Journal of Machine Learning Research 13 (2012) 3691-3734 Submitted 12/10; Revised 12/11; Published 12/12

Exploration in Relational Domains for Model-based Reinforcement Learning

Tobias Lang TOBIAS.LANG@FU-BERLIN.DE
Marc Toussaint MARC.TOUSSAINT@FU-BERLIN.DE
Freie Universität Berlin
Machine Learning and Robotics Group
Arnimallee 7, 14195 Berlin, Germany

Kristian Kersting KRISTIAN.KERSTING@IAIS.FRAUNHOFER.DE
Fraunhofer Institute for Intelligent Analysis and Information Systems
Knowledge Discovery Department
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Editor: Satinder Baveja

Abstract

A fundamental problem in reinforcement learning is balancing exploration and exploitation. We address this problem in the context of model-based reinforcement learning in large stochastic relational domains by developing relational extensions of the concepts of the E^3 and R-MAX algorithms. Efficient exploration in exponentially large state spaces needs to exploit the generalization of the learned model: what in a propositional setting would be considered a novel situation and worth exploration may in the relational setting be a well-known context in which exploitation is promising. To address this we introduce relational count functions which generalize the classical notion of state and action visitation counts. We provide guarantees on the exploration efficiency of our framework using count functions under the assumption that we had a relational KWIK learner and a near-optimal planner. We propose a concrete exploration algorithm which integrates a practically efficient probabilistic rule learner and a relational planner (for which there are no guarantees, however) and employs the contexts of learned relational rules as features to model the novelty of states and actions. Our results in noisy 3D simulated robot manipulation problems and in domains of the international planning competition demonstrate that our approach is more effective than existing propositional and factored exploration techniques.

Keywords: reinforcement learning, statistical relational learning, exploration, relational transition models, robotics

1. Introduction

Acting optimally under uncertainty is a central problem of artificial intelligence. In reinforcement learning (RL), an agent's learning task is to find a policy for action selection that maximizes its reward over the long run. Model-based approaches learn models of the underlying transition process, usually formalized as Markov decision processes, from the agent's interactions with the environment. These models are then analyzed to compute optimal plans.

Generally, an agent has limited data from its interaction with an environment and its model only approximates the true dynamics therein. One of the key challenges in reinforcement learning is thus the exploration-exploitation tradeoff, which strives to balance two competing types of behavior of an autonomous agent in an unknown environment: the agent can either make use of its current model of the environment to maximize its cumulative reward (that is, to exploit), or sacrifice short-term rewards to gather information about the environment (that is, to explore) in the hope of increasing future long-term return by improving its model. This exploration-exploitation tradeoff has received considerable attention in unstructured, non-relational domains. There exist algorithms which define unique optimal solutions (such as Bayesian reinforcement learning, Poupart et al., 2006) or provably polynomial time solutions (E^3, Kearns and Singh, 2002, and R-MAX, Brafman and Tennenholtz, 2002; Strehl et al., 2009) to the exploration-exploitation trade-off problem. However, while they give a clear idea of how in principle exploration and exploitation can be organized, the basic algorithms in their original formulation only work on discrete enumerated state spaces. Therefore, we believe that the core scientific problem is not to find new exploration-exploitation theories, but how these principles can be realized on non-trivial representations (representations other than enumerated state spaces), and how the generalization and abstraction implicit in non-trivial representations interferes with these exploration-exploitation principles.

The environment of the agent typically contains varying numbers of objects with relations among them. Learning and acting in such large relational domains is a second key challenge in reinforcement learning. Relational approaches (Getoor and Taskar, 2007) can generalize information about one object to reach conclusions about other objects and thereby exploit the relational structure of natural environments. Such domains are hard—or even impossible—to represent meaningfully using an enumerated or a propositional state space. As an example, consider a hypothetical household robot which, after taken out of the shipping box and turned on, explores autonomously the environment in order to learn how to perform its cleaning chores. Without a compact knowledge representation that supports abstraction and generalization of previous experiences to the current state and potential future states, it seems to be hopeless for such a "robot-out-of-the-box" to explore one's home in reasonable time. For instance, after having opened one or two water-taps in bathrooms, the priority for exploring further water-taps in bathrooms, and also in other rooms such as the kitchen, should be reduced. Generalization