Model-Based Relational RL When Object Existence is Partially Observable

Ngo Anh Vien VIEN.NGO@IPVS.UNI-STUTTGART.DE
Machine Learning and Robotics Lab, University of Stuttgart, 70569 Germany

Marc Toussaint MARC.TOUSSAINT@IPVS.UNI-STUTTGART.DE
Machine Learning and Robotics Lab, University of Stuttgart, 70569 Germany

Abstract

We consider learning and planning in relational MDPs when object existence is uncertain and new objects may appear or disappear depending on previous actions or properties of other objects. Optimal policies actively need to discover objects to achieve a goal; planning in such domains in general amounts to a POMDP problem, where the belief is about the existence and properties of potential not-yet-discovered objects. We propose a computationally efficient extension of model-based relational RL methods that approximates these beliefs using discrete uncertainty predicates. In this formulation the belief update is learned using probabilistic rules and planning in the approximated belief space can be achieved using an extension of existing planners. We prove that the learned belief update rules encode an approximation of the exact belief updates of a POMDP formulation and demonstrate experimentally that the proposed approach successfully learns a set of relational rules appropriate to solve such problems.

1. Introduction

In realistic robotic domains, objects of importance are more often than not hidden from view—in drawers, behind doors, within clutter—and efficient reasoning and planning becomes challenging. These problems are inherently partially observable, both with respect to the number of objects physically present in the area (and accordingly the cardinality of the state space) and with respect to the properties or relations of potentially discoverable objects. Optimal policies need to reason about actively seeking out objects, which is a form of belief planning when modeled as a POMDP. We aim to both reason efficiently in such partially observable domains and learn models of these domains using model-based Reinforcement Learning.

Several existing formalisms can represent such domains, including general BLOG (Milch et al., 2005) and first-order POMDPs (Sanner & Kersting, 2010; Wang & Khardon, 2010). However, efficient learning with such frameworks remains elusive and is known to be particularly hard even in ordinary POMDPs.

We instead propose the use of uncertainty predicates to approximate the belief state, including the belief over the existence of objects, and learning probabilistic rules that model their dynamics. This approach is able to express and learn the probability of object discovery (or disappearance) depending on the current (belief) state and chosen action, including probabilities over predicates (object properties and relations) of the to-be-discovered object. Planning on this uncertainty predicate representation will entail active learning strategies for seeking out and discovering objects.

Concretely, our approach builds on existing methods for learning sets of Noisy Deictic Rules (NDR) rules (Pasula et al., 2007) and stochastic planners such as UCT (Kocsis & Szepesvári, 2006) or the PRADA planner presented in (Lang & Toussaint, 2010). However, to our knowledge none of the existing learning and planning methods can cope explicitly with uncertainty over the existence of objects.

Our contributions are three-fold. First, for every ordinary predicate P of the domain we introduce a binary uncertainty predicate *ucP* which expresses whether the value of P is known. In addition, we introduce the special unary predicate $\mathit{Exists}(X)$ (does object X exist?) and the respective uncertainty predicate $\mathit{ucExist}(X)$ (do we know whether X exists?) to approximate the belief over object existence. The uncertainty predicates are similar to the 3-valued structures (Sagiv et al., 2002) from the work of Srivastava et. al. (Srivastava et al., 2009). Alternatively, representing degrees of belief using integer valued uncertainty predicates would be a straightforward extension (all used

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

methods can cope with interger-valued predicates), but the additional degrees of freedom are ultimately unnecessary for the domains that we are interested in here. Second, rather than assuming full observability as in (Pasula et al., 2007), our approach addresses partial observability by introducing a *belief augmentation* of the observed data. This assumes that *in retrospect* we can augment the observed data with uncertainty predicates to form consistent data which allows us to learn rules that predict both observations and belief state dynamics as a function of the current belief state. Third, existing literature on learning (Pasula et al., 2007) and planning (Lang & Toussaint, 2010) for this problem assumes unique covering rules. That assumption inherently contradicts the *ucExist*(X) predicate since it can be grounded with infinitely many hypothetical logical constants (since the concrete object ID of not-yet-discovered objects is arbitrary). We address this issue by relaxing the unique covering assumption for learning and planning.

We evaluate our Uncertainty Predicate Rule Learning (UPRL) using four examples. The first example illustrates the efficacy of our learner by comparing the model learnt by UPRL to an optimal POMDP solution found given oracle knowledge of the true propositional model. The second example demonstrates how UPRL generally learns rules explaining the birth-death process of creatures in nature. The third and fourth examples demonstrate (respectively on a 3D simulated blocksworld manipulator and a 3D simulated robot in an office environment) that UPRL is able to both efficiently learn a symbolic representation of a generative process for potential objects and find a (near-) optimal policy online that leverages that model.

2. Related Work

Several formalisms have been proposed to deal with uncertainty over the number of objects. Dietterich et. al. have been addressing *object invention* (Dietterich et al., 2008) as one of the future directions in the field of inductive logic programming (ILP). BLOG (Bayesian Logic) (Milch et al., 2005), its nonparametric extension (Carbonetto et al., 2005) and similar formalisms represent relational probabilistic models in stochastic worlds where the number of objects is unknown. The Church language (Goodman et al., 2008), a Lisp-like functional programming language, describes generative models as stochastic functions. While the above formalisms may be more general than what we propose, our aim is efficiency of learning and planning for typical problems that involve the discovery of objects.

