Robust Neuro-Symbolic Goal and Plan Recognition

Leonardo Rosa Amado[1], **Ramon Fraga Pereira**[2,3], **Felipe Meneguzzi**[4,1]

[1]Pontifical Catholic University of Rio Grande do Sul, Brazil
[2] University of Manchester, England, UK
[3] Sapienza University of Rome, Italy
[4] University of Aberdeen, Scotland, UK
leonardo.amado@edu.pucrs.br, ramon.fragapereira@manchester.ac.uk, felipe.meneguzzi@abdn.ac.uk

Abstract

Goal Recognition is the task of discerning the intended goal agent aims to achieve given a sequence of observations, whereas *Plan Recognition* consists of identifying the plan to achieve such intended goal. Regardless of the underlying techniques, most recognition approaches are directly affected by the quality of the available observations. In this paper, we develop *neuro-symbolic* recognition approaches that can combine learning and planning techniques, compensating for noise and missing observations using prior data. We evaluate our approaches in standard human-designed planning domains as well as domain models automatically learned from real-world data. Empirical experimentation shows that our approaches reliably infer goals and compute correct plans in the experimental datasets. An ablation study shows that we outperform existing approaches that rely exclusively on the domain model, or exclusively on machine learning, in problems with both noisy observations and low observability.

1 Introduction

Plan Recognition is the task of inferring the actual plan an observed agent is performing to achieve a goal, given a domain model and a partial, and possibly noisy, sequence of observations (Carberry 2001; Ramírez and Geffner 2009; Sukthankar et al. 2014; Mirsky, Keren, and Geib 2021). Such task arises in a multitude of different areas, including natural language processing (Geib and Steedman 2007), elder-care (Geib 2002), multi-agent systems (Shvo, Sohrabi, and McIlraith 2018), epistemic problems (Shvo et al. 2020) and more (Granada et al. 2017; Wayllace et al. 2020; Shvo and McIlraith 2020). Real-world recognition problems impose limitations on the quality and quantity of the observations from an agent's plan, resulting in observations missing parts of the underlying plan or including spurious observations from silent errors in the sensors. While recent approaches have substantially improved performance under partial observability and noisy (spurious) observations (Zhi-Xuan et al. 2020; Pereira, Oren, and Meneguzzi 2020; Santos et al. 2021), these problems remains a challenge.

Recent work on *Goal and Plan Recognition* use *learned models* to assist planning-based approaches in modeling domain knowledge (Pereira et al. 2019; Zhuo et al. 2020; Amado, Mirsky, and Meneguzzi 2022). *Learning approaches* deal well with noisy data (Kim et al. 2011; Schlimmer and Granger 1986), creating robust models capable of accurate predictions with missing or noisy data. Inspired by such developments, we develop novel recognition approaches that mitigate low and faulty observability, solving both goal and plan recognition problems simultaneously, combining *planning heuristics* and *learning models* into a novel class of *neuro-symbolic* approaches. On the learning side, we train a predictive statistical model of the most likely next states given a set of state observations. We combine such predictive models with *symbolic* heuristics for goal recognition to predict relevant states towards a goal hypothesis given a sequence of observations. In the resulting approaches, the predictive models address the two most common flaws in observations in goal and plan recognition: missing and noisy observations. This allows us to fill in missing observations and rebuild the sequence of states of a complete plan from an initial state to a goal state. While completing missing observations, we detect faulty (noisy) observations and build state sequences that do not necessarily comply with all observations.

We empirically evaluate our approaches in standard *handcrafted* planning domain models, as well as in image-based learned domain models, showing their effectiveness at recognizing both goals and plans. We compare the optimality of the computed plans and the precision of the predicted goals in scenarios with missing and faulty observations against the seminal recognition approaches of Ramírez and Geffner (2009; 2010). In learned domain models, the seminal recognition approaches struggle to achieve high precision, as the number of returned goals is usually very large. Our approaches achieve high precision in all evaluated domains, excelling in learned domains with a precision increase of up to 60%, including problems with noisy observations. We show that our approaches can compute complete optimal plans in most problems, resulting in reliable plan recognition approaches. An ablation study shows the impact of the learning component and the symbolic component on the overall performance of our approaches. This showcases the potential for *neuro-symbolic* approaches for goal and plan recognition, combining the robustness of learned predictive models and the generalization provided by heuristic search guided by informed heuristics.

2 Background and Notation

Automated Planning

A *planning domain* Ξ is a tuple $\langle \mathcal{F}, \mathcal{A} \rangle$, in which $\mathcal{F}$ is a set of facts and $\mathcal{A}$ is a set of actions. States $s \subseteq \mathcal{F}$ are composed of facts, indicating properties that are true at any moment in time. Conditions or formulas comprise positive and negative facts (f, $\neg \mathrm{f}$) representing an implicit conjunctive formula indicating what must be true (alternatively, false) in a state. The positive part $\mathrm{pos}(c)$ of a condition c comprises the positive facts, and the negative part $\mathrm{neg}(c)$ of a condition comprises the negative facts. We say a state s supports a condition c, $s \models c$ (alternatively, c is valid in s), iff all positive facts are present in s, and all negative facts are absent in s, i.e. $s \models c$ iff $(s \cup \mathrm{pos}(c) = s) \wedge (s \cap \mathrm{neg}(c) = \emptyset)$. An action $\mathrm{a} \in \mathcal{A}$ is represented by a tuple $o = \langle \mathrm{pre}(\mathrm{a}), \mathrm{eff}(\mathrm{a}), \mathrm{cost}(\mathrm{a}) \rangle$ containing the preconditions $\mathrm{pre}(\mathrm{a})$, the effects $\mathrm{eff}(\mathrm{a})$, and a non-negative cost $\mathrm{cost}(\mathrm{a})$. The transition of a state s into a new state s' using an action a is represented as $s' = \gamma(s, \mathrm{a})$. The transition is valid iff $s \models \mathrm{pre}(\mathrm{a})$, and $s' = (s \cup \mathrm{pos}(\mathrm{eff}(\mathrm{a}))) - \mathrm{neg}(\mathrm{eff}(\mathrm{a}))$. A *planning problem* is a tuple $\Pi = \langle \Xi, \mathcal{I}, G \rangle$, in which $\mathcal{I}$ is an initial state, G is a goal condition (a conjunctive formula), and Ξ is a planning domain. A *solution* for Π is a sequence of actions (i.e., a plan) $\pi = \langle \mathrm{a}_1, \ldots, \mathrm{a}_n \rangle$ that induces a sequence of states $\langle s_0, s_1, \ldots, s_n \rangle$ such that $\mathcal{I} = s_0 \models \mathrm{pre}(\mathrm{a}_1)$, $s_n \models G$ and that every state $s_i \in \pi$ is such that $s_{i-1} \models \mathrm{pre}(\mathrm{a}_i)$ and $s_i = \gamma(s_{i-1}, a_i)$. The cost of a plan is the sum of the cost of all of its actions such that $\mathrm{cost}(\pi) = \sum_{i=1}^{n} \mathrm{cost}(\mathrm{a}_i)$. An *optimal plan* π^* has the minimum possible cost for achieving a state s_G such that $s_G \models G$ from an initial state I. We assume that every action in $\mathcal{A}$ has cost 1, hence, the optimal plan is the one with the smallest number of actions.

