Towards Learning Lifted Action Models from Unsupervised Visual Traces

Kai Xi[1], **Stephen Gould**[1], **Sylvie Thiébaux**[1, 2]

[1]School of Computing, The Australian National University
[2]LAAS-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 program (MIP) to prevent self-reinforcing errors among predictions. The MIP 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 MIP solution are then used to guide further training through supervision losses. While the deep learning framework can operate independently, our experiments across multiple domains show that integrating MIP-based correction helps the model escape local optima and converge toward globally consistent solutions.

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 and Fukunaga 2017, 2018; Asai and Kajino 2019; Asai and Muise 2020; 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. 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.

In this work, we address the key limitations of both Latplan and ROSAME. We aim to learn lifted, human-readable action models from visual traces consisting of state images, without access to 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 and performs well in simple domains, we observe that it is prone to self-reinforcing errors among predictions, which prevent convergence to globally consistent solutions.

To overcome this, we introduce a mixed-integer program (MIP) as an external source of logical consistency. The MIP corrects a small subset of predicted traces and the action model to make them consistent with the logical constraints of planning while remaining close to the original predictions. We extract pseudo-labels from the MIP solution and use them to supervise further training of the deep learning framework. This tight feedback loop allows the model to escape local optima and correct its own errors.

We evaluate our approach in three domains: Blocks World, Gripper, and Logistics. While the deep learning framework alone performs well in Gripper, integrating MIP-based correction significantly improves convergence and logical consistency in Blocks World and Logistics, enabling recovery of the ground-truth domain 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 $\langle name, params \rangle$, where $name$ is a symbol and $params$ is a list of types. We write $name(\rho)$ and $params(\rho)$ for the name and parameters of ρ, respectively. Similarly, an **action schema** $\alpha \in \mathcal{A}$ is a tuple $\langle name, params \rangle$ whose name and parameters are denoted by $name(\alpha)$ and $params(\alpha)$, 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 $b_{\rho,\alpha} : params(\rho) \to params(\alpha)$ that maps every parameter $params(\rho)_i$ of ρ to a parameter of α subsumed by $params(\rho)_i$. 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 $M(\alpha) = \langle \mathrm{Pre}(\alpha), \mathrm{Add}(\alpha), \mathrm{Del}(\alpha) \rangle$ 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 $\mathrm{type}(o) \in \mathcal{T}$ of the type hierarchy. A **binding** of a predicate ρ is an *injective* function $b_\rho : params(\rho) \to O$ mapping every parameter $params(\rho)_i$ of ρ to an object $o \in O$ such that $params(\rho)_i$ subsumes $type(o)$. 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 $b_\alpha : params(\alpha) \to O$ where $\forall i$, $params(\alpha)_i$ subsumes $type(b_\alpha(params(\alpha)_i))$. 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 $\mathrm{Pre}(a)$, 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 $\mathrm{Pre}(\alpha)$ 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 $\langle \mathrm{Pre}(a), \mathrm{Add}(a), \mathrm{Del}(a) \rangle$. 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 $\mathrm{Pre}(a) \subseteq s$. The result of applying a in s is the **successor** state $res(s, a) = (s \setminus \mathrm{Del}(a)) \cup \mathrm{Add}(a)$. A **state trace** for planning instance I is a state sequence: $e = s_1 \to s_2 \to \ldots \to s_{|e|} \to s_{|e|+1}$. We refer to s_1 as the **initial state** and $s_{|e|+1}$ as the **final state** of the trace. Trace e is ***consistent*** with an action model M if and only if, according to M, for all $t \in 1, \ldots, |e|$, there exists an action a_t that is applicable in s_t and $res(s_t, a_t) = s_{t+1}$. 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 of learning action models from visual traces that we aim to solve. For an execution trace e, we observe only a **visual trace**, which is a sequence of images: $obs_e = s_1 \to z_2 \to z_3 \to \ldots \to z_{|e|} \to s_{|e|+1}$, where s_1 and $s_{|e|+1}$ are the fully observed initial and final states [1]. Fig. 1 illustrates our observations in a 6-step visual trace compared to the ground-truth state trace.

In this work, we assume access to a set $\{obs_j\}_{j=1}^n$ of visual traces for a planning instance $I^M = \langle O, D^M \rangle$, where $D^M = \langle \mathcal{T}, \mathcal{P}, \mathcal{A}, M \rangle$ for some unknown action model $M \in \mathcal{M}(\mathcal{P}, \mathcal{A})$. We use a neural network with parameters θ_s to estimate the probability of each state s_t from the state image z_t in each visual trace, i.e., $\Pr(s_t \mid z_t; \theta_s)$. Additionally, we employ another neural network with parameters θ_a to predict a probability distribution over actions between every two consecutive states s_{t-1} and s_t, i.e., $\Pr(a \mid s_{t-1}, s_t; \theta_a)$.

The probability of trace e given the visual observation obs_e can then be written as:

$$\begin{aligned} \Pr(e \mid obs_e) &= \prod_t \Pr(s_t \,|\, s_{t-1}, z_t) \\ &= \prod_t \sum_{a \in A_I} \Pr(s_t, a \,|\, s_{t-1}, z_t) \\ &= \prod_t \sum_{a \in A_I} \Pr(a \,|\, s_{t-1}, s_t) \Pr(s_t \,|\, s_{t-1}, z_t) \end{aligned} \tag{1}$$

To simplify the problem, we assume that the state information is dominated by the observation of the state image, i.e., $\Pr(s_t \mid s_{t-1}, z_t) = \Pr(s_t \mid z_t)$. Thus, we have $\Pr(e \mid obs_e) = \prod_t \sum_{a \in A_I} \Pr(a \mid s_{t-1}, s_t) \Pr(s_t \mid z_t)$.