The work by Srivastava et. al. (Srivastava et al., 2009) considers planning using abstractions for state aggregation to represent unknown object quantities using 3-valued structures (Sagiv et al., 2002). However, the planner is deterministic and cannot deal with uncertainty in learned transition models. Their method plans on state abstraction aggregated from problem instances having different numbers of objects. The predicates we will introduce for belief approximation are similarly discrete, but used within relational probabilistic rules.

POMDPs (Smallwood & Sondik, 1973) are known to be a general framework for planning under uncertainty. Its propositional algorithms have complexity of an exponential power of the time horizon and the number of observations. Relational representations in POMDPs, first-order POMDPs or relational POMDPs (Sanner & Kersting, 2010; Wang & Khardon, 2010) exploit lifted abstraction and use situations to represent a full history, which replaces an explicit belief representation. However, this representation is only efficient for planning. Learning such a POMDP model in an RL setting from training experiences has not been shown efficient.

The method of Amir (2005) learns action effects in partially observable STRIPS domains. However, this method and the probabilistic STRIP method (Blum & Langford, 1999) are less flexible compared to the NDR rules of (Pasula et al., 2007). They have not dealed with the noise effects and the change of the objects not listed in the action's arguments directly as NDRs do (Pasula et al., 2007).

In conclusion we believe there is a lack of practical methods to model-based relational RL in the face of uncertainty over object existence, as it naturally is the case when objects are hidden in drawers of behind doors. With our approach we aim to provide such a method.

3. Background

The problem of planning under uncertainty in relational worlds can be modeled as a relational Markov decision process (MDP) which is 5-tuple $\{\mathcal{C}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \mathcal{U}\}$; $\mathcal{C}$ is a set of objects; $\mathcal{P}$ is a set of predicates (relations) and functions over these objects; the action set $\mathcal{A}$; and $\mathcal{U}$ is a set of reward formulae. The state of the world is represented by all ground predicates and functions

$$\bigwedge_{p \in \mathcal{P}} \bigwedge_{c \in C(\mathcal{C}, |p|)} p(c) = * \tag{1}$$

where $*$ means any values in the predicate or function domain; $|p|$ is the arity of predicate or function p; and $C(\mathcal{C}, |p|)$ is the set of all combinations of length $|p|$ from $\mathcal{C}$.

In the work of Pasula et. al. (Pasula et al., 2007) transitions are represented as a set of Noisy Deictic Rules (NDRs). A NDR is given as

$$a_r(\mathcal{X}_a) : \Phi_r(\mathcal{X}) \rightarrow \begin{cases} p_{r,1} & : \Omega_{r,1}(\mathcal{X}) \\ & \vdots \\ p_{r,m_r} & : \Omega_{r,m_r}(\mathcal{X}) \\ p_{r,0} & : \text{noise} \end{cases} \tag{2}$$

where $\mathcal{X}, \mathcal{X}_a$ is a set of logical variables. All the NDRs in $\mathcal{R}$ must have logical variables as their arguments. Each rule r consists of an action a_r having $\mathcal{X}_a \subset \mathcal{X}$ as arguments; a context formula Φ; and a set of $m_r + 1$ different outcomes associated with probabilities $p_{r,i} \in [0, 1]$. The subset of logical variables $\mathcal{X} \setminus \mathcal{X}_a$ is called deictic references. The context Φ and outcomes $\Omega_{r,i}$ are conjunctions of predicates and functions of logical variables. The last outcome $\Omega_{r,0}$ is the noise. This essentially allows for stronger compression and better regularization of the rule set, see (Pasula et al., 2007) for details.

If all logical variables appearing in the rule are substituted by a substitution $\sigma : \mathcal{X} \mapsto \mathcal{C}$ (constants or concrete world objects), it is a *ground rule* $r(\sigma(\mathcal{X}))$. For each action a, we denote $\Gamma(a)$ as the set of all ground rules of the action a. A state s and an action a are covered uniquely by a ground rule if there is only one rule in $\Gamma(a)$ such that $s \models \Phi(\mathcal{X}) \wedge a = a_r$. For such a unique covering rule, the distribution $P(s'|s, a)$ to predict a next state of a state action pair (s, a) is computed as

$$P(s'|s, a) = \sum_{i=0}^{m_r} p_{r,i} P(s'|\Omega_{r,i}, s) \tag{3}$$

where the probability $P(s'|\Omega_{r,i}, s)$ is deterministic, which is one if s' is received by taking the changes of $\Omega(\mathcal{X})$ into s, otherwise zero. If this state action pair (s, a) is not covered by any unique rule, then its transition can apply the default rule.

Learning NDR sets is in general an NP-hard problem (Walsh, 2010). Pasula et. al. (Pasula et al., 2007) have proposed a supervised offline learning algorithm to learn NDRs. Given training data $E = \{(s, a, s')_i\}_{i=1}^{M}$, they do greedy search for a rule set that maximizes the likelihood of observing E subject to a penalty on the rules' complexity

$$S(\Gamma) = \sum_{(s,a,s') \in E} \log(P(s'|s, r_{(s,a)})) - \alpha \sum_{r \in \Gamma} PEN(r)$$

where $r(s, a)$ is the rule covering the state-action pair (s, a); $P(s'|s, r_{(s,a)})$ is defined as in Eq. (3); and the function $PEN(\cdot)$ is the complexity term which penalizes the complex rules. The greedy search algorithm consists of three search levels. The outermost level, called *LearnRules*, searches through all possible rule sets to create new rule sets. The middle level is *InduceOutcomes* which would fill possible outcomes into the given incomplete rule. The inner most level is *LearnParameters* which learns the probability distribution of the given incomplete rule returned from the middle level.