Goal and Plan Recognition as Planning

Goal Recognition is the task of identifying the goal an agent is trying to achieve, whereas *Plan Recognition* is the task of identifying the underlying plan to achieve such goal (Mirsky, Keren, and Geib 2021). Key to solving recognition problems are the observations generated as a consequence of an agent's plan execution. Most recognition approaches use a very specific notion of *observations* consisting of a sequence of identifiers of the actions executed by an agent, or a sequence of states properties, which we formally define as follows. Let $\pi = \langle \mathrm{a}_1, \ldots, \mathrm{a}_n \rangle$ be a plan for a planning problem Π. We define Ω_π as a *sequence of observed actions* from π maintaining the same order but possibly missing actions. Let $\pi = \langle \mathrm{a}_1, \ldots, \mathrm{a}_n \rangle$ be a plan for a planning problem Π, with a sequence of induced states $\mathcal{S}_\pi = \langle s_0, \ldots, s_n \rangle$. We define Ω_s as a *sequence of observed states* from $\mathcal{S}_\pi$ with possibly missing states that maintains the same order. Action and state observations may be noisy if they contain at least one observation not included in the sequence from which they originate (Sohrabi, Riabov, and Udrea 2016; Meneguzzi and Pereira 2021). We denote individual observations corresponding actions a_i as $\vec{a_i}$, and to states s_i as $\vec{s_i}$.

In this paper, we use Ω to refer to any sequence of observations, such that we can define recognition problems independently of the nature of the observations. Thus, using the planning formalism defined above and the standard definition of *Plan Recognition as Planning* of Ramírez and Geffner (2009), we formally define a plan (alternatively goal) *recognition problem* Π^Ω_π (alternatively $\Pi^\Omega_\mathcal{G}$) as tuple $\langle \Xi, \mathcal{I}, \mathcal{G}, \Omega \rangle$, where Ξ is a planning domain, $\mathcal{I}$ is an initial state, $\mathcal{G}$ is a set of *goal hypotheses*, which includes a correct goal G^* (unknown to the observer), and Ω is a sequence of observations (either action or state observations).

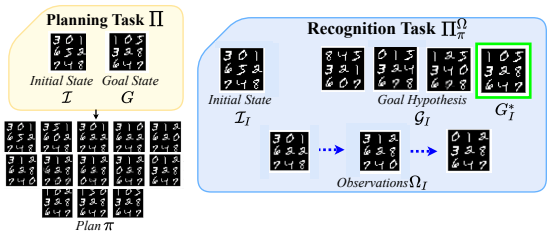

Figure 1: Goal and Plan Recognition in Latent Space.

We establish the key difference between goal and plan recognition as what we expect the solution to be. Namely, we define the solutions as follows. Let $\langle \Xi, \mathcal{I}, \mathcal{G}, \Omega \rangle$ be a plan recognition problem Π^Ω_π (respectively, goal recognition problem $\Pi^\Omega_\mathcal{G}$) with domain Ξ, initial state $\mathcal{I}$, goal hypotheses $\mathcal{G}$, and observations Ω. A *solution* π^* for plan recognition problem Π^Ω_π is one of the least-cost plan resulting from executing the observed plan that generated Ω, whereas a *solution* G^* for goal recognition problem $\Pi^\Omega_\mathcal{G}$ is recognizing the correct goal $G^* \in \mathcal{G}$ resulting from executing the plan that generated Ω. Intuitively, solving a plan recognition problem also solves a goal recognition problem, hence goal recognition problems are a subset of plan recognition problems.

Goal and Plan Recognition in Latent Space

We extend the definition of goal recognition in *latent space* (i.e., image-based domains) proposed by Amado et al. (2018) to also formalize the task of plan recognition in latent space. Here, an image-based plan (alternatively goal) recognition problem $I_{\Pi^\Omega_\pi}$ (alternatively $I_{\Pi^\Omega_\mathcal{G}}$) is a tuple $\langle \Xi_I, \mathcal{I}_I, \mathcal{G}_I, \Omega_I \rangle$, where Ξ_I is an inferred domain knowledge from a set of images, $\mathcal{I}_I$ is an image representation of an initial state, $\mathcal{G}_I$ is a set of image representations of goal hypotheses, which includes a correct goal G^*_I (unknown to the observer), and Ω_I is a sequence of image observations. A solution to $I_{\Pi^\Omega_\pi}$ (alternatively $I_{\Pi^\Omega_\mathcal{G}}$) is to recognize (or anticipate) the sequence of images (alternatively the final image) from a sequence of observed images. Figure 1 illustrates an image-based recognition problem for the 8-puzzle problem.