Ideally, we aim to jointly select an action model M and neural network parameters θ_s and θ_a that maximize the log-likelihood of the visual traces being observations of state traces consistent with the action model:

$$\ell(M, \theta_s, \theta_a) = \sum_{j=1}^{n} \log \left(\sum_{e \in \mathcal{E}_M^{|obs_j|}} \Pr(e \,|\, obs_j; \theta_s, \theta_a) \right) \tag{2}$$

Predicting the state probability $\Pr(s_t \mid z_t; \theta_s)$ requires computing the joint distribution over all propositions, and it is intractable to sum over $e \in \mathcal{E}_M^{|obs_j|}$. Therefore, following Xi, Gould, and Thiébaux (2024), we relax the formulation by predicting a **probabilistic state vector**—a vector listing the marginal probabilities of each proposition—and by modifying the successor state operator res to be probabilistic. We denote the probabilistic state vector by $\boldsymbol{ps} \in [0, 1]^{|P_I|}$. At each time step t, for any action a, we can compute the

[1] Having fully observed initial and final states helps prevent degenerate solutions and ensures that the learned state representation remains human-readable. Note that we do not include images for these states.

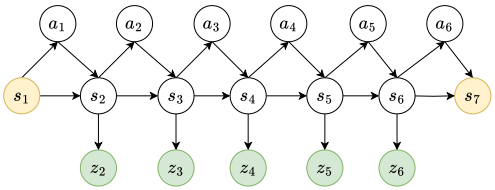

Figure 1: Observations in a visual trace obs_e compared to the state trace e. Symbol s denotes states and z denotes images. Shaded nodes are observed.

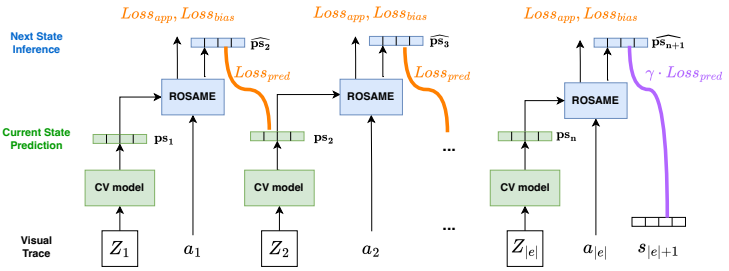

Figure 2: Integrating ROSAME with a state predictor to learn action models from visual traces with fully observed actions (Xi, Gould, and Thiébaux 2024).

expected next probabilistic state vector $res(\boldsymbol{ps}_t, a)$. Furthermore, since we assume that the state image dominates the state information, we can reconstruct the state image $\widehat{z}_t$ from $\boldsymbol{ps}_t$. We then minimize the following objective:

$$\left[\sum_{a \in A_I} \|res(\boldsymbol{ps}_t, a) - \boldsymbol{ps}_{t+1}\|_2^2 + \mathcal{L}(a, \boldsymbol{ps}_t)\right] + \mathcal{L}(\widehat{z}_t, z_t)$$

Here, $\boldsymbol{ps}_t$ is estimated from state image z_t, except for the initial and final states. Term $\mathcal{L}(a, \boldsymbol{ps}_t)$ measures the applicability of action a in the probabilistic state $\boldsymbol{ps}_t$, and $\mathcal{L}(\widehat{z}_t, z_t)$ ensures that the reconstructed image matches the input.

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 and omit the indices j. The final objective is obtained by summing over all traces and all 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.

In general, 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 associated 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 the specific 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 $\boldsymbol{ps}_t$ and an observed action a_t, we use ROSAME to compute the preconditions and effects of a_t, and infer the next state as:

$$\widehat{\boldsymbol{ps}}_{a_t} = \boldsymbol{ps}_t \odot (1 - \boldsymbol{del}_{a_t}) + (1 - \boldsymbol{ps}_t) \odot \boldsymbol{add}_{a_t} \quad (3)$$

where $\odot$ denotes elementwise product.

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

$$L_{\text{pred}} = \text{MSE}(\widehat{\boldsymbol{ps}}_{a_t}, \boldsymbol{ps}_{t+1}) \quad (4)$$

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

$$L_{\text{app}} = \text{MSE}(\boldsymbol{pre}_{a_t} \odot (\mathbf{1} - \boldsymbol{ps}_t), \mathbf{0}) \quad (5)$$

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):

$$L_{\text{bias}} = \text{MSE}(\boldsymbol{pre}_{a_t}, \mathbf{1}) \quad (6)$$

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:

$$L_{a_t} = L_{\text{pred}} + L_{\text{app}} + \lambda \cdot L_{\text{bias}} \quad (7)$$

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 hyperparamter $\gamma \geq 1$ is introduced to scale the prediction loss at the last step because the final state is fully observed.

We treat ROSAME as a black-box action model learner. Our method is, in theory, compatible with any model that computes the same type of probabilistic preconditions and effects for actions.

5 Deep Learning Framework

Action Prediction and Regularization. In this work, we assume that actions are not observed in the example 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, $\Pr(a_t \mid \boldsymbol{ps}_t, \boldsymbol{ps}_{t+1})$.