Given a set of learned NDRs, planning can be realized using UCT (Kocsis & Szepesvári, 2006) or PRADA (Lang & Toussaint, 2010). The latter translates the learned rules into a graphical model to evaluate action sequences using approximate inference.

4. Uncertainty Predicates as Belief Approximation in Relational POMDPs

We first give a motivating example and briefly discuss the inefficiency of a full but propositional POMDP formulation. We then introduce uncertainty predicates that can be used efficiently in a model-based relational RL setting.

4.1. Example

Consider a world in which there is initially one closed box on a table. The robot's *observation* of the world might be

$$\mathbf{y}^0 = \{table(01), box(02), on(02, 01), closed(02)\} .$$

Assuming that there is a block hidden in box 02, the robot's observation after taking action $open(02)$ might be

$$\mathbf{y}^1 = \{table(01), box(02), block(03), on(02, 01), contains(02, 03), \neg closed(01)\} .$$

A naive application of Pasula et. al.'s NDR-rule learning will fail on this data. If one assumes that the set of constants $\mathcal{C}$ changed from $\mathcal{C} = \{01, 02\}$ in time slice $t = 0$ to $\mathcal{C} = \{01, 02, 03\}$ in time slice $t = 1$ (that is, the object really appeared and the state space's cardinality expanded), NDRs do not apply as they can not create any contexts. The alternative case, assuming $\mathcal{C} = \{01, 02, 03\}$ also for $t = 0$, directly leads to a partial observability problem as the values of all predicates associated to 03 (including type and location, etc) are not observed at $t = 0$. A heuristic approach might try to set all these unobserved predicate values to negative because this is the only option to ensure the learned NDRs will have the positive valued outcome predicates ($block(03)$ and $contains(02, 03)$) as part of the outcome prediction (NDRs only include changed predicate values in their outcomes, as motivated by the frame assumption (Pasula et al., 2007)). This however would also turnout infeasible: In modeling the data's substitution $\sigma = \{X \to 01, Y \to 02\}$ would be created. The deictic reference in this case consists of the constant $\{03\}$. Thus, relying on the changed predicates an incomplete rule with the context

$$open(X) : closed(X), \neg block(Y), \neg contains(X, Y)$$

would be learned. However, this rule is not uniquely covering the state y^0, because both substitutions $\{X = 02; Y = 01\}$ and $\{X = 02; Y = 03\}$ can hold, rendering the approach infeasible. If there are more objects concurrently appearing, there are even more possible substitutions.

Apart from these problems in directly applying Pasula et. al.'s method, such an approach would not yield probabilistic transition models that would allow for belief planning, that is planning to reduce uncertainty as it is typical for POMDPs, because such uncertainties are not explicitly represented.

A POMDP formulation realizes such belief planning ide-

ally. In a *propositional* setting we may assume that batch data is given and we therefore know the maximal number of objects that have been appearing in this data. We may choose $\mathcal{C}$ as this constant superset for all t. Propositionalizing the data we could try to learn transition models and use standard POMDP planners. However, assuming that the state is represented by a set of predicates $\mathcal{P}$, the state and observation spaces are very large, $O(2^{(|\mathcal{P}||\mathcal{C}|)^2})$, with the assumption of maximally 2-ary predicates. The number of entries to store $\mathcal{T}$ and $\mathcal{O}$ is $O(2^{(|\mathcal{P}||\mathcal{C}|)^{2^2}})$ each. One of our experiments will neglect the learning problem (assume that the correct propositional POMDP is given) and compare the optimal POMDP policy with our approach.

However, the propositional POMDP formulation does not generalize easily to relational settings, let alone efficiently learning transition models in relational RL settings. Our approach below will introduce uncertainty predicates that bridge between NDRs and the POMDP formulation.

4.2. Uncertainty Predicates

First, we introduce an auxiliary binary-valued predicate $\mathit{Exists}(X)$ that indicates certain existence or non-existence of an object. It will play a necessary role in deictic reference resolution in the NDRs and the learning of these rules, as described below.

Further, for every predicate that may not be observable by the robot we introduce an uncertainty counterpart: In our example domain the potential non-observed predicates are $\mathit{Exists}(X)$, $block(X)$, $box(X)$, $contains(X,Y)$, $\cdots$, for which we introduce the uncertainty predicates $\mathit{ucExist}(X)$, $\mathit{ucBox}(X)$, $\mathit{ucBlock}(X)$, $\mathit{ucContain}(X,Y)$, $\cdots$ (where uc abbreviates *uncertain*). In comparison to (Sagiv et al., 2002), a true uncertainty predicate corresponds to the 1/2 logic value, where a false uncertainty predicate together with the values of the original predicates ($\mathit{Exists}(X)$ and $contains(X,Y)$) represent the 0 or 1 logical values of (Sagiv et al., 2002).