3 Predictive Plan Recognition (PPR)

We now introduce *Predictive Plan Recognition* (PPR), which is a novel class of approaches to solve both goal and plan recognition problems, dealing with noisy and missing observations. We solve goal and plan recognition by computing a sequence of intermediary states achieved by a plan π given a plan recognition problem $\Pi^\Omega_\pi = \langle \Xi, \mathcal{I}, \mathcal{G}, \Omega \rangle$.

Our algorithm rebuilds the sequence of intermediary states for each goal hypothesis induced by a plan π by iterating through the sequence of observations Ω and *filling in any gaps* due to partial observability. PPR is the first end-to-end goal and plan recognition approach to cope with partial observability with a predictive model. In what follows, we explain each of the key components of this approach: *completing observations*, *predicting states*, and *inferring the goal*.

Observation Completion Toward the Next Subgoal

PPR relies on mechanisms to predict missing observations, allowing it to estimate which states are missing from the observations of a plan. We define this problem as estimating the most likely next state s', given an ordered set of consecutive states $\mathcal{S}$. We refer to this mechanism as a **predictor function**, which we treat as a "black-box", and provide three distinct predictor functions that yield three different types of PPR. Such predictor function can fill in any number of missing observations, which we use to fully reconstruct the sequence of states induced by the plan towards a single goal hypothesis in the process. We determine where observations are faulty or missing by using the known transition model to identify invalid transitions. Formally, let $\Omega_s^{\mathcal{I}} = \langle \vec{s_0} = \mathcal{I}, \vec{s_1}, \ldots, \vec{s_n} \rangle$ be an observation sequence from $\mathcal{I}$, we then check whether $\forall_{i \in [1,\ldots,n]} \exists_{\mathrm{a} \in \mathcal{A}} (\vec{s_i} = \gamma(\vec{s_i{-}1}, \mathrm{a}))$. Namely, if the sequence of states is invalid, we conclude that an observation is missing at the point of the invalid transition of the observation sequence, thus we must predict the next state expected at this point.

Algorithm 1 formalizes our approach to compute a *sequence of states* (or a plan) for a goal hypothesis G. This algorithm takes as input a plan recognition problem Π_π^Ω, a goal G, and a limit λ. To generate a sequence of states $\mathcal{S}$ that a plan π achieves for each goal hypothesis, we use the initial state $\mathcal{I}$ as the starting point (Line 1).

We concatenate the list of state observations Ω with the goal hypothesis G (Line 2) creating a new list $\hat{\Omega}$. We assume that the goal G is a complete state, same as the observations. We iterate through $\hat{\Omega}$ (Line 4), checking if o (an observation from $\hat{\Omega}$) is the result of a valid transition from the last known valid state of $\mathcal{S}$ (denoted as $\mathcal{S}_{|\mathcal{S}|}$). If there exists a valid transition between these two states, we add the observation to the sequence of states and move to the next observation (Line 5 and 10). Otherwise, in Line 6, we use PREDICTNEXTS to predict a missing state in the observations. PREDICTNEXTS can be any function capable of predicting the next state. The algorithm predicts successor states until the predicted state supports a single action to achieve the next observation in $\hat{\Omega}$. This process repeats for every observation, including the goal G, until we compute a sequence of states that can achieve G (Line 9). The algorithm stops when it achieves G during the prediction phase, returning the current sequence of states $\mathcal{S}$ (Line 12).

Since we assume approximate optimality, unlikely goal hypotheses should lead to longer plans. In practice, incorrect goal hypotheses induce much longer plans than the one for the correct hypothesis in domains with connected state spaces, or infinite plans that never reach it. To prevent the algorithm from generating such plans, we stop trying to complete a plan for a goal hypothesis G if during the prediction process we predict λ (a threshold) consecutive states that are unable to achieve the current observation o, returning the current sequence of states. This threshold can be any heuristic value estimating the maximum length of a plan.

Algorithm 1: COMPUTESEQUENCE($\mathcal{I}, \mathcal{A}, \Omega, G, \lambda$)

```
1  S ← ⟨I⟩
2  if Ω_|Ω| ⊨ G then Ω̂ ← Ω else Ω̂ ← Ω · G
3  predicted ← 0
4  for o in Ω̂ do
5      while ¬∃_{a∈A}(o = γ(S_|S|, a)) do
6          s' ← PREDICTNEXTS(A, S, G, o)
7          S ← S · s'
8          predicted += 1
9          if G ∈ S or predicted > λ then return S
10     S ← S · o
11     predicted ← 0
12 return S
```

To deal explicitly with noisy observations, we develop a variation of Algorithm 1. We adopt the usual notion of noisy observation sequence from (Sohrabi, Riabov, and Udrea 2016; Pereira, Oren, and Meneguzzi 2020), which defines that an observation is noisy if it contains observations emitted without a corresponding state or action in the actual plan executed by the observed agent, as illustrated in the center of Figure 2. To recognize plans with noisy observations, we compute plans that can be non-compliant with all observations in Ω, while keeping the assumption of ordered observations. Algorithm 1 will likely fail to find a valid transition to o_i in Line 4 when dealing with noisy observations, either because there is no path to such a state, or because it needs more steps than the threshold. In order to overcome this limitation, we must be able to compute plans that ignore the noisy observations. We assume an observation o_i is noisy if we can predict a state induced by subsequent observations (i.e., o_j, such that $i < j$) that can be reached by valid transitions between the last valid inferred state before o_i and o_j. A skipping mechanism in Algorithm 1 implements this notion.

Figure 2 illustrates the process of computing a plan given three observations, where o_2 is a noisy observation. Blue boxes represent states achieved by plan π; purple boxes represent correct observations; gray boxes represent states predicted by a predictor function (such as Algorithm 2), and finally, red boxes represent noisy observations. In this example, our approach computes the sequence of states and skips observation o_2 by predicting s_4 which corresponds to observation o_3, thus eliminating the need to achieve a valid transition to observation o_2. Our approach thus modified can compute plans that skip noisy observations.