To compute the loss in the absence of observed actions, we take the expected value over the ROSAME loss in Eq. 7 for all possible actions to get:

$$\bar{L}_{a_t} = \sum_{a_t \in A_I} \Pr(a_t \mid \boldsymbol{ps}_t, \boldsymbol{ps}_{t+1}) L_{a_t} \tag{8}$$

This formulation allows the preconditions and effects for different actions to remain separate during applicability checking and next-state inference (Eq. 4 and 5), 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 Blocks World, the only difference between $\mathrm{pickup}(b1)$ and $\mathrm{unstack}(b1, b2)$ lies in the state they apply to. Confusing them incurs less loss than predicting an unrelated action like $\mathrm{putdown}(b1)$. These ambiguities lead to a loss landscape with many local minima, making it difficult for gradient-based optimization to reach the correct solution.

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, the log-probability that its inferred next state $\widehat{\boldsymbol{ps}}_a$ matches the state prediction $\boldsymbol{ps}_{t+1}$ is:

$$\log \Pr_{\text{pred}} = \sum_{i=1}^{|P_I|} \log \left[\widehat{\boldsymbol{ps}}_a \odot \boldsymbol{ps}_{t+1} + (1 - \widehat{\boldsymbol{ps}}_a) \odot (1 - \boldsymbol{ps}_{t+1})\right]_i \tag{9}$$

Similarly, the log-probability of action a being applicable in state $\boldsymbol{ps}_t$ is:

$$\log \Pr_{\text{app}} = \sum_{i=1}^{|P_I|} \log \left[\mathbf{1} - \boldsymbol{pre}_a \odot (\mathbf{1} - \boldsymbol{ps}_t)\right]_i \tag{10}$$

We define the locally best action as:

$$a^* = \operatorname{argmax} \left[\log \Pr_{\text{pred}} + \log \Pr_{\text{app}}\right] \tag{11}$$

Finally, we augment the loss from Eq. 8 with an additional cross-entropy term that encourages the action prediction to match a^*:

$$L_{\text{reg}} = -\log \Pr(a^*) \tag{12}$$

Image Reconstruction. In addition to requiring action prediction, the removal of action supervision introduces a greater risk of degenerate solutions—cases where the state predictions, action predictions, and action model are internally consistent, but do not reflect the true underlying dynamics. As such, it is insufficient to learn state representations solely from unlabelled state images. Without additional constraints, the model may converge to trivial or collapsed solutions that fail to capture meaningful semantics.

A common approach to address this issue is to use an autoencoder framework, which regularizes the representation by enforcing input-output consistency. Concretely, we use an image encoder to map a state image z_t to a latent state representation s_t, and an image decoder to reconstruct the image $\widehat{z}_t$ from s_t. We then introduce a reconstruction loss to encourage the latent state to retain sufficient information to recover the original image:

$$L_{\text{recon}} = \mathrm{MSE}(\widehat{z}_t, z_t) \tag{13}$$

Fig. 3 presents our framework for learning a lifted action model from image 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 MIP Solution Fixer

The framework described above can be used as a standalone model to solve the relaxed task defined in Sec. 3. There exists at least one globally optimal solution—one in which the predicted states, actions, and action model match the ground truth. However, in practice, the model often fails to converge to this solution. A key reason is that prediction errors across different components can reinforce one another, leading to persistent, self-reinforcing mistakes that gradient-based optimization struggles to correct.

For example, in the Blocks World domain, suppose the action predictor consistently misclassifies all pickup actions as $\mathrm{unstack}$. In response, ROSAME may adjust the $\mathrm{unstack}$ action model to better fit the pre- and post-states of what are actually pickup actions. This reduces the inconsistency loss and reinforces the misclassification. Over time, both components become increasingly confident in these errors, diverging further from the correct solution.

To escape such feedback loops, we introduce an external source of regularization. Specifically, we formulate a mixed-integer programming (MIP) problem to correct persistent logical errors in the predictions made by the deep learning framework. The goal is to find a logically consistent set of states, actions, and a lifted action model that remain as close as possible to the network outputs. In the deep learning framework, logical constraints are incorporated as soft, relaxed losses. Such constraints can be violated if doing so reduces the overall training objective, which is undesirable but still possible. In contrast, the MIP enforces these constraints strictly: it provides a way to fix the deep learning outputs by imposing hard logical consistency while making only minimal edits to the predicted states, actions, and action models.

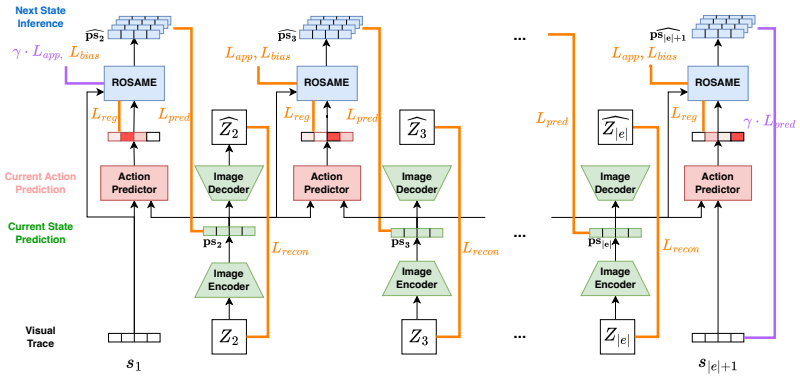

Figure 3: Deep learning framework that predicts states, actions and the action model; a scaling factor $\gamma \geq 1$ emphasizes losses at the fully observed initial and final steps.