To give an example, after augmenting the data with the introduced uncertainty predicates (discussed in detail below), the above transition might look like $(y^0, open(02), y^1)$, where

$\mathbf{y}^0 = \{table(01), box(02), on(02,01), closed(02), \mathit{Exists}(01), \mathit{Exists}(02), \mathit{ucContain}(02,03), \mathit{ucExist}(03), \mathit{ucBox}(03), \mathit{ucBlock}(03), \mathit{ucTable}(03)\}$

$\mathbf{y}^1 = \{table(01), box(02), block(03), on(02,01), \mathit{Exists}(01), \mathit{Exists}(02), \mathit{Exists}(03), contains(03,02), \neg\mathit{ucContain}(02,03), \neg\mathit{ucBox}(03), \neg\mathit{ucBlock}(03), \neg\mathit{ucTable}(03), \neg closed(02)\}$

which is in a new introduced form of dynamics representation as in Eq. (4).

$$a_r(\mathcal{X}_a) : \Phi_r(\mathcal{X}, \mathcal{X}_{new}) \rightarrow \Phi'_r(\mathcal{X}, \mathcal{X}_{new}) \tag{4}$$

where $\mathcal{X}_{new}$ is a new set of constants appearing at next time slice. We expect that the learned rule explaining this experience might look like

$$open(X) : box(X), closed(X), \mathit{ucContain}(X,Y), \mathit{ucExist}(Y) \rightarrow \begin{cases} 1.0 : \mathit{Exists}(Y), block(Y), contains(X,Y), \\ \neg closed(X), \neg\mathit{ucContain}(X,Y), \neg\mathit{ucExist}(Y) \end{cases}$$

The only feasible substitution is $\sigma = \{X = 02, Y = 03\}$ and therefore this rule uniquely covers (y^0, a). When there is more than one object appearing the learned rule has two new skolem constants in its outcomes. For instance, in the case of two appearing objects the learnt rule's context contains two similar predicates $\mathit{ucExist}(Y)$ and $\mathit{ucExist}(Z)$.

$$open(X) : box(X), closed(X), \mathit{ucContain}(X,Y), \mathit{ucContain}(X,Z), \mathit{ucExist}(Y), \mathit{ucExist}(Z) \rightarrow \begin{cases} 1.0 : \mathit{Exists}(Y), block(Y), contains(X,Y), \\ \mathit{Exists}(Z), block(Z), contains(X,Z), \\ \neg closed(X), \neg\mathit{ucContain}(X,Y), \\ \neg\mathit{ucContain}(X,Z), \neg\mathit{ucExist}(Y), \\ \neg\mathit{ucExist}(Z) \end{cases}$$

with the augmented data as

$\mathbf{y}^0 = \{table(01), box(02), on(02,01), closed(02), \mathit{Exists}(01), \mathit{Exists}(02), \mathit{ucContain}(02,03), \mathit{ucContain}(02,04), \mathit{ucExist}(03), \mathit{ucExist}(04), \cdots\}$

It has two substitutions $\{X = 02, Y = 03, Z = 04\}$ and $\{X = 02, Y = 04, Z = 03\}$. As we can see, the difference of two substitutions is at the double substitution of new appearing objects. Therefore, if in a learning algorithm we could check the substitution with respect to appearing objects, then the unique covering problem can be handled. We will make this change explicitly in the learning algorithm of (Pasula et al., 2007) how to handle the multiple substitutions related to new appearing constants.

5. Learning State and Belief Transitions

In this section, we describe how to extend the learning algorithm of Pasual et. al. (Pasula et al., 2007) which can not handle the multiple substitutions related to new appearing constants. There are two significant changes. First, we introduce a preprocessing step, called belief augmentation, which augments each data point having new appearing objects by adding uncertainty predicates to the previous state: e.g. *Exists*, *ucExist*, *ucContain*, *ucBox*, *ucBlock*, $\cdots$. Second, we need a major change in the learning algorithm of Pasual et. al. (Pasula et al., 2007) in how to handle the multiple substitutions related to new appearing constants.

5.1. Belief Augmentation of the Data

Belief Augmentation denotes a preprocessing step where we add uncertainty predicates to the observed data as described in Algorithm 1. Starting with a data set $D = \{(y_t, a_t, y_{t+1})\}_{t=0}^{N-1}$, for each data point, the algorithm checks if there are new objects appearing ($\mathcal{X}_{new} \neq \emptyset$). To

allow for a deictic reference to potentially appearing objects in the rule context, uncertainty predicates are added to the previous state: *ucP*(c), where *ucP* are all uncertainty predicates of constants c if $c \wedge \mathcal{X}_{new} \neq \emptyset$. After Belief Augmentation we have an augmented dataset $\mathcal{D} = \{(y_t^{\text{aug}}, a_t, y_{t+1})\}_{t=0}^{N-1}$.

Algorithm 1 beliefAugmentation Algorithm

Input: dataset $\mathcal{D} = \{(y_t, a_t, y_{t+1})_t\}_{t=0}^{N-1}$
for $t = 0$ **to** $N-1$ **do**
  % calculate the set of objects appearing.
  Calculate $\mathcal{X}_{new}^t = \mathcal{X}^{t+1} \setminus \mathcal{X}^t$
  if $\mathcal{X}_{new}^t \neq \emptyset$ **then**
    for each $c \in \mathcal{X}_{new}^t$ **do**
      Add *ucExist*(c) to $\Phi^t(\mathcal{X}^t)$
    end for
    for each *ucP* predicates and $c \wedge \mathcal{X}_{new}^t \neq \emptyset$ **do**
      Add *ucContain*(c) to $\Phi^t(\mathcal{X}^t)$
    end for
  end if
end for

Note that in the augmented data, within each data point $(y_t^{\text{aug}}, a_t, y_{t+1})$, y_{t+1} and y_t^{aug} have the same set of constants $\mathcal{C}_{t+1}$. This means that any predicates having new appearing objects in their list of arguments are considered as uncertain or unobserved in the previous state. However, y_t and y_t^{aug} (as well as their respective data triplets) have different sets of constants $\mathcal{C}$. This inherently requires a first order learning method to deal with the augmented data.