Predictor Functions

Given a sequence of states $\mathcal{S}$, a *predictor function* should ideally compute the probability $\mathbb{P}(s' | \mathcal{S})$ of all states in the

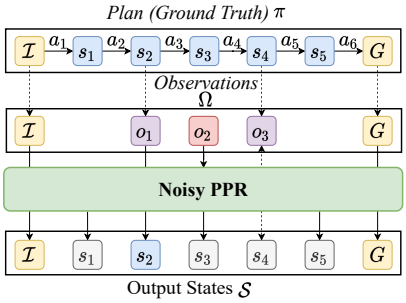

Figure 2: PPR workflow for noisy observations.

state space $2^{\mathcal{F}}$ being the next successor state s', thus selecting the most likely state $\arg\max_{s' \in 2^{\mathcal{F}}} |p(s' | \mathcal{S})|$. Following the maximum likelihood principle, we try to estimate the most likely successor states to a history of states. Here, we detail three distinct predictor functions to predict the most likely next state in our PPR algorithm. First, a machine learning approach using LSTMs to predict s' ($\texttt{PPR}^{\sigma}$). Second, a purely symbolic function leveraging planning heuristics ($\texttt{PPR}_{\text{h}}$). Finally, we combine both previous predictors functions to create a *neuro-symbolic approach* ($\texttt{PPR}^{\sigma}_{\text{h}}$).

PPR$^{\sigma}$: Our first predictor function computes the most probable next state using an LSTM. While any model capable of predicting consecutive states could be used, we use a simple 5-layer neural network. The first layer is an embedding in which we tokenize the input. The second layer is an LSTM with 1024 hidden neurons, which is connected to a self-attention layer with `softmax` activation. A flatten layer takes this output and feeds into a fully connected layer activated with `sigmoid`. The final output is a vector with the size of unique facts $\mathcal{F}$ of the state, where each output node represents a possible fact. Thus, this neural network computes the conditional probability of each individual proposition $\text{f} \in \mathcal{F}$ in the vocabulary, given a history of propositions, i.e., $\mathbb{P}(\text{f} \,|\, \mathcal{S})$. The output of the network is a set of facts (true or false) comprising an approximated state $\hat{s}$, thus reconstructing a state in a binary representation. We train the network using *Binary Cross-Entropy* as a loss function, which applies well here since this is a binary choice: a proposition is either true or false. Notice that our prediction is analogous to building a probabilistic planning graph trained from a sample of plans from a single agent.

Since we have no guarantees that the reconstructed state is valid, we instead use this reconstructed state as a metric to discriminate the successor state. Logically, the successor state must be a state achievable from the last state $\mathcal{S}_{|\mathcal{S}|}$ of the prior state sequence $\mathcal{S}$. $\texttt{PPR}^{\sigma}$ computes the achievable states C from $\mathcal{S}_{|\mathcal{S}|}$ using the planning domain. We discriminate these candidate states by comparing the cosine distance of each achievable state $s \in C$, selecting as s' the state that is closer to the model's prediction. Here, we could use any vector distance metric, but we choose cosine distance as we want to find similarities between two states, as the output of the network can be any number between 0 and 1.

We compute the most likely next s' using Equations 1 and 2 , formalizing our first predictor function, as follows:

$$s' = \arg\min_{c \in C} |\texttt{cos}(bin(c), \hat{s})| \tag{1}$$

$$\texttt{cos}(bin(c), \hat{s}) = \frac{c \cdot \hat{s}}{\|c\| * \|\hat{s}\|} \tag{2}$$

where $bin(c)$ is a binary representation of a candidate state c. Using this predictor function, it is possible to compute state sequences for each one of the goal hypotheses. However, this predictor function is not goal-driven, ignoring the goal hypotheses in its prediction.

PPR$_{\text{h}}$: Our second predictor function, denoted as $\texttt{PPR}_{\text{h}}$, predicts the most likely next state s' without machine learning techniques using exclusively a heuristic function h. Our goal with this predictor function is to compute the most likely next state without relying upon prior data and consider the pursued goal hypothesis. Given a sequence of states $\mathcal{S}$, we use the domain model Ξ to compute all achievable successor states from the last state in the sequence $\mathcal{S}$, following the same procedure as $\texttt{PPR}^{\sigma}$. Given all the achievable states C (candidate states), we apply a domain-independent heuristic to evaluate which state in C is the most likely next state. While the heuristic here can be any planning heuristic, we experiment using the Fast-Forward heuristic (Hoffmann and Nebel 2001) (in the main paper), due to its good balance in terms of speed and information, and Additive Heuristic (in the supplement), as it is an admissible heuristic. To choose the most likely next state in C, we compute the mean h_s of the heuristic value h from each state $s \in C$ to the next observation o we are trying to achieve, and from each state to the goal hypothesis G, as follows:

$$h_s(s, o, G) = (h(s, o) + h(s, G))/2 \tag{3}$$

where we measure how close the state is to the following observation and how close it is to the goal hypothesis. Finally, Equation 3, selects the state with the lowest h_s value as the most likely next state s', using the following equation:

$$s' = \arg\min_{c \in C} |h_s(c, o, G)| \tag{4}$$

PPR$^{\sigma}_{\text{h}}$: Our final predictor function, $\texttt{PPR}^{\sigma}_{\text{h}}$, combines the two previous predictor functions, using a machine learning model and a planning heuristic. This approach aims to combine the reconstruction of $\texttt{PPR}^{\sigma}$ with the heuristic distance between goal hypotheses and the next observation of $\texttt{PPR}_{\text{h}}$.

