS}_{\text{Add}}(\alpha, \langle \rho, b_{\rho, \alpha} \rangle) - 1) \cdot \text{add}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + \quad (14)$$

$$\sum_{\alpha, \langle \rho, b_{\rho, \alpha} \rangle} (2 \cdot \text{ROS}_{\text{Del}}(\alpha, \langle \rho, b_{\rho, \alpha} \rangle) - 1) \cdot \text{del}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + \quad (15)$$

$$\sum_{\alpha, \langle \rho, b_{\rho, \alpha} \rangle} \lambda \cdot \text{pre}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] \quad (16)$$

$$\text{s.t. } \forall \alpha \text{ and } \langle \rho, b_{\rho, \alpha} \rangle, \quad \text{pre}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + \text{add}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] \leq 1 \quad (17)$$

$$\text{pre}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] \geq \text{del}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] \quad (18)$$

$$\forall p \in s_1, \quad \text{hol}[p, 1] = 1 \quad (19)$$

$$\forall p \notin s_1, \quad \text{hol}[p, 1] = 0 \quad (20)$$

$$\forall p \in s_{|e|+1}, \quad \text{hol}[p, |e| + 1] = 1 \quad (21)$$

$$\forall p \notin s_{|e|+1}, \quad \text{hol}[p, |e| + 1] = 0 \quad (22)$$

$$\forall t, \quad \sum_{a \in A_I} \text{act}[a, t] = 1 \quad (23)$$

$$\forall t, \forall p = \langle \rho, b_{\rho} \rangle, \forall a = \langle \alpha, b_{\alpha} \rangle \in \text{actors}(p), \quad \text{stepadd}[p, t] \leq \text{add}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + 1 - \text{act}[a, t] \quad (24)$$

$$\text{stepadd}[p, t] \geq \text{add}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + \text{act}[a, t] - 1 \quad (25)$$

$$\text{stepdel}[p, t] \leq \text{del}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + 1 - \text{act}[a, t] \quad (26)$$

$$\text{stepdel}[p, t] \geq \text{del}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + \text{act}[a, t] - 1 \quad (27)$$

$$\text{steppre}[p, t] \leq \text{pre}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + 1 - \text{act}[a, t] \quad (28)$$

$$\text{steppre}[p, t] \geq \text{pre}[\alpha, \langle \rho, b_{\rho, \alpha} \rangle] + \text{act}[a, t] - 1 \quad (29)$$

$$\text{stepadd}[p, t] \leq \sum_{a \in \text{actors}(p)} \text{act}[a, t] \quad (30)$$

$$\text{stepdel}[p, t] \leq \sum_{a \in \text{actors}(p)} \text{act}[a, t] \quad (31)$$

$$\text{steppre}[p, t] \leq \sum_{a \in \text{actors}(p)} \text{act}[a, t] \quad (32)$$

$$\text{hol}[p, t + 1] \geq \text{stepadd}[p, t] \quad (33)$$

$$1 - \text{hol}[p, t + 1] \geq \text{stepdel}[p, t] \quad (34)$$

$$\text{hol}[p, t] \geq \text{steppre}[p, t] \quad (35)$$

$$\text{stepadd}[p, t] \geq \text{hol}[p, t + 1] - \text{hol}[p, t] \quad (36)$$

$$\text{stepdel}[p, t] \geq \text{hol}[p, t] - \text{hol}[p, t + 1] \quad (37)$$

Eq. 17 and 18 enforce the validity of the lifted action model according to ROSAME’s assumptions: add effects and preconditions cannot intersect, and only preconditions may be deleted. Eq. 19–22 enforce that the proposition truth values at time steps 1 and $|e| + 1$ match the observed initial and final states. Eq. 23 ensures that exactly one action is executed per timestep.

Eq. 24–29 use variables $\text{act}[a, t]$ to activate the auxiliary step-level indicators when an action is executed. These constraints ensure that, if action a is chosen at time t , its corresponding preconditions and effects are correctly reflected in *steppre*, *stepadd*, and *stepdel*. Conversely, Eq. 30–32 ensure that the auxiliary step-level indicators remain zero if no action capable of influencing proposition p is executed at t .