5.2. Learning

The original algorithm of Pasula et. al. (Pasula et al., 2007) requires that there is a *unique* covering of rule contexts. However, in our case the choice of constants c that we associate with an appearing object (or multiple appearing objects) is fully arbitrary: we can assign any constant to the appearing objects. Therefore, the deictic reference to a *ucExist*(X) in a context can never be *uniquely* covered. However, as any choice of covering is exactly equivalent we may arbitrarily choose a covering, e.g. the next smallest not yet used constants. To realize this in the greedy rule learning algorithm we need to modify each search operator.

Theorem 1 proves that the model learnt by UPRL encodes the belief updates of the general POMDP formulation.

Theorem 1 *The learnt model of UPRL approximates the belief transition in the POMDP formulation for the problem of uncertainty over existence of objects.*

Proof: For a sketch, we compute the next state and belief update of both methods, then compare them.

Step 1: In the POMDP formulation the belief over observed predicates has value 1.0 and over unobserved predicates is 0.5. The belief $b(p(c))$ has a value of 1.0 (0.0 for negative value of c) if c is the set of constants previously observed. $b(p(c)) = 0.5$ if c contains at least one unobserved constant. The belief update is in general

$$b'(p(c)) \propto \mathcal{Z}(o|s',a) \sum\nolimits_{s(p(c))} T(s'|s,a) b(p(c)) + \sum\nolimits_{s(\neg p(c))} T(s'|s,a) b(\neg p(c))$$

where $s(p(c))$ means all states consisting of positive predicates $p(c)$. We investigate two cases:

(**a**.) In the case of previously non-appeared predicates

$$\begin{aligned} b'(p(c)) &\propto \mathcal{Z}(o|s',a) \sum\nolimits_{s(p(c))} T(s'|s,a) \times 0.5 \\ &\quad + \sum\nolimits_{s(\neg p(c))} T(s'|s,a) \times 0.5 \\ &= 0.5 \times \mathcal{Z}(o|s',a) \sum\nolimits_{s \in \mathcal{S}} T(s'|s,a) \end{aligned}$$

There are two possibilities:

(**a.1**) From *non-appeared* $\rightarrow$ *appeared*: $\mathcal{Z}\big(o(p(c))|s',a\big) = 1.0$, then $b'(p(c)) = 1.0$, $b'(\neg p(c)) = 0.0$ after normalizing.

(**a.2**) From *non-appeared* $\rightarrow$ *non-appeared*: $\mathcal{Z}\big(o(p(c))|s',a\big) = 0.0$, then $b'(p(c)) = b'(\neg p(c)) = 0.0$ after normalizing which are still considered as uniform probability.

(**b**.) The belief update of already appeared objects is deterministic as (assuming that $b(p(c)) = 1.0$)

$$\begin{aligned} b'(p(c)) &\propto \mathcal{Z}(o|s',a) \sum\nolimits_{s(p(c))} T(s'|s,a) \times 1.0 \\ &\quad + \sum\nolimits_{s(\neg p(c))} T(s'|s,a) \times 0.0 \\ &= \mathcal{Z}(o|s',a) \sum\nolimits_{s(p(c))} T(s'|s,a) \end{aligned}$$

There are two possibilities:

(**b.1**) From *appeared* $\rightarrow$ *appeared*: $\mathcal{Z}\big(o(p(c))|s',a\big) = 1.0$, then $b'(p(c))$,$b'(\neg p(c)) = 0.0$ after normalizing.

(**b.2**) From *appeared* $\rightarrow$ *disappeared*: $\mathcal{Z}\big(o(p(c))|s',a\big) = 0.0$, then $b'(p(c)) = b'(\neg p(c)) = 0.0$ after normalizing which are again considered as uniform probability.

Step 2: Assuming that we have learned NDRs successfully in the sense that every state-action pair has one unique covering rule. We study its outcomes in four corresponding cases as in the POMDP formulation:

– From *non-appeared* $\rightarrow$ *appeared*: The predicates $ucExist$ and other uncertainty predicates of the appearing objects become false, and all predicates of the appearing objects have values (true or false) with probability one. This corresponds to the case (a.1).
– From *non-appeared* $\rightarrow$ *non-appeared*: The rule does not affect neither non-appearing objects nor their uncertainty predicates. Therefore, it corresponds to the case (a.2).

– From *appeared* → *appeared*: The predicate's value changes as in conventional NDRs. This is exactly representing the deterministic belief transition of the corresponding case (b.1).
– From *appeared* → *disappeared*: The predicates $ucExist$, and other uncertainty predicates of the appearing objects become true, all predicates of this appearing objects are back to false. This represents the transition of certainty to uncertainty in case (b.2).

Comparing two methods, we receive the identical match between POMDP's belief update and the learnt NDR's approximate belief transition. □

6. Planning Using the Learned Models

In this section, we propose general modifications applicable to planning algorithms such as PRADA (Lang & Toussaint, 2010), UCT (Kocsis & Szepesvári, 2006), SST (Kearns et al., 2002) to enable planning with our learnt model. First we clarify how to convert the learnt NDR rules to DBNs, for which we assume to have the superset of all constants $\mathcal{C}$.