Like our other predictor functions, we compute the achievable states C from $\mathcal{S}_{|\mathcal{S}|}$, using the planning domain. Using the machine learning model, we compute $\hat{s}$, but instead of using Equation 1 to discriminate the next state s', we compare the cosine distance of each achievable state $s \in C$, removing any state with a distance to $\hat{s}$ larger than a confidence threshold (ϑ) between 0 and 1 in the machine learning model. The following section details how we compute this threshold. Finally, since the machine learning model is goal-independent, we apply a heuristic to measure the distance of all remaining candidate states to the goal and the next observation (Equations 3-4). If there is a tie after using Equation 4, we return the state closest to the goal hypothesis

Algorithm 2: PREDICTNEXTS($\mathcal{A}, \mathcal{S}, G, o$)

$\mathcal{M} \leftarrow$ any model for $p(\mathrm{f} \mid \mathcal{S})$ /* e.g. an ANN*/
$\vartheta \leftarrow$ confidence of $\mathcal{M}$
$\hat{s} \leftarrow$ a state computed using $\mathcal{M}(\mathcal{S})$
$s \leftarrow \mathcal{S}_{|\mathcal{S}|}$
$C \leftarrow \{s' \mid s' = \gamma(s, a), \text{for } a \in A, \text{s.t. } s \models \mathrm{pre}(a)\}$
/* Candidate next states.*/
for $s' \in C$ **do**
  if $distance(bin(s'), \hat{s}) > \vartheta$ **then**
    $C \leftarrow C - s'$
return $\arg\min_{c \in C} |h_s(c, o, G)|$ /* s' with max L_G.*/

G using Equation 1. Algorithm 2 formalizes the entire process for selecting the most likely next state s. The machine learning confidence threshold modulates the impact of the model and the planning heuristic on the selection process. If the threshold is 1.0, the machine learning model has no impact, and the heuristic value dictates the selection of the next state. The planning heuristic has no impact if the threshold is 0, in which case the algorithm selects only the closest state.

Discriminating the Correct Goal and Plan

We solve a plan recognition problem Π^{Ω}_{π} by applying Algorithm 1 to all goals $G \in \mathcal{G}$ and ranking the returned sequences of states with regards to the observations. To infer the intended goal, we compare the predicted sequence of states $\mathcal{S}_G$ for each goal. First, we try to discard goal hypotheses G that are not in the last state of the their predicted sequence $\mathcal{S}_G$ from Algorithm 1. Then, we rank the remaining $\mathcal{S}_G$ based on their compliance with the set of observations Ω, i.e. $|\mathcal{S}_G \cap \Omega|$. More formally, let $\mathcal{S}_G$ be the sequences predicted by COMPUTESEQUENCE$(\mathcal{I}, \mathcal{A}, \Omega, G, \lambda)$ such that $\mathcal{S}_G \models G$, the predicted goal is $G^* = \arg\min_{G \in \mathcal{G}} |C_G|$ where $C_{\mathcal{G}} = \{c \text{ such that } |c| = \max_{G \in \mathcal{G}} |\mathcal{S}_G \cap \Omega| \}$. If there is a tie, we select the shortest sequence of states $\mathcal{S}_G$, following the notion that agents are at least approximately rational and prefer shorter plans (Ramírez and Geffner 2009). Finally, if Algorithm 1 yields no sequence of states that complies with a goal hypothesis, we cannot discriminate the current goal, which happens with $\mathtt{PPR}^{\sigma}$, as it is not goal-driven. In this case, we compare the cosine distance of the last state of each computed sequence with its respective goal hypothesis and select the sequence of states that is closest to its goal hypothesis. Our approach predicts a single sequence of states (from which we can derive a plan), for a single goal as the most likely goal and plan the agent is pursuing, solving both the problems of goal and plan recognition. Hence, it is **guaranteed** to return a goal and a plan, even if not a complete (and valid) plan to achieve the goal hypothesis.

4 Experiments and Evaluation

We evaluate our approaches over two types of datasets: *hand-crafted planning domains*; and *latent-space domains* learned through auto-encoders (Asai and Fukunaga 2018).

Hand-Crafted Domain Datasets: These datasets consist of two domains from the International Planning Competition (IPC), i.e., Blocks-World (BLOCKS) and LOGISTICS. We build datasets using 100 planning problems for each domain. To solve these problems, we use a planner, which computes a plan to solve each problem instance. The plan is converted to a sequence of states, starting from the initial state to the goal state. After computing a plan for each problem, we separate the data into a training set, containing 80 plan instances, and a test set with 20 plan instances. We augment the data used in training by generating windows of the training instances (plans) as new data instances. For example, if we have the train instance $x_1 = [s1, s2, s3, s4]$, where s_n is a state, and $y_1 = [s5]$ is the label to this training instance, we create new training instances, such as $x_{1a} = [s1, s2, s3]$ using $y_{1a} = [s4]$ as label. The plan windows have minimum length of three states and maintain the state contiguity.

Latent Space Datasets: To evaluate our approaches in real-world data, we generated a set of image-based datasets based on existing recognition problems (Amado et al. 2018). We select two domains from (Asai and Fukunaga 2018): MNIST 8-puzzle and Lights-Out Digital (LODIGITAL). The MNIST 8-puzzle uses handwritten digits from the MNIST dataset as tiles. The Lights-Out puzzle game (Fleischer and Yu 2013) consists of a 4 by 4 grid of lights that can be turned on and off, thus named Lights-Out Digital (LODIGITAL). This domain starts with a random number of lights initially on—toggling any of the lights toggles every adjacent light—and the goal is to turn every light off. To generate the training dataset, we create 100 planning problems for each latent space domain. We augment the training set for these domains similarly to the hand-crafted domains.

Training and Testing

After building the training and test datasets, we train 4 distinct models for $\mathtt{PPR}^{\sigma}$ and $\mathtt{PPR}^{\sigma}_{\mathrm{h}}$, one for each domain. We extract the final trace in each sequence and use it as a label to the remainder of the sequence. We train the models with the `Adam` optimizer. During training, our model receives a trace as input, and outputs a prediction, which is a reconstruction of the correct state. We interrupt training after 10 consecutive epochs with no improvement in validation loss.