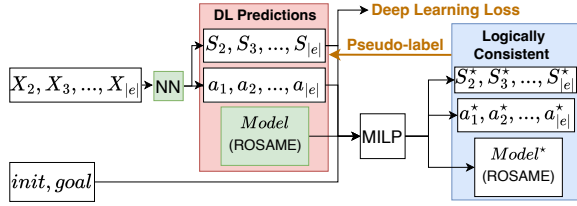


Figure 4: MILP module integrated as a plug-in to generate pseudo-labels from predicted states, actions and the action model, which guide further training.

Finally, Eq. 33–35 ensure that the step-level effects and preconditions are properly reflected in the pre- and post-states, and Eq. 36–37 encode the frame axioms.

In addition to probabilistic predictions encoded in the objectives, we can also encode deterministic knowledge—such as a partial action model, known states or actions, background knowledge, static predicates, and model restrictions such as well-formedness (Gösgens, Jansen, and Geffner 2025)—as constraints directly into the MILP when such information is available.

Integrating MILP with the Framework

Ideally, we would formulate a single MILP over all traces in the training dataset. However, this is infeasible in practice: the solving time grows rapidly with the number of variables, which scales with both the number of traces and the length of each trace. Instead, at regular intervals during training, we solve a MILP on a small, randomly selected subset of traces from the dataset.

We integrate the MILP solver as a plug-in module within the deep learning framework. The deep learning model is trained independently, and periodically we sample several visual traces from the dataset. For each sampled trace, we obtain its predicted states and predicted actions from the neural predictors, and we obtain the lifted action model from ROSAME. Using these predictions, we construct a MILP whose objective measures how closely a logically consistent solution aligns with them. Solving the MILP yields a globally consistent explanation of the sampled traces, from which we derive pseudo-labels that supervise subsequent training. Fig. 4 illustrates this iterative interaction.

In this work, we treat action schema names and parameters as pure symbols, ignoring their semantic meaning. As a result, permuting schema names with identical signatures, or permuting parameters of the same type within a schema, produces another valid representation of the same underlying action model. Consequently, multiple equivalent solutions may exist due to this symmetry. The deep learning model may converge to one such variant, while the MILP may produce another. To align these representations, we enumerate all valid permutations of schema names and parameters for the MILP solution. For each permutation, we evaluate its agreement with the ROSAME action model according to Eq. 13–15. We then select the permutation with the highest agreement and apply it to the MILP solution before extracting pseudo-labels.

MILP Pseudo-label Supervision

The MILP solution provides pseudo-labels for three components of the neural model: the state predictor, the action predictor, and ROSAME.

State supervision. For each time step t , the binary state vector $s_t^* \in \{0, 1\}^{|P_r|}$ is obtained from the MILP decision variables via $s_{t,i}^* = 1$ iff $\text{hol}[p_i, t] = 1$. This serves as the target for the probabilistic state prediction ps_t . We supervise this prediction using binary cross-entropy.

Action supervision. Since the MILP enforces that exactly one action is executed at each time step, we define $a_t^* \in A_I$ as the (unique) action satisfying $\text{act}[a_t^*, t] = 1$. We replace the self-regularization target a_t^* in Sec. 5, with a_t^* as the cross-entropy target.

Action model supervision. For each action schema α and parameter-bound predicate $\langle \rho, b_{\rho, \alpha} \rangle$, ROSAME predicts a probability distribution over four mutually exclusive cases: irrelevant (not used in the action model), add effect, precondition (but not delete effect), or delete effect. From the MILP decision variables pre, add, and del, we determine the target case $c^* \in \{1, 2, 3, 4\}$. We then train ROSAME using the cross-entropy loss against c^* , averaged over all pairs of action schemas and parameter-bound predicates.

Pseudo-label Aging and Updating

The quality of a MILP solution depends strongly on the accuracy of the neural predictions used to construct it. Early in training, these predictions may be noisy, and solving the MILP on a small subset of traces can yield solutions that do not generalize across the dataset. Moreover, MILP solutions obtained at different epochs may not be mutually consistent.

To account for this, we apply an exponential decay to each pseudo-label based on its age. If a pseudo-label is created at epoch e_0 , then at epoch $e > e_0$ its loss contribution is weighted by ψ^{e-e_0} with $\psi < 1$. This schedule ensures that older pseudo-labels gradually lose influence, while newer pseudo-labels—derived from increasingly accurate predictions—are emphasized. When a trace is sampled again during training, its pseudo-labels are recomputed and replace the previous ones.

Finally, the MILP objective itself is flexible: we may include all, some, or weighted combinations of its terms depending on our confidence in the corresponding neural predictions. This flexibility allows the MILP to emphasize reliable predictions while downweighting inconsistent ones, thereby providing more effective supervision for learning.