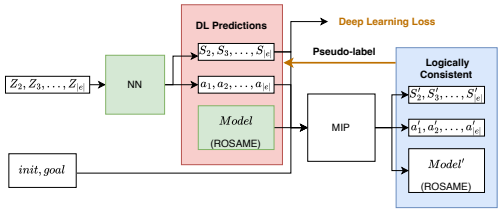

Figure 4: MIP module integrated as a plug-in to generate pseudo-labels from predicted states, actions and the action model, which guide further training.

In the MIP, we define binary decision variables pre, add, and del, indexed by action schema α and parameter-bound predicate $\langle \rho, b_{\rho,\alpha} \rangle$, to indicate whether $\langle \rho, b_{\rho,\alpha} \rangle$ is a lifted precondition, add effect, or delete effect of α. These correspond to the structure learned 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.

The input parameters to the MIP include 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 the action schema α, respectively.

The objective of the MIP is to maximize agreement between the binary decision variables and the soft predictions of the deep learning framework. For example, the probability that $\text{OBS}(p,t)$ agrees with $\text{hol}[p,t]$ is given by:

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

We rewrite this term to bring out its linear form in the objective below, and simplify it by removing a constant offset.

We present the MIP objective for a single trace below; the formulation naturally extends to multiple traces. Eq. 19 adds a prior bias to encourage the inclusion of prevail conditions, where λ is set to the same value used in Eq. 7.

$$\text{maximize}_{\text{pre,add,del,act,hol}}$$

$$\sum_{p,t} (2 \cdot \text{OBS}(p,t) - 1) \cdot \text{hol}[p,t] \quad + \quad (14)$$

$$\sum_{a,t} (2 \cdot \text{OBS}(a,t) - 1) \cdot \text{act}[a,t] \quad + \quad (15)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} (2 \cdot \text{ROS}_{\text{Pre}}(\alpha, \langle \rho, b_{\rho,\alpha} \rangle) - 1) \cdot \text{pre}[\alpha, \langle \rho, b_{\rho,\alpha} \rangle] + \quad (16)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} (2 \cdot \text{ROS}_{\text{Add}}(\alpha, \langle \rho, b_{\rho,\alpha} \rangle) - 1) \cdot \text{add}[\alpha, \langle \rho, b_{\rho,\alpha} \rangle] + \quad (17)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} (2 \cdot \text{ROS}_{\text{Del}}(\alpha, \langle \rho, b_{\rho,\alpha} \rangle) - 1) \cdot \text{del}[\alpha, \langle \rho, b_{\rho,\alpha} \rangle] + \quad (18)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} \lambda \cdot \text{pre}[\alpha, \langle \rho, b_{\rho,\alpha} \rangle] \quad (19)$$

$$\text{s.t.} \quad \text{logical constraints.} \quad (20)$$

The constraints enforce logical consistency among the decision variables. They ensure that the initial and final states match the fully observed values, that one action is executed at each time step, and that the trace is consistent with the action model, i.e., that the preconditions and effects of each executed action hold and that the frame axiom applies. The full formulation is provided in the supplementary material.

Integrating MIP with the Framework

Ideally, we would formulate a single MIP over all traces in the training dataset. However, this is infeasible in practice: the solving time grows exponentially with the number of MIP variables, which increases with the number and length of traces. As a result, we restrict the MIP to a small, randomly selected subset of short traces at a time.

We integrate the MIP solver as a plug-in module into the deep learning framework. The deep learning model is trained independently, and at regular intervals, we sample a few traces from the dataset. From these, we extract the predicted states, actions, and the action model learned by ROSAME, and formulate a MIP whose objective is defined using these predictions. Solving the MIP yields a logically consistent solution, from which we extract pseudo-labels to supervise further training. Fig. 4 illustrates this iterative interaction between the deep learning framework and the MIP.

In this work, we treat action schema names and parameters as purely symbolic, ignoring their semantic meaning. As a result, swapping the names of any two action schemas with identical signatures, or exchanging parameters of the same type within a schema, does not affect the validity of the action model—only the labeling of grounded actions changes

accordingly. This symmetry implies the existence of multiple equivalent solutions. Without any action supervision, our deep learning model may converge to one symmetric variant, while the MIP may output another. To align them, we permute the MIP solution to match the representation learned by the deep learning model.

Specifically, we first enumerate all valid permutations of schema names and parameters for the MIP solution. For each permuted variant, we compute its agreement with the ROSAME action model, as per Eq. 16–18. We select the permutation with the highest agreement score and apply it to the MIP solution. Once aligned, we extract pseudo-labels from the MIP decision variables. These pseudo-labels then serve as supervision signals for the deep learning framework, which we detail in the following subsection.

MIP Pseudo-label Supervision

We use the MIP solution to derive pseudo-labels for state predictions, action predictions, and ROSAME action model predictions.