Given the state $\mathbf{s}$, we can compute the probability of each rule context,

$$P(\phi|\mathbf{s}) = \prod_{i=1}^{K} P(\phi_i|\mathbf{s}_{\pi(\phi_i)}) \,, \tag{5}$$

where the function $\pi(\phi_i)$ gives a set of predicates in $\mathbf{s}$ contained in the context of ϕ_i. Similarly to the unique covering issue detailed above, the binary random variables of several rule contexts (referring to different constants) having *ucExist* predicates might hold true. However, by construction of the DBN in (Lang & Toussaint, 2010), the random variable R would choose only one covering rule. We therefore propose to modify Eq. (16) in (Lang & Toussaint, 2010) to choose the first of them as in Eq. (6),

$$P(R=r|a,\phi) = I\Big(r \in \Gamma(a) \wedge r = \Gamma_0^*(a) \wedge \Phi_r = 1 \bigwedge_{r' \in \Gamma(a)\setminus\Gamma^*(a)} \Phi_{r'} = 0\Big) \tag{6}$$

where $\Gamma^*(a) \subset \Gamma(a)$ is the set of coverings which might be multiple with respect to the *ucExist* predicates, and $\Gamma_0^*(a)$ is its first element. If that is the action having no *ucExist* in its context, then $\Gamma^*(a) = r$ which yields the original probability in Eq. (16) in (Lang & Toussaint, 2010). This extended PRADA basically mitigates multiple coverings with respect to uncertainty predicates.

As PRADA is an action sampling technique, it cannot deal with uncertainty of observation. We propose an online planning algorithm UPRL+P, which uses our extended PRADA technique as a search daemon in UCT. For each state, we run our extended PRADA to find the best action.

UCT, SST can also be applied directly on our learnt NDRs, the needed change which is similar to PRADA is to choose one representative rule if there are more than one covering rule having different $ucExist(c)$ predicates.

7. Experiments

The first two experiments are toy domains to test the optimality of policies derived from the learned rule sets and the power of UPRL to learn complex dynamics of appearing and disappearing entities. The third and fourth experiment concern more challenging robotic domains.

7.1. Illustrative Example: POMDP vs. uncertainty predicates

Here we first consider a propositional example to illustrate how UPRL does belief approximation and how the policy derived from a *learned* model compares to the optimal policy for the *true* propositional POMDP model. We use the true model to generate data, then UPRL learns a NDR set using 20 training examples generated from the true model and does planning on the learned model.

The domain consists of two closed boxes b_1 and b_2 and one block c which might reside in one of the two boxes. We use the following six predicates: $box(X)$, $block(X)$, $closed(X)$, $empty(X)$, $contains(X,Y)$, and two action predicates: $open(X)$, $grab(X)$. The task is to find the object, then grab it, and conceptually similar to the Tiger Problem. The state space is given by the set of the nine grounded predicates, $\mathcal{P} = \{box(b_1), box(b_2), block(c), closed(b_1), closed(b_2), empty(b_1), empty(b_2), contains(b_1,c), contains(b_2,c)\}$; the action space by three grounded action predicates, $\mathcal{A} = open(b_1), open(b_2), grab(c)$. The robot's starting observation is

$$\mathbf{y}^0 = \{box(b_1), box(b_2), closed(b_1), closed(b_2)\}$$

The unobservable predicates are $block(X)$, $empty(X)$, $contains(X,Y)$, for each of which uncertainty predicates are introduced.

We use SARSOP (Kurniawati et al., 2008) to find an optimal policy for the propositional POMDP formulation. The reported time of SARSOP includes offline planning and online evaluation.

In contrast to the POMDP, UPRL learns a model of the environment as well as of the approximate belief transitions from data. We found that with sufficient data UPRL will reliably learn three correct rules for the $open$ action that distinguish three situations: opening a box when being uncertain about the box's content, opening a box being certain that it contains an object, opening a box when being certain it is empty.

The results for UPRL+SST, UPRL+UCT and UPRL+P

Table 1. Two box problem: The number of actions needed have the block in the hand, and the trial time in each episode. The results are averaged over 40 runs. The standard deviation of the average value is reported.

Algorithm	Actions	Trial Time (s)
SARSOP	2.50 ± 0.037	< 0.01
UPRL+SST	2.57 ± 0.105	3.94 ± 0.154
UPRL+UCT	2.51 ± 0.071	0.09 ± 0.002
UPRL+P	2.50 ± 0.069	0.07 ± 0.002

are reported in Table 1. All three planners use the UPRL-learned rules and lead to optimal policies. The trial time subsumes the time required to re-plan the next action in each step.

7.2. Learning Rule Sets for Complex Appearance and Disappearance Dynamics

In order to test the generality of UPRL we considered a more complex domain of stochastically appearing and disappearing entities and test whether UPRL can learn the correct dependencies of object (dis-)appearance, which was previously described in (Vien & Toussaint, 2013). To make it more illustrative we describe this process as an evolution of bacteria: Every bacteria has one of three *colors* (*black, red, blue*) which can change with a probability of p_c uniformly to a different color. Two bacteria can make a pair with a probability δ if their colors are blue. If a pair is established, they can spawn a new bacteria with a probability p_s. Every bacteria can die with a probability of p_d. The evolution starts with a set of random bacteria. At each time step of evolution, bacteria can change colors, die, create pairs, and spawn. We show that our proposed learning algorithm, UPRL, can correctly learn such a relational rule model. The predicates used for state representation are: $Black(X), Red(X), Blue(X)$ for bacteria colors, $Pair(X, Y)$ to tell whether two bacteria make a pair, $Exist(X)$. The process events are represented as action predicates $eColor(X), eDeath(X), ePair(X, Y), eSpawn(X, Y)$, which evolve the color, the kill a bacteria, decide whether to make a pair, and spawn an offspring.