7 Experiments

We implement the deep learning framework in PyTorch 2.6 and use Gurobi 12.0.1 as the MILP solver. All experiments were conducted on a machine equipped with an AMD Ryzen 9 9900X CPU (12 logical cores), 24 GB of RAM, and a single NVIDIA RTX 4090 GPU.

Domains	$ P_T $	$ A_T $	# Steps	# Fixes	TL(MILP)[s]	Err	Agree	State Acc	Action Acc	t/ep[s]
Blocksworld (MNIST grid)	36	50	3	4	60	0	0.977	97.81%	85.33%	2.50
Gripper	28	50	9	3	150	0	0.978	100%	100%	2.91
Logistics	72	196	9	3	150	0	0.983	99.89%	99.56%	10.64
Blocksworld (Synthesized)	36	50	3	3	60	0	0.976	99.29%	88.67%	1.51
Hanoi	55	120	5	3	90	0	0.940	98.55%	81.40%	4.77
8-puzzle	153	576	5	3	90	0	0.985	99.77%	92.60%	14.54

Table 1: Evaluation results across domains. TL: MILP time limit; t/ep: average training time per epoch.

Visual trace dataset. We use the same dataset as Xi, Gould, and Thiébaux (2024). There are five domains—Blocksworld, Gripper, Logistics, Hanoi, and 8-puzzle—and two types of visualization: (1) digit and letter figures from the MNIST (Deng 2012) and EMNIST (Cohen et al. 2017) datasets to represent objects and backgrounds, arranged in grids for Blocksworld, Gripper, and Logistics; and (2) synthesized images rendered with the PDDLgym library (Silver and Chitnis 2020) for Blocksworld, Hanoi, and 8-puzzle. Note that multiple distinct images may represent the same state. For Blocksworld (both visual representations), block piles may occupy different horizontal positions on the table; for Gripper, balls may appear in different positions within the same room; and for Logistics, objects may occupy different grid cells assigned to the same location.

The object sets for each domain are fixed across all traces: 5 blocks for Blocksworld (both visual representations); 6 balls, 2 grippers, and 2 rooms for Gripper; 6 packages, 2 cities (each with 2 locations, one of which is an airport), 2 trucks, and 2 airplanes for Logistics; 4 discs and 3 pegs for Hanoi; and 8 tiles on a 3×3 board for 8-puzzle.

For each trace, we sample a random state as the initial state using a problem generator, and generate a trace by random walk, uniformly selecting an applicable action at every step. For Blocksworld and 8-puzzle, the generator is provided by the PDDL-generators library (Seipp, Torralba, and Hoffmann 2022); for the remaining domains, we implement problem generators ourselves, since the library does not support generating the range of random initial states we require.

Training. We generate 2,000 traces per domain and split them into training and test sets (90:10). We train for 5,000 epochs using Adam with $\beta = (0.9, 0.999)$, learning rate 10^{-4} , and batch size 128. We set the prior-bias weight $\lambda = 0.4$ and the supervision weight $\gamma = 10$. We also apply data augmentation with image variations of the same state.

Training proceeds in two phases. We begin by training the deep learning model alone for 50 epochs, without MILP integration. At every epoch thereafter, we randomly sample a small set of 3–4 traces from the training set and formulate a corresponding MILP problem. Each MILP is solved with a time limit, and pseudo-labels are extracted from the best feasible solution, if any. These pseudo-labels are used to supervise the neural model, with their influence annealed over time using an exponential decay factor $\psi = 0.99$.

Evaluation and results. To evaluate the learned action model, we follow Xi, Gould, and Thiébaux (2024) in reporting the *Error*, defined as the syntactic difference between the

learned and ground-truth models under the best valid permutation of action schema names and parameters. We also report the maximum agreement score (normalized to $[0, 1]$) between the learned model and the ground truth (as defined in Sec. 6). For state prediction, we compute proposition-wise accuracy using a threshold of 0.5. For action prediction, we report accuracy under the best-matching schema and parameter permutation identified during evaluation.

Tab. 1 summarizes the results. For each domain, we report results for the longest trace length we considered given the time available. We recover the ground-truth action model (up to permutation) and achieve high accuracy in both state and action prediction across all domains. In addition to evaluation metrics, we report the number of traces which the MILP aims to fix at each epoch, the MILP time limit, and the average training time per epoch. We give longer time limits to longer traces, since the MILP problems are larger. During training, we observe that the MILP often requires substantial time to fix traces in early epochs and may produce suboptimal corrections under the time limit. As training progresses, MILP problems become easier to solve, and the average time per epoch becomes much lower than the time limit.