State supervision. For each time step t, we construct a binary state vector $\widetilde{s}_t \in 0,1^{|P_I|}$, where $\widetilde{s}_{t,i} = 1$ iff proposition $p_i \in P_I$ holds at t according to $\text{hol}[p_i, t]$. This serves as the pseudo-label for the state prediction $\boldsymbol{ps}_t \in [0,1]^{|P_I|}$. We apply binary cross-entropy loss:

$$\text{BCE}(\boldsymbol{ps}_t, \widetilde{s}_t) = \frac{1}{|P_I|} \sum_{i=1}^{|P_I|} \big[\widetilde{s}_{t,i} \log \boldsymbol{ps}_{t,i} + (1 - \widetilde{s}_{t,i}) \log(1 - \boldsymbol{ps}_{t,i})\big]$$

Action supervision. The MIP enforces that exactly one action is executed at each step. We therefore define the pseudo-label $\widetilde{a}_t \in A_I$ such that $\text{act}[\widetilde{a}_t, t] = 1$. The neural network action predictor outputs a probability distribution $\boldsymbol{a}_t = [\Pr(a)]_{a \in A_I}$. We apply cross-entropy loss:

$$\text{CE}(\boldsymbol{a}_t, \widetilde{a}_t) = -\log \Pr(\widetilde{a}_t).$$

Action model supervision. For each action schema α and parameter-bound predicate $\langle \rho, b_{\rho,\alpha} \rangle$, ROSAME predicts a probability vector $\overrightarrow{pr} \in [0,1]^4$ over the only four possible and mutually exclusive cases: (1) none, (2) add effect, (3) precondition but not delete effect, or (4) delete effect. From the MIP solution, we construct the target case index $\widetilde{c} \in \{1,2,3,4\}$ using the decision variables pre, add, and del:

$$\widetilde{c} = \begin{cases} 2 & \text{if } \text{add}[\alpha, \langle \rho, b_{\rho,\alpha} \rangle] = 1 \\ 4 & \text{if } \text{del}[\alpha, \langle \rho, b_{\rho,\alpha} \rangle] = 1 \\ 3 & \text{if } \text{pre}[\alpha, \langle \rho, b_{\rho,\alpha} \rangle] = 1 \text{ and } \text{del}[\alpha, \langle \rho, b_{\rho,\alpha} \rangle] \neq 1 \\ 1 & \text{otherwise} \end{cases}$$

We then compute cross-entropy loss:

$$\text{CE}(\overrightarrow{pr}, \widetilde{c}) = -\log \overrightarrow{pr}_{\widetilde{c}}.$$

This loss is averaged over all pairs of action schemas and parameter-bound predicates in ROSAME.

The quality of the MIP solution depends heavily on the input predictions. While it enforces logical consistency, the MIP may converge to degenerate solutions that ignore the visual content of the state images. Since each MIP is solved on only a small subset of traces, the solution may not generalize to the full training dataset. Moreover, solutions obtained at different training epochs may be mutually inconsistent. However, as training progresses and the neural network's predictions become more accurate, the MIP is more likely to converge to solutions that are closer to the ground truth.

To reflect this, we anneal the influence of each pseudo-label based on its age. Suppose a pseudo-label is generated at epoch e_0. For every subsequent epoch $e > e_0$, we weight the pseudo-label loss by an exponentially decaying factor ψ^{e-e_0}, i.e., $L_{pseudo}(e) = \psi^{e-e_0} L_{pseudo}$, with $\psi < 1$. This decay ensures that older pseudo-labels have diminishing impact on training, while newer pseudo-labels—derived from more accurate predictions—are emphasized. If a trace is re-selected during training, the corresponding pseudo-labels are updated and replace the previous ones.

Finally, it is worth noting that the the MIP objective can be customized: we may include all, some, or a weighted combination of the terms depending on our confidence in each component of the deep learning model. This flexibility allows the MIP to selectively retain reliable predictions while discarding those that are inconsistent.

7 Experiments

We evaluate our approach on 3 classical planning domains: Blocks World, Gripper, and Logistics. For each domain, we consider a specific instance setting: the Blocks World setting includes five blocks; Gripper includes six balls, two grippers, and two rooms; and Logistics involves six packages, four locations in two cities, two trucks, and two airplanes.

To generate state images, we adopt the same grid world visual representation used in the original ROSAME paper (Xi, Gould, and Thiébaux 2024). Objects and backgrounds are represented using figures from the MNIST (Deng 2012) and EMNIST (Cohen et al. 2017) datasets. We assign a specific digit or letter to each object—for example, block 1 is represented by the digit "1", background tiles use "0", and so on. For each such symbol, we select a single figure instance from the dataset (e.g., one particular image of the digit "1") and use it consistently for that object throughout all image constructions. The mapping from symbolic state to image is deterministic.

For each domain, we generate 2000 traces and split them into training and testing sets with a 90:10 ratio. The model is trained for 5000 epochs using the Adam optimizer with learning rate 10^{-4} and $\beta = (0.9, 0.999)$, and with a batch size of 128. We set the prior bias weight $\lambda = 0.4$ and the supervision weight $\gamma = 10$.

Training proceeds in two phases. We begin by training the deep learning model alone for 50 epochs, without any MIP integration. Then, at every epoch thereafter, we randomly sample 3 traces from the training set and formulate the corresponding MIP problem. Each MIP 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 deep model, with their influence annealed over time using an exponential decay factor $\psi = 0.99$.

Domains	$\|P_I\|$	$\|A_I\|$	# Traces	# Steps	MIP Time Limit	Error	Agreement	State Acc	Action Acc
Blocks World	36	50	2000	3	60 s	0	0.963	98.89%	87.33%
Gripper	28	50	2000	3	60 s	0	0.988	99.62%	99.50%
				5	120 s	0	0.984	100%	100%
				7	240 s	0	0.943	99.60%	100%
Logistics	72	196	2000	3	60 s	0	0.992	97.57%	84.14%
				5	120 s	0	0.989	96.91%	74.07%
				7	240 s	0	0.986	99.66%	98.07%