We use the true model with a setting $\{\delta, p_c, p_d, p_s\}$=$\{0.7, 0.2, 0.3, 0.7\}$ to generate training data of variable size and evaluate the learnt rule set by measuring the *variational distance* between the true model P and the learnt model $\tilde{P}$, $\delta(P, \tilde{P}) = \frac{1}{|E|} \sum_{e \in E} |P(e) - \tilde{P}(e)|$. We compare three learning methods: UPRL, propositional UPRL (where rules have only constants), and the original learning algorithm of Pasula et. al. (Pasula et al., 2007) using data without *BeliefAugmentation*. The performance of UPRL compared to others is given in Fig. 1, which significantly shows the importance of using both the variable abstractions and the uncertainty predicates in order to capture the innate uncertainty over existence of objects. The learnt rules of Pasula's method could not explain data points having new appearing constants and therefore learned to predict noise outcomes, explaining the large distance to the true probability. The propositional method was too slow to test for large training data.

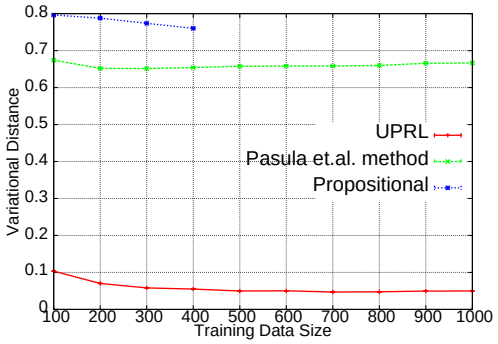

Figure 1. Variational distance between the learned model $\tilde{P}$ and the true model P for varying training data size. The results are averaged over 10 trials and include very small error bars. The test data set is of size 500. The propositional method runs with more than 400 training examples have not stopped after 1 day.

7.3. Simulated Robot Manipulation

We use the simulator of Lang and Toussaint (Lang & Toussaint, 2010) (using the physics simulation ODE internally) to simulate realistic behaviour of the objects. We define four actions: $open(X)$, $close(X)$, $grab(X)$ and $puton(X)$. All actions have only one argument to specify which object to be manipulated. The environment's state is represented with predicates $on(X, Y)$, $contains(X, Y)$, $closed(X)$, $inhand(X)$, $block(X)$, $box(X)$ and $table(X)$ which are primitive concepts. To evaluate the UPRL-learned action model, we use the three planners as above.

Using UPRL, the NDR set is learnt from a training set of 200 experience triples (y, a, y'), generated in a world of two boxes containing two cubes each, two boxes having only one cube each and two empty boxes by performing random actions with a small bias to build high stacks. This would yield a uniform probability of maximally two objects able to appear. Initially all boxes are closed.

We investigate two task variants in this domain: *high towers* and *magic box*, as described below. In both variants URPL+P uses a horizon $d = 4$ and $N = 200$ sample action sequences in PRADA. The settings of SST are: the horizon $d = 3$, and the branching factor $b = 2$. For SST, we tried increasing both d and b, however the simulation did not finish after two days. The settings of UCT are: the horizon $d = 4$, the bias parameter $c = 1.0$ is the best choice among those we have experimentally tested and the number of sampling is $N = 200$. All three algorithms use a discount factor $\gamma = 0.95$.

Table 2. High Towers. Success rates, actions and taken time for each trial are averaged over 50 runs different random seeds and environments. The error is the standard deviation of the mean estimator itself.

Rew.	Alg.	Succ.	Actions	Time (s)
10	UPRL+SST	1.0	20.2 ± 0.6	3754 ± 23.6
10	UPRL+UCT	1.0	$\mathbf{18.4 \pm 0.5}$	204 ± 15.0
10	UPRL+P	0.94	31.8 ± 0.6	275 ± 11.1
14	UPRL+SST	1.0	$\mathbf{29.6 \pm 1.4}$	4314 ± 24.7
14	UPRL+UCT	0.86	30.1 ± 1.6	378 ± 16.9
14	UPRL+P	0.82	32.3 ± 1.2	392 ± 12.3

Table 3. The reported number of success rates, actions and taken time for each trial are averaged over 50 runs (20 runs in office environment).

Prob.	Alg.	Succ.	Actions	Time (s)
Magic	UPRL+SST	0.0	50.0 ± 0.0	$\sim$2 hours
Magic	UPRL+UCT	1.0	$\mathbf{30.9 \pm 0.9}$	165 ± 6.79
Magic	UPRL+P	1.0	$\mathbf{30.9 \pm 1.0}$	195 ± 6.29
Office	UPRL+SST	0.0	60.0 ± 0.0	$>$ 1 hour
Office	UPRL+UCT	0.34	52.0 ± 4.24	599 ± 58.7
Office	UPRL+P	1.0	$\mathbf{23.1 \pm 2.97}$	251 ± 32.4

7.3.1. High Towers

In the *high towers* domain, described in (Pasula et al., 2007), the reward is the average height of objects. We considered six closed boxes, each of which contains either two objects, one object, or is empty. This was an easy planning problem if the robot knew where the blocks are, because there are many possible policies able to achieve the high reward. We use two reward thresholds to stop online planning: 10 and 14. To achieve reward 10, the robot would not need to open all boxes, 4 to 5 blocks (depending on whether they fall down) is sufficient to achieve this reward. To achieve reward 14, the robot needs to discover almost all possible objects in six boxes. The performance of three algorithms is reported in Table 2.

The high success rates and the reasonably small number of needed actions tell that all three algorithms not only can disambiguate uncertainty to discover objects, but also quickly select important initial actions (open closed boxes). As expected, the computation time of SST is much higher than for the others.