In general, the problem becomes harder as visual traces become longer, for two reasons. First, longer visual traces provide weaker indirect supervision for intermediate states. Second, the complexity of solving the MILP grows rapidly with trace length, limiting how much correction the MILP can provide. In our experiments, agreement with the ground truth model consistently decreases as trace length increases. Interestingly, however, state and action accuracy sometimes increase for longer traces. Our hypothesis is that, since pseudo-labels are generated via MILP, longer traces yield more pseudo-supervision for both states and actions, which can counteract the increased learning difficulty.

Ablation study. Fig. 5 compares learning outcomes with and without MILP. Each bar shows the change in performance obtained by adding MILP on top of the baseline trained *without* MILP. Where needed, incorporating pseudo-label supervision from MILP prevents prediction collapse and helps training escape local optima, enabling convergence to a globally consistent solution. Including MILP reduces model error, improves agreement with the ground truth, and increases both state and action prediction accuracy. For Logistics and 8-puzzle, however, the baseline neural model already attains a correct solution without MILP.

The MILP objective incorporates terms corresponding to state predictions, action predictions, and the action model predicted by ROSAME. As discussed in Sec. 6, this objec-

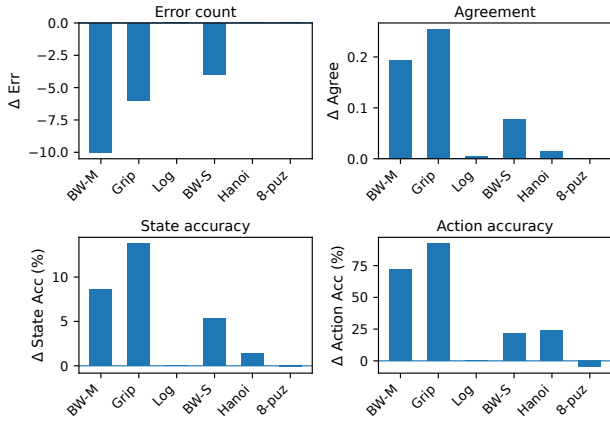


Figure 5: Ablation of MILP correction. Bars show (with MILP) – (without MILP).

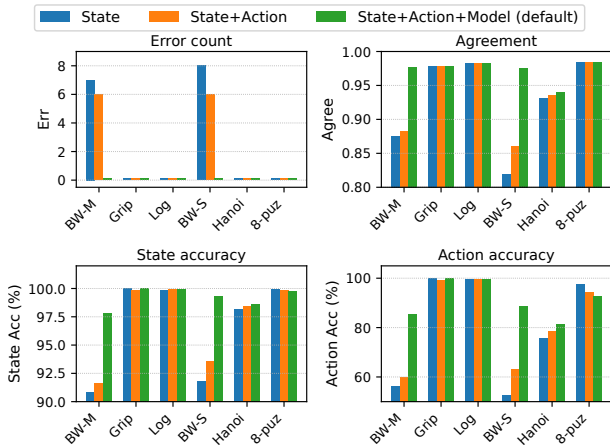


Figure 6: Comparison among different MILP objectives

tive can be flexibly customized. Fig. 6 compares three different configurations of the MILP objective: (1) using only state predictions, (2) using both state and action predictions, and (3) using all terms. While several domains show comparable results across the three configurations, in Blocksworld (both visual representations) including all three terms significantly outperforms the other two. Allowing fewer terms in the objective increases correction flexibility but reduces fidelity to the neural model’s predictions; these effects compete and are difficult to balance. Identifying a more principled way to balance these terms remains future work.

8 Conclusion, Related and Future Work

Our work contributes to the broader problem of automatically constructing planning models from data. Beyond action model learning, a fully autonomous pipeline also involves defining or inventing a symbolic vocabulary that is sufficient for planning (Konidaris, Kaelbling, and Lozano-Perez 2014; Konidaris, Kaelbling, and Lozano-Pérez 2018; Liang et al. 2025; Athalye et al. 2025), abstracting available data into a symbolic representation (James, Rosman, and Konidaris 2022; Verma, Marpally, and Srivastava 2022;

Liang et al. 2025), repairing or reconciling inconsistencies in the learned models (Gragera et al. 2023; Lin, Grastien, and Bercher 2023), and integrating the resulting models with executable low-level skills (Chitnis et al. 2022; Shah, Nagpal, and Srivastava 2025). In this work, we focus on the intersection of symbolic abstraction and action model learning in the context of visual observations. In particular, we address errors and uncertainty introduced during abstraction by leveraging constraints induced by action model and trace consistency. Works that focus on other components of this pipeline are complementary but outside the scope of this paper. Ultimately, a complete solution will require a unified framework that jointly addresses all components.