Table 1 summarizes details for the trained models in each dataset, where $|\mathcal{F}|$ is the size of the output layer, **Max. Len** is the maximum sequence length that can be fed to the network, **State Acc** is the accuracy of the model when reconstructing an entire state (i.e., predicting all facts correctly), **Rec. Acc** is the reconstruction accuracy of the `sigmoid` function (i.e. how many facts the model predicts correctly within each state), and ϑ is our confidence threshold of the ML model used in $\mathtt{PPR}^{\sigma}_{\mathrm{h}}$, measured as the mean value of **State Acc** and **Rec. Acc**.

Experimentation and Setup

We use the 20 planning problems from each domain test dataset to generate observation traces and create plan recognition problems. Recognition approaches use different levels of observability to assess their robustness. We randomly

Domain	$\|f\|$	Max Len	State Acc	Rec. Acc	ϑ
BLOCKS	41	16	0.68	0.97	0.82
LOGISTICS	48	22	0.85	0.99	0.92
MNIST	36	8	0.52	0.97	0.74
LODIGITAL	36	8	0.35	0.92	0.64

Table 1: ML models details and results.

remove observations from the test plans to obtain levels of observability of 10%, 30%, 50%, and 70%, as well as full observability (100%). The resulting traces are similar to that illustrated in Figure 2 (without the noise), where three out of seven observations are missing. Finally, we generate goal and plan recognition problems using the traces of each domain in the datasets for each level of observability, resulting in 100 plan recognition problems for each domain, each including 6 distinct goal hypotheses. We compare the variations of PPR against two seminal plan recognition approaches, RG2009 (Ramírez and Geffner 2009) and RG2010 (Ramírez and Geffner 2010). We experimented with other plan recognition approaches from the literature, such as Vered et al. (2018) and Sohrabi et al. (2016). However, these approaches time-out or run out of memory.

We ran all experiments with a timeout of 1200 seconds per problem in a single core of a 24 core Intel Xeon vE5-2620 CPU @2.00GHz with 160GB of RAM and a memory limit of 8GB and an NVIDIA Titan Xp GPU. For latent space problems, RG2010 timed out in all recognition problems. RG2010 outputs results even when timing out, so we included such results in our evaluation.

Results for Missing and Full Observations

Table 2 shows the results for all four domains comparing our three approaches and RG2009 and RG2010. Column **P** measures *Precision*, which is how many times each approach computes a valid plan for the correct goal and select this plan as the most likely one between six goal hypotheses, divided by the number of recognized goals. Since our approaches always return a single goal, precision measures their accuracy. We use precision instead of accuracy because the spread for latent space domains for both RG2009 and RG2010 is close to 6 (the number of goal hypotheses), making accuracy a less informative metric. Column $\mathbf{R}_{\mathcal{G}}$ shows the average number of returned goals for each approach. Column $\mathbf{t}(s)$ shows the average time an approach takes to solve a problem in any of the given domains. Finally, the π^* column measures the percentage of optimal plans found, since our approach has no formal guarantee of optimality. We compute this value by dividing the number of optimal plans found by the number of correctly predicted goals e.g., if the *accuracy* is 0.5 in 20 problems and 8 of these plans are optimal, π^* is 0.80 (8/10). $\text{PPR}^{\sigma}_{\text{h}}$ computes plans with high precision outperforming the other approaches in almost every scenario. For LOGISTICS, $\text{PPR}^{\sigma}_{\text{h}}$ had slightly less precision when compared to PPR_{h}, which does not rely on machine learning models. This is likely due to the inherent parallelism of the LOGISTICS domain, which enables multiple linearizations for the same plan, confusing the predictive model. Still, $\text{PPR}^{\sigma}_{\text{h}}$ is

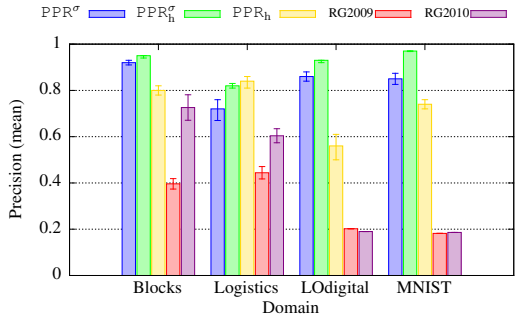

Figure 3: Performance of all approaches for each domain.

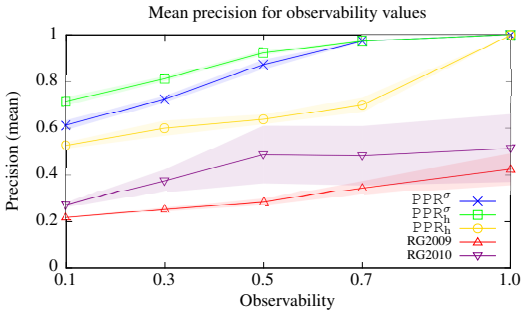

Figure 4: Performance of all approaches per observability.

faster and computes more optimal plans than PPR_{h}. Figure 3 shows the mean precision (with variance) for all domains. Overall, $\text{PPR}^{\sigma}_{\text{h}}$ outperforms all approaches with low variance through all domains. Figure 4 shows the mean precision and variance of all approaches over all observability levels. $\text{PPR}^{\sigma}_{\text{h}}$ dominates the other approaches (within variance) over all levels of observability. With the Table 2 and Figures 3 and 4, we can conclude that $\text{PPR}^{\sigma}_{\text{h}}$ is the most accurate approach, and returns optimal plans more often when compared to PPR_{h} and PPR^{σ}.

Results for Noise in Missing and Full Observations

To further stress our approaches, we introduce noise in the observations using two new datasets for each of our domains, one with 10% noise and the other with 20% noise. We introduce noise by iterating through all observations of the dataset and swapping a correct observation for a noisy one with a probability of either 10% or 20%. Thus, a dataset with 10% of noise means that, of all observations in the dataset, 10% are invalid observations through all problems, with no guarantee that a given problem will have a noisy observation. Table 3 shows the results for the 8 noisy datasets. We do not include RG2010 in this comparison because it either times out, or it is dominated by RG2009. Our approaches vastly outperform RG2009 in terms of precision while trading off time to recognition. RG2009 has poor precision since it tries to compute optimal plans, including the noisy observation, while our approaches skip the noisy observation altogether when disagreeing with the predictive model. The addition of noise significantly increases the execution time of PPR_{h}, but both approaches that rely on machine learning models (PPR^{σ}, PPR^{σ}) are less affected.