7.3.2. Stacking With Magic Boxes

This task rewards building a pile of blocks of a specific height, but additionally assumes some boxes which are *magic*: Whenever the robot opens a closed magic box, new objects might spawn. We show that UPRL can still learn a correct model of the environment. We consider an environment with three boxes (one is empty, one has one block, and one has *magic* property), while the robot has to build a 4-block pile. The results are shown in Table 3. Only SST failed in discovering the magic box.

7.4. Office Environment

Finally we consider an office environment in which the robot needs to find a hammer and nails. Objects reside in different closets and at different locations. Therefore, this task has many interesting uncertainties: uncertainty over types, locations and existence of objects. In addition to all the predicates described in the blockworld domain, we have two more predicates: the action $goto(X)$ brings the robot near to a desired object with a probability of 0.8, and stays unmoved with probability of 0.2; the predicate $locatedAt(X)$ indicates whether the robot is near an object X. Closets are represented exactly as boxes above and $isHammer$ and $isNail$ are additional predicates.

Learning setting: We put six closets in an office: one is empty, one has only one hammer, one has two hammers, one has a hammer and a nail, one has two nails, and one has only one nail. This setting is consistent with the uniform initial belief of the robot over all possibilities, subject to a maximum two objects within a closet.

Planning setting: We test the learnt model by creating an environment consisting of four closets: a closet of a hammer, an empty closet, a closet of two nails and a closet of one nail. The task of the robot is to look for a hammer and a nail and put them on the table. SST has settings of $b = 2, d = 3$. UCT and UPRL+P has a horizon $d = 6$ and the number of sampling is $N = 1000$. The performance of three algorithms are reported in Table 3. As the goal is reached by not many possible policies in such case UCT often has bad performance, UPRL+UCT turns out to be much worse than UPRL+P.

8. Conclusion

We have proposed UPRL+P, a framework for model-based relational RL when object existence in only partially observable and objects may appear or disappear. By introducing uncertainty predicates we can represent a belief approximation of the respective relational POMDP. Our belief augmentation of the observed data allows for learning probabilistic rules that predict action effects depending on the belief state (including predictive probabilities of object appearance/disappearance and their properties/relations) as well as the belief dynamics itself. A crucial technical detail was to relax the unique covering assumption made by previous learning and planning methods. We also compared UPRL+P with optimal POMDP models and policies in a propositional setting. The experiments confirmed the efficiency of the approach to deal with complex stochastic processes of appearing and disappearing entities, as well as manipulation domains that require searching for hidden objects.

Acknowledgments: This work is funded by the DFG (German Research Foundation) within SPP 1527, Autonomous Learning.

References

Amir, Eyal. Learning partially observable deterministic action models. In *IJCAI*, pp. 1433–1439, 2005.

Blum, Avrim and Langford, John. Probabilistic planning in the graphplan framework. In *ECP*, pp. 319–332, 1999.

Carbonetto, Peter, Kisynski, Jacek, de Freitas, Nando, and Poole, David. Nonparametric Bayesian logic. In *UAI*, pp. 85–93, 2005.

Dietterich, Thomas G., Domingos, Pedro, Getoor, Lise, Muggleton, Stephen, and Tadepalli, Prasad. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008.

Goodman, Noah D., Mansinghka, Vikash K., Roy, Daniel M., Bonawitz, Keith, and Tenenbaum, Joshua B. Church: a language for generative models. In *UAI*, pp. 220–229, 2008.

Kearns, Michael J., Mansour, Yishay, and Ng, Andrew Y. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 49(2-3):193–208, 2002.

Kocsis, Levente and Szepesvári, Csaba. Bandit based monte-carlo planning. In *ECML*, pp. 282–293, 2006.

Kurniawati, Hanna, Hsu, David, and Lee, Wee Sun. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

Lang, Tobias and Toussaint, Marc. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, 39:1–49, 2010.

Milch, Brian, Marthi, Bhaskara, Russell, Stuart J., Sontag, David, Ong, Daniel L., and Kolobov, Andrey. BLOG: Probabilistic models with unknown objects. In *IJCAI*, pp. 1352–1359, 2005.

Pasula, Hanna M., Zettlemoyer, Luke S., and Kaelbling, Leslie Pack. Learning symbolic models of stochastic domains. *J. Artif. Intell. Res. (JAIR)*, 29:309–352, 2007.

Sagiv, Shmuel, Reps, Thomas W., and Wilhelm, Reinhard. Parametric shape analysis via 3-valued logic. *ACM Trans. Program. Lang. Syst.*, 24(3):217–298, 2002.

Sanner, Scott and Kersting, Kristian. Symbolic dynamic programming for first-order pomdps. In *AAAI*, 2010.

Smallwood, Richard D. and Sondik, Edward J. The Optimal Control of Partially Observable Markov Processes Over a Finite Horizon. *Operations Research*, 21(5): 1071–1088, 1973.

Srivastava, Siddharth, Immerman, Neil, and Zilberstein, Shlomo. Abstract planning with unknown object quantities and properties. In *SARA*, 2009.

Vien, Ngo Anh and Toussaint, Marc. Reasoning with uncertainties over existence of objects. In *AAAI Fall Symposium: How Should Intelligence Be Abstracted in AI Research?*, 2013.

Walsh, Thomas J. *Efficient learning of relational models for sequential decision making*. PhD thesis, The State University of New Jersey, 2010.

Wang, Chenggang and Khardon, Roni. Relational partially observable mdps. In *AAAI*, 2010.