In this paper, we propose a framework for learning lifted action models from visual traces without action supervision. A closely related line of work is Latplan (Asai and Fukunaga 2017, 2018; Asai and Kajino 2019; Asai and Muise 2020; Asai et al. 2022), an unsupervised framework that learns *latent, propositional* action models from state image pairs. Latplan does not rely on a known symbolic vocabulary or fully specified initial/final states like we do. Meanwhile, its learned model is uninterpretable, cannot generalize to instances with different object counts or visual styles, and cannot be used to plan without images. In contrast, we assume access to the domain’s predicates and action schema signatures, which allow us to leverage ROSAME (Xi, Gould, and Thiébaux 2024) to learn lifted, interpretable action models that generalize across instances. They also enable us to create a simpler and more transparent framework than Latplan.

Bonet and Geffner (2020) investigate a different but related setting, where lifted action models—as well as predicates and action schemas—are learned from a labeled graph representing the state space structure for small instances. Rodriguez et al. (2021) extend this framework to handle partial knowledge of the graph and a limited form of noise that obscures the identity of the target state for certain transitions. These approaches rely on different assumptions and offer complementary strengths.

Our MILP model is also related to model repair methods such as FAMA (Aineto, Celorrio, and Onaindia 2019), which learns or repairs lifted action models—sometimes using only initial and final states—via compilation into classical planning with context-dependent effects. A key difference is that our approach incorporates *probabilistic* knowledge about states, actions, and the action model, whereas FAMA requires deterministic knowledge (which may also be incorporated in our MILP). While it may be possible to simulate our MILP within the FAMA compilation by using context-dependent action costs, few optimal planners support this feature (Geißer, Keller, and Mattmüller 2016; Ivankovic, Gordon, and Haslum 2019), and those that do remain less competitive than mature MILP solvers.

The scalability of our approach is currently bounded by the complexity of solving the MILP. Future work will explore more scalable constraint-solving or approximation techniques that preserve logical structure while reducing computational cost, enabling correction over more and longer traces. Another important direction is to remove the assumption that predicates and action schemas are provided.

Acknowledgments

This work was funded by the Australian Research Council (ARC) under the Discovery Project grant DP220103815 and by the Artificial and Natural Intelligence Toulouse Institute (ANITI) under the grant agreement ANR-23-IACL-0002. We also thank the the anonymous reviewers for their helpful suggestions.