Missing and Full Observations																				
	PPR$^{\sigma}$				PPR$_{\text{h}}$				PPR$^{\sigma}_{\text{h}}$				RG2009				RG2010			
	P	$\mathbf{R}_{\mathcal{G}}$	**t**(***s***)	π^*	**P**	$\mathbf{R}_{\mathcal{G}}$	**t**(***s***)	π^*	P	$\mathbf{R}_{\mathcal{G}}$	**t**(***s***)	π^*	P	$\mathbf{R}_{\mathcal{G}}$	**t**(***s***)	π^*	P	$\mathbf{R}_{\mathcal{G}}$	**t**(***s***)	π^*
BLOCKS	0.92	1.0	0.55	0.90	0.80	1.0	0.05	0.51	**0.95**	1.0	0.68	0.97	0.40	2.1	0.40	1.0	0.73	1.2	168.8	1.0
LOGISTICS	0.72	1.0	6.8	0.2	**0.84**	1.0	35.9	0.67	0.82	1.0	7.20	0.81	0.44	1.9	0.56	1.0	0.60	1.4	12.1	1.0
LODIGITAL	0.86	1.0	2.8	0.56	0.56	1.0	41.4	0.30	**0.93**	1.0	17.26	0.88	0.20	4.9	81.1	1.0	0.19	4.9	1200.0	1.0
MNIST	0.85	1.0	2.70	0.70	0.74	1.0	25.23	0.40	**0.97**	1.0	7.7	0.90	0.18	5.3	41.5	1.0	0.19	4.9	1200.0	1.0

Table 2: Results for missing and full observations.

Missing and Full Observations with 10% Noise												
	PPR$^{\sigma}$			PPR$_{\text{h}}$			PPR$^{\sigma}_{\text{h}}$			RG2009		
	P	**t**(***s***)	π^*	**P**	**t**(***s***)	π^*	**P**	**t**(***s***)	π^*	**P**	**t**(***s***)	π^*
BLOCKS	0.85	0.52	0.75	0.64	0.06	0.36	**0.92**	0.58	0.76	0.11	0.36	1.0
LOGISTICS	0.72	6.8	0.32	**0.84**	34.4	0.46	0.81	13.86	0.8	0.53	0.27	1.0
LODIGITAL	0.79	3.72	0.70	0.67	134.66	0.41	**0.84**	18.9	0.79	0.20	85.3	1.0
MNIST	0.86	2.43	0.44	0.67	24.63	0.33	**0.96**	9.70	0.61	0.18	27.6	1.0

Missing and Full Observations with 20% Noise												
	PPR$^{\sigma}$			PPR$_{\text{h}}$			PPR$^{\sigma}_{\text{h}}$			RG2009		
	P	**t**(***s***)	π^*	**P**	**t**(***s***)	π^*	**P**	**t**(***s***)	π^*	**P**	**t**(***s***)	π^*
BLOCKS	0.84	0.82	0.33	0.63	0.08	0.39	**0.91**	0.73	0.68	0.07	0.35	1.0
LOGISTICS	0.73	7.14	0.47	**0.82**	39.81	0.39	0.80	16.54	0.77	0.40	0.23	1.0
LODIGITAL	0.86	3.01	0.66	0.58	82.04	0.31	**0.87**	20.06	0.75	0.20	87.6	1.0
MNIST	0.78	2.50	0.49	0.70	29.25	0.24	**0.94**	12.44	0.63	0.18	30.8	1.0

Table 3: Results for missing and full observations with noise.

5 Related Work

Current approaches to *Goal and Plan Recognition* employ both symbolic-based search and machine-learning based techniques in isolation, but they all share limitations handling noise or partial observability. Ramirez and Geffner's (2009; 2010) seminal approaches for *goal and plan recognition as planning* model the problem as a planning problem towards to a set of goal hypotheses and solve it using planning algorithms. Work by Sohrabi, Riabov, and Udrea (2016) explicitly deals with noisy and missing observations by running an expensive top-k planner. More recent work by Vered et al. (2018) efficiently recognizes goals combining the planning landmarks and plan *mirroring*. Related approaches to goal (but not plan) recognition include that of E-Martin et al. (2015) and Pereira et al. (2017; 2020), both of which perform recognition based on the structure of planning problems. Most recently, Zhi-Xuan et al. (2020) relax the optimality (rationality) assumption, and introduce a Bayesian approach for online goal inference that deals with sub-optimal behavior by modeling the observed agents as *boundedly-rational* planners, restricting the amount of resources available to an observed agent for planning.

By contrast, approaches based on machine learning forgo engineered domains by learning the transition function using data. Min et al. (2014) develop an LSTM-based approach to recognize the goals of a player in an educational game. The dataset used for training an LSTM (Hochreiter and Schmidhuber 1997) and a player behavior corpus consisting of distinctive player actions, which are noisy and sub-optimal, annotated with the corresponding goal. Unlike our work, their approach can only recognize goals included in the training dataset. Zhuo et al. (2020) develop plan recognition using learned models to predict the next action given a set of observed actions, aiming to reconstruct a plan. Unlike our work, they predict actions instead of states, and assume much higher observability and no noise. By contrast, we deal with very low observability and noisy observations. Finally, while LatPlan (Asai and Fukunaga 2018) is not an approach for goal recognition, its use of a learned *latent* model to plan serves as inspiration to our work.

6 Discussion and Future Work

We developed a *neuro-symbolic* approach for goal and plan recognition, PPR, combining machine learning statistical prediction with domain knowledge within planning techniques. The resulting approach achieves very high precision in hand-crafted and automatically generated plan recognition domains. We empirically show that our best approaches can reconstruct plans with very low observability (up to 90% missing) and noisy observations (up to 20% noise). Our machine learning model is simple enough that the same network architecture works for all domains; thus, tuning the machine learning model is not really required for our approach. Importantly, we show that the machine learning model is critical for the robustness of our approaches, as the purely symbolic approaches have substantially lower accuracy and higher variance.