Table 1: Evaluation results

Domain		Err	Agree	State Acc	Action Acc
Blocks World	w/o MIP	8	0.849	91.78%	45.00%
	with MIP	**0**	**0.963**	**98.89%**	**87.33%**
Gripper	w/o MIP	**0**	**0.976**	**100%**	**100%**
	with MIP	**0**	0.943	99.60%	**100%**
Logistics	w/o MIP	6	0.838	95.81%	45.79%
	with MIP	**0**	**0.986**	**99.66%**	**98.07%**

Table 2: Comparison between with and without MIP

MIP Objective	Err	Agree	State Acc	Action Acc
State	7	0.879	93.76%	56.83%
State, Action	0	0.967	99.05%	88.33%
State, Action, Model	0	0.963	98.89%	87.33%

Table 3: Comparison among different MIP objectives

We implement the deep learning framework in PyTorch 2.6 and use Gurobi 12.0.1 as the MIP problem solver. All experiments were conducted on a machine equipped with an AMD Ryzen 9 9900X CPU (24 logical cores), 55 GB of RAM, and a single NVIDIA RTX 4090 GPU.

To evaluate the learned action model, we report Error, defined as the syntactic difference between the learned and ground-truth models under the best valid permutation of action schema names and parameters. This metric follows the definition introduced in the original ROSAME paper. We also report the maximum normalized agreement score 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 across all three domains.

In general, agreement between the learned model and the ground truth decreases as trace length increases. This degradation arises from two main factors. First, as traces grow longer, state and action predictions receive weaker indirect supervision, making them harder to learn accurately. Second, the complexity of solving the MIP grows exponentially with trace length. This restricts the extent of correction the MIP can provide and reduces solution quality. For longer traces, the MIP may return only suboptimal solutions—or fail to produce any solution within the time limit—preventing the extraction of pseudo-labels. To mitigate this, we extend the MIP time limit for longer traces, at the expense of increased training time.

Ablation Study

Tab. 2 compares learning outcomes with and without the MIP module. In Gripper, both settings learn a perfect action model with zero error. As shown in the table, training without MIP slightly outperforms training with MIP on 7-step traces, while the reverse holds on 5-step traces (agreement scores 0.91 vs. 0.984, with 100% state and action accuracies in both cases). In Blocks World and Logistics, however, the standalone deep learning model fails to converge to a globally consistent solution. It becomes trapped in a local optimum, yielding non-zero applicability and prediction losses despite the existence of a zero-loss solution. By contrast, incorporating pseudo-label supervision from the MIP enables training to escape such local optima and converge to a globally consistent solution.

The MIP objective incorporates terms corresponding to state predictions, action predictions, and the action model predicted by ROSAME. As discussed in Sec. 6, this objective can be flexibly customized. We investigate how different configurations of the MIP objective affect learning in the Blocks World domain. Tab. 3 compares results under three configurations: (1) the MIP objective includes only state predictions, (2) includes both state and action predictions, and (3) includes all terms. Configuration (1) performs worse than (2) and (3), though it still outperforms the baseline without MIP. Interestingly, configurations (2) and (3) achieve comparable performance, with (2) slightly ahead. This highlights the potential benefit of tuning the MIP objective based on domain characteristics or prediction confidence.

8 Conclusion, Related and Future Work

We proposed a deep learning framework for learning lifted action models from visual traces without action supervision, along with a MIP module that corrects logical inconsistencies in the learned predictions. Combined, they enable accurate recovery of action models and improved state and action prediction across multiple domains.

Our work contributes to the growing area of learning action models from visual observations. 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 *propositional* action models from pairs of state images. Latplan encodes images

into latent state and action vectors, whose dimensions must be set as hyperparameters. As a result, the learned model is purely latent and cannot generalize to domains with different object counts or visual styles. Moreover, because the model operates entirely in the latent space, users must provide the initial and goal states as images. The resulting plans consist of latent actions that are uninterpretable and cannot be executed in the real world; the only way to interpret them is by decoding intermediate latent states back to images. In contrast, we assume access to the domain's predicates and action schema signatures, which allows us to define the symbolic state and action spaces explicitly. These structural assumptions 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, action schema names, and parameters—are learned from a labeled graph representing the structure of the state space for small instances. In this setting, nodes correspond to black-box states, and transition labels indicate the name of the action schema taken (e.g., pickup). Rodriguez et al. (2021) extend this framework to handle a limited form of noise that obscures the identity of the target state for certain transitions, as well as partial knowledge of the graph, where missing transitions from a state are summarized by a count of how many are labeled with each schema. The learning problem is formulated and solved using combinatorial optimization. Clearly, these frameworks make different assumptions and exhibit different strengths.

Our MIP model is also related to model repair methods such as FAMA (Aineto, Celorrio, and Onaindia 2019), which learns or repairs lifted action models from trajectory data—sometimes using only the initial and final states—via compilation into classical planning with context-dependent effects. A key difference is that our approach incorporates *probabilistic* knowledge about (parts) of states, actions, and the action model, whereas FAMA requires deterministic knowledge[2]. While it may be possible to simulate our MIP 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 they remain less competitive than mature MIP solvers.

Currently, our approach is limited to learning from short traces only. In particular, the scalability of our method is bounded by the complexity of solving the MIP. Future work will explore more scalable constraint solving or approximation techniques that preserve logical structure while reducing computational cost, enabling the correction of longer traces. Another important direction is to remove the assumption that predicate and action symbols are provided. Learning these directly from visual traces would broaden the applicability of our approach to domains lacking symbolic priors. Finally, our current data generation process enforces a one-to-one mapping between symbolic states and state images. While this does not imply that the learning method knows the identity of states, in real-world scenarios, the same symbolic state may have multiple visually distinct appearances. A key challenge for future work is to filter out irrelevant visual details while preserving those that are pertinent to planning.