References

- Aineto, D.; Celorrio, S. J.; and Onaindia, E. 2019. Learning Action Models with Minimal Observability. *Artificial Intelligence*, 275: 104–137.
- Asai, M.; and Fukunaga, A. 2017. Classical Planning in Deep Latent Space: From Unlabeled Images to PDDL (and back). In *Proceedings of the 12th International Workshop on Neural-Symbolic Learning and Reasoning (NeSy-17)*.
- Asai, M.; and Fukunaga, A. 2018. Classical Planning in Deep Latent space: Bridging the Subsymbolic-Symbolic Boundary. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*.
- Asai, M.; and Kajino, H. 2019. Towards Stable Symbol Grounding with Zero-Suppressed State Autoencoder. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS-19)*, 592–600.
- Asai, M.; Kajino, H.; Fukunaga, A.; and Muise, C. 2022. Classical Planning in Deep Latent Space. *Journal of Artificial Intelligence Research*, 74: 1599–1686.
- Asai, M.; and Muise, C. 2020. Learning Neural-Symbolic Descriptive Planning Models via Cube-Space Priors: The Voyage Home (to STRIPS). In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI-20)*, 2676–2682.
- Athalye, A.; Kumar, N.; Silver, T.; Liang, Y.; Lozano-Pérez, T.; and Kaelbling, L. P. 2025. Predicate invention from pixels via pretrained vision-language models. In *AAAI 2025 Workshop LM4Plan*.
- Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS+ Planning. *Computational Intelligence*, 11(4): 625–655.
- Bonet, B.; and Geffner, H. 2020. Learning First-Order Symbolic Representations for Planning from the Structure of the State Space. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI-20)*, 2322–2329.
- Chitnis, R.; Silver, T.; Tenenbaum, J. B.; Lozano-Pérez, T.; and Kaelbling, L. P. 2022. Learning Neuro-Symbolic Relational Transition Models for Bilevel Planning. In *Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-22)*, 4166–4173. IEEE.
- Cohen, G.; Afshar, S.; Tapson, J.; and Van Schaik, A. 2017. EMNIST: Extending MNIST to Handwritten Letters. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN-17)*, 2921–2926.
- Cresswell, S.; and Gregory, P. 2011. Generalised Domain Model Acquisition from Action Traces. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS-11)*, 42–49.
- Deng, L. 2012. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Processing Magazine*, 29(6): 141–142.
- Geißer, F.; Keller, T.; and Mattmüller, R. 2016. Abstractions for Planning with State-Dependent Action Costs. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS-16)*, 140–148.
- Gösgens, J.; Jansen, N.; and Geffner, H. 2025. Learning Lifted STRIPS Models from Action Traces Alone: A Simple, General, and Scalable Solution. In *Proceedings of the 35th International Conference on Automated Planning and Scheduling (ICAPS-25)*.
- Gragera, A.; Fuentetaja, R.; García-Olaya, Á.; and Fernández, F. 2023. A planning approach to repair domains with incomplete action effects. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS-23)*, volume 33, 153–161.
- Ivankovic, F.; Gordon, D.; and Haslum, P. 2019. Planning with Global State Constraints and State-Dependent Action Costs. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS-19)*, 232–236.
- James, S.; Rosman, B.; and Konidaris, G. 2022. Autonomous Learning of Object-Centric Abstractions for High-Level Planning. In *Proceedings of the 10th International Conference on Learning Representations (ICLR-22)*.
- Juba, B.; Le, H. S.; and Stern, R. 2021. Safe Learning of Lifted Action Models. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning (KR-21)*, 379–389.
- Konidaris, G.; Kaelbling, L.; and Lozano-Pérez, T. 2014. Constructing symbolic representations for high-level planning. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, volume 28.
- Konidaris, G. D.; Kaelbling, L. P.; and Lozano-Pérez, T. 2018. From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *Journal of Artificial Intelligence Research*, 61: 215–289.
- Lamanna, L.; and Serafini, L. 2024. Action Model Learning from Noisy Traces: a Probabilistic Approach. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS-24)*, 342–350.
- Liang, Y.; Kumar, N.; Tang, H.; Weller, A.; Tenenbaum, J. B.; Silver, T.; Henriques, J. F.; and Ellis, K. 2025. VisualPredicator: Learning Abstract World Models with Neuro-Symbolic Predicates for Robot Planning. In *Proceedings of the 13th International Conference on Learning Representations (ICLR-25)*.
- Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards automated modeling assistance: An efficient approach for repairing flawed planning domains. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI-23)*, volume 37, 12022–12031.
- Rodríguez, I. D.; Bonet, B.; Romero, J.; and Geffner, H. 2021. Learning First-Order Representations for Planning from Black Box States: New Results. In *Proceedings of the*

18th International Conference on Principles of Knowledge Representation and Reasoning (KR-21), 539–548.

Seipp, J.; Torralba, Á.; and Hoffmann, J. 2022. PDDL Generators. <https://doi.org/10.5281/zenodo.6382173>.

Shah, N.; Nagpal, J.; and Srivastava, S. 2025. From Real World to Logic and Back: Learning Generalizable Relational Concepts For Long Horizon Robot Planning. In *Conference on robot learning*. PMLR.

Silver, T.; and Chitnis, R. 2020. PDDL Gym: Gym Environments from PDDL Problems. In *International Conference on Automated Planning and Scheduling (ICAPS) PRL Workshop*.

Verma, P.; Marpally, S. R.; and Srivastava, S. 2022. Discovering User-Interpretable Capabilities of Black-Box Planning Agents. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning (KR-22)*.

Xi, K.; Gould, S.; and Thiébaux, S. 2024. Neuro-Symbolic Learning of Lifted Action Models from Visual Traces. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS-24)*, 653–662.

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

Zhuo, H. H.; and Kambhampati, S. 2013. Action-Model Acquisition from Noisy Plan Traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*, 2444–2450.