The main limitation of our PPR approach is the requirement of data, which is absent in standard recognition approaches. This limitation entails that the length of the binarized vector representing state features imposes a limit on the generalization of the networks within the same domain. This limitation is relatively strong for standard recognition using hand-crafted domain models. However, we argue that this is less of an issue in automatically learned domains (e.g., in latent space (Asai and Fukunaga 2018)), as they are both intrinsic to the learned domain. On the flip side, the predictive model implicitly encodes a preference relation for goals given a sequence of observations, i.e., $\mathbb{P}(\,\Omega \mid G)$ used for some approaches in the literature (Ramírez and Geffner 2010). Our neuro-symbolic PPR approach is entirely modular and can use any new predictor function, which we intend to explore as future work with more complex learning models. This approach provides a critical initial step for neuro-symbolic approaches to goal and plan recognition, bridging the gap between learned behavior models and planning algorithms, allowing the scalable, robust and accurate recognition of complex image-based domains.

Acknowledgements

Felipe Meneguzzi acknowledges support from the CNPq with project 302773/2019-3 (PQ Fellowship). Ramon Fraga Pereira acknowledges support from the ERC Advanced Grant WhiteMech (No. 834228) and the EU ICT-48 2020 project TAILOR (No. 952215).

References

Amado, L.; Mirsky, R.; and Meneguzzi, F. 2022. Goal Recognition as Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Amado, L.; Pereira, R. F.; Aires, J. P.; Magnaguagno, M.; Granada, R.; and Meneguzzi, F. 2018. Goal Recognition in Latent Space. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

Asai, M.; and Fukunaga, A. 2018. Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Carberry, S. 2001. Techniques for Plan Recognition. *User Modeling and User-Adapted Interaction*, 11: 31–48.

Fleischer, R.; and Yu, J. 2013. *A Survey of the Game "Lights Out!"*, 176–198. Springer Berlin Heidelberg.

Geib, C. W. 2002. Problems with intent recognition for elder care. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Geib, C. W.; and Steedman, M. 2007. On natural language processing and plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Granada, R.; Pereira, R. F.; Monteiro, J.; Barros, R.; Ruiz, D.; and Meneguzzi, F. 2017. Hybrid Activity and Plan Recognition for Video Streams. In *The AAAI 2017 Workshop on Plan, Activity, and Intent Recognition*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Kim, S.; Zhang, H.; Wu, R.; and Gong, L. 2011. Dealing with noise in defect prediction. In *Proceedings of the International Conference on Software Engineering (ICSE)*.

Martín, Y. E.; Moreno, M. D. R.; and Smith, D. E. 2015. A Fast Goal Recognition Technique Based on Interaction Estimates. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Meneguzzi, F.; and Pereira, R. F. 2021. A Survey on Goal Recognition as Planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Min, W.; Ha, E.; Rowe, J. P.; Mott, B. W.; and Lester, J. C. 2014. Deep Learning-Based Goal Recognition in Open-Ended Digital Games. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AIIDE)*.

Mirsky, R.; Keren, S.; and Geib, C. 2021. Introduction to Symbolic Plan and Goal Recognition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 16(1): 1–190.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2017. Landmark-Based Heuristics for Goal Recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2020. Landmark-Based Approaches for Goal Recognition as Planning. *Artificial Intelligence*, 279: 103217.

Pereira, R. F.; Vered, M.; Meneguzzi, F.; and Ramírez, M. 2019. Online Probabilistic Goal Recognition over Nominal Models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Ramírez, M.; and Geffner, H. 2009. Plan Recognition as Planning. In *Proceedings of International Joint Conference on Artifical Intelligence (IJCAI)*.

Ramírez, M.; and Geffner, H. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Santos, L. R. A.; Meneguzzi, F.; Pereira, R. F.; and Pereira, A. G. 2021. An LP-Based Approach for Goal Recognition as Planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Schlimmer, J. C.; and Granger, R. H. 1986. Incremental Learning from Noisy Data. *Machine Learning*, 1(3): 317–354.

Shvo, M.; Klassen, T. Q.; Sohrabi, S.; and McIlraith, S. A. 2020. Epistemic Plan Recognition. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Shvo, M.; and McIlraith, S. A. 2020. Active Goal Recognition. In *The AAAI Conference on Artificial Intelligence*.

Shvo, M.; Sohrabi, S.; and McIlraith, S. A. 2018. An AI Planning-Based Approach to the Multi-Agent Plan Recognition Problem. In *Canadian Conference on AI*.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan Recognition as Planning Revisited. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Sukthankar, G.; Goldman, R. P.; Geib, C.; Pynadath, D. V.; and Bui, H. H. 2014. *Plan, Activity, and Intent Recognition: Theory and Practice*. Elsevier.

Vered, M.; Pereira, R. F.; Magnaguagno, M. C.; Kaminka, G. A.; and Meneguzzi, F. 2018. Towards Online Goal Recognition Combining Goal Mirroring and Landmarks (Extended Abstract). In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*.

Wayllace, C.; Ha, S.; Han, Y.; Hu, J.; Monadjemi, S.; Yeoh, W.; and Ottley, A. 2020. DRAGON-V: Detection and Recognition of Airplane Goals with Navigational Visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhi-Xuan, T.; Mann, J. L.; Silver, T.; Tenenbaum, J.; and Mansinghka, V. 2020. Online Bayesian Goal Inference for Boundedly Rational Planning Agents. In *Advances in Neural Information Processing Systems (NIPS)*.

Zhuo, H. H.; Zha, Y.; Kambhampati, S.; and Tian, X. 2020. Discovering Underlying Plans Based on Shallow Models. *ACM Transactions on Intelligent Systems and Technology*, 11(2).