[2]Note that we could also include deterministic knowledge in our MIP to model state information, actions, parts of the action model, or background knowledge known with certainty.

Acknowledgments

This work was supported by the Australian Research Council grant DP220103815. Sylvie Thiébaux was also supported by the Artificial and Natural Intelligence Toulouse Institute (ANITI) under the grant agreement ANR-23-IACL-0002.

Appendix

A ROSAME Implementation Details

In this section, we provide technical details about the implementation of ROSAME (Xi, Gould, and Thiébaux 2024). Given a set of predicate symbols $\mathcal{P}$ and action symbols $\mathcal{A}$, ROSAME assumes that any valid action model must satisfy the following two constraints:

- add effects and preconditions cannot intersect, i.e., $\mathrm{Add}(\alpha) \cap \mathrm{Pre}(\alpha) = \emptyset$;
- only preconditions can be deleted, i.e., $\mathrm{Del}(\alpha) \subseteq \mathrm{Pre}(\alpha)$.

Based on these constraints, ROSAME enumerates all valid relationships between an action schema α and a parameter-bound predicate $\langle \rho, b_{\rho,\alpha} \rangle$. There are four mutually exclusive cases:

- **Case 1:** $\langle \rho, b_{\rho,\alpha} \rangle$ is not involved in the action model of α.
- **Case 2:** $\langle \rho, b_{\rho,\alpha} \rangle$ is an add effect only.
- **Case 3:** $\langle \rho, b_{\rho,\alpha} \rangle$ is a precondition only.
- **Case 4:** $\langle \rho, b_{\rho,\alpha} \rangle$ is both a precondition and a delete effect (but not an add effect).

ROSAME frames the task of learning action models as a classification problem: for each pair $(\alpha, \langle \rho, b_{\rho,\alpha} \rangle)$, it predicts which of the four cases applies. To do this, ROSAME trains a deep learning classifier for each action schema α, where the inputs are fixed latent vectors—one for each parameter-bound predicate $\langle \rho, b_{\rho,\alpha} \rangle$—and the outputs are probability distributions over the four cases.

Let $\overrightarrow{pr}_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle}$ denote the predicted probability vector for the four cases. These probabilities are then decoded into the lifted precondition and effect indicators via:

$$pre_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} = \overrightarrow{pr}_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} \cdot (0, 0, 1, 1)$$
$$add_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} = \overrightarrow{pr}_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} \cdot (0, 1, 0, 0)$$
$$del_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} = \overrightarrow{pr}_{\alpha,\langle \rho, b_{\rho,\alpha} \rangle} \cdot (0, 0, 0, 1)$$

These expressions compute the expected value for each label by summing over the compatible cases. For instance, the probability that $\langle \rho, b_{\rho,\alpha} \rangle$ is a precondition is the sum of the probabilities assigned to cases 3 and 4.

By aggregating the outputs over all parameter-bound predicates, ROSAME constructs the full lifted precondition and effect vectors pre_α, add_α, and del_α for each action schema α.

B Full MIP Formulation

This section provides the full formulation of the MIP module. Binary decision variables $\text{hol}[p,t]$ and $\text{act}[a,t]$ indicate whether proposition p holds at time t, and whether action a is executed at time t, respectively. The variables pre, add, and del indicate whether a parameter-bound predicate is a lifted precondition or effect of an action schema.

We also define several auxiliary binary variables: prod, clob, user, stepprod, stepclob, and stepuser. Specifically, $\text{prod}[a,p,t]$, $\text{clob}[a,p,t]$, and $\text{user}[a,p,t]$ indicate whether action a adds, deletes, or requires proposition p at time t, respectively; $\text{stepprod}[p,t]$, $\text{stepclob}[p,t]$, and $\text{stepuser}[p,t]$ indicate whether the action executed at step t adds, deletes, or requires proposition p.

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, i.e., $\text{actors}(p) = \{a \mid a = \langle \alpha, b_\alpha \rangle \text{ and } b_\rho = b_\alpha \circ b_{\rho,\alpha}\}$.

We now present the MIP formulation for a single trace; to extend it to multiple traces, we instantiate copies of the hol, act, and auxiliary variables for each trace, while sharing the lifted action model variables pre, add, and del across traces. The objectives in Eq. 21 and 22 are summed over all traces.

$$\text{maximize}_{\text{pre,add,del,act,hol}}$$

$$\sum_{p,t} (2 \cdot \text{OBS}(p,t) - 1) \cdot \text{hol}[p,t] \quad + \quad (21)$$

$$\sum_{a,t} (2 \cdot \text{OBS}(a,t) - 1) \cdot \text{act}[a,t] \quad + \quad (22)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha}\rangle} (2 \cdot \text{ROS}_{\text{Pre}}(\alpha, \langle \rho, b_{\rho,\alpha}\rangle) - 1) \cdot \text{pre}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] + \quad (23)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha}\rangle} (2 \cdot \text{ROS}_{\text{Add}}(\alpha, \langle \rho, b_{\rho,\alpha}\rangle) - 1) \cdot \text{add}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] + \quad (24)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha}\rangle} (2 \cdot \text{ROS}_{\text{Del}}(\alpha, \langle \rho, b_{\rho,\alpha}\rangle) - 1) \cdot \text{del}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] + \quad (25)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha}\rangle} \lambda \cdot \text{pre}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \quad (26)$$

s.t. $\forall \alpha$ and $\langle \rho, b_{\rho,\alpha}\rangle$,

$$\text{pre}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] + \text{add}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \leq 1 \quad (27)$$

$$\text{pre}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \geq \text{del}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \quad (28)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha}\rangle} \text{pre}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \geq 1 \quad (29)$$

$$\sum_{\alpha,\langle \rho, b_{\rho,\alpha}\rangle} \text{add}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \geq 1 \quad (30)$$

$$\forall p \in s_1, \qquad \text{hol}[p,1] = 1 \quad (31)$$

$$\forall p \notin s_1, \qquad \text{hol}[p,1] = 0 \quad (32)$$

$$\forall p \in s_{|e|+1}, \qquad \text{hol}[p,|e|+1] = 1 \quad (33)$$

$$\forall p \notin s_{|e|+1}, \qquad \text{hol}[p,|e|+1] = 0 \quad (34)$$

$$\forall t, \qquad \sum_{a \in A} \text{act}[a,t] = 1 \quad (35)$$

$$\forall t,\ \forall p = \langle \rho, b_\rho \rangle,\ \forall a = \langle \alpha, b_\alpha \rangle \in \text{actors}(p),$$

$$\text{prod}[a,p,t] \leq \text{act}[a,t] \quad (36)$$

$$\text{prod}[a,p,t] \leq \text{add}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \quad (37)$$

$$\text{prod}[a,p,t] \geq \text{act}[a,t] + \text{add}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] - 1 \quad (38)$$

$$\text{clob}[a,p,t] \leq \text{act}[a,t] \quad (39)$$

$$\text{clob}[a,p,t] \leq \text{del}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \quad (40)$$

$$\text{clob}[a,p,t] \geq \text{act}[a,t] + \text{del}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] - 1 \quad (41)$$

$$\text{user}[a,p,t] \leq \text{act}[a,t] \quad (42)$$

$$\text{user}[a,p,t] \leq \text{pre}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] \quad (43)$$

$$\text{user}[a,p,t] \geq \text{act}[a,t] + \text{pre}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] - 1 \quad (44)$$

$$\forall p = \langle \rho, b_\rho \rangle,\ \forall a = \langle \alpha, b_\alpha \rangle \in \text{actors}(p),$$

$$\text{stepprod}[p,t] \geq \text{prod}[a,p,t] \quad (45)$$

$$\text{stepclob}[p,t] \geq \text{clob}[a,p,t] \quad (46)$$

$$\text{stepuser}[p,t] \geq \text{user}[a,p,t] \quad (47)$$

$$\forall t, \forall p,$$

$$\text{stepprod}[p,t] \leq \sum_{a \in \text{actors}(p)} \text{prod}[a,p,t] \quad (48)$$

$$\text{stepclob}[p,t] \leq \sum_{a \in \text{actors}(p)} \text{clob}[a,p,t] \quad (49)$$

$$\text{stepuser}[p,t] \leq \sum_{a \in \text{actors}(p)} \text{user}[a,p,t] \quad (50)$$

$$\forall t, \forall p, \forall a \in \text{actors}(p),$$

$$\text{hol}[p,t+1] \geq \text{prod}[a,p,t] \quad (51)$$

$$1 - \text{hol}[p,t+1] \geq \text{clob}[a,p,t] \quad (52)$$

$$\text{hol}[p,t] \geq \text{user}[a,p,t] \quad (53)$$

$$\forall t, \forall p,$$

$$\text{stepprod}[p,t] \geq \text{hol}[p,t+1] - \text{hol}[p,t] \quad (54)$$

$$\text{stepclob}[p,t] \geq \text{hol}[p,t] - \text{hol}[p,t+1] \quad (55)$$

Eq. 27 and 28 enforce the validity of the lifted action model according to ROSAME's assumptions: add effects and preconditions cannot intersect, and only preconditions can be deleted. Eq. 29 and 30 require that each action schema has at least one precondition and one add effect. Eq. 31–34 ensure that the decision variables for propositions at time steps 1 and $|e|+1$ match the observations in the initial and final states. Eq. 35 enforces that exactly one action is executed at each time step.

Eq. 36–44 define the relationships between lifted and grounded preconditions and effects. Each group of three constraints encodes a logical conjunction: for example, Eq. 36–38 specify that a grounded action a adds proposition p at step t if and only if a is executed at step t and the corresponding parameter-bound predicate is a lifted add effect of the associated action schema α (i.e., $\text{add}[\alpha, \langle \rho, b_{\rho,\alpha}\rangle] = 1$).

Eq. 45–47 specify that if any grounded action adds, deletes, or requires a proposition p at time t, then the corresponding step-level variable $\text{stepprod}[p,t]$, $\text{stepclob}[p,t]$, or

stepuser$[p, t]$ is activated. Conversely, Eq. 48–50 ensure that if a step-level variable is active, it must be due to at least one grounded action with the matching effect or precondition.

Eq. 51–53 propagate state transitions: if an action adds a proposition p at step t, then p must hold at step $t + 1$; if an action deletes p, then p must not hold at $t + 1$; and if an action requires p, then p must hold at t.

Finally, Eq. 54 and 55 encode the frame axioms. If a proposition p holds at step $t + 1$ but not at t, it must have been added at t. If it holds at t but not at $t + 1$, it must have been deleted.

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.

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.

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)*.

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.

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.

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.

Rodriguez, 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.

Xi, K.; Gould, S.; and Thiébaux, S. 2024. Neuro-Symbolic Learning of Lifted Action Models from Visual Traces. In *Proceedings of the 43th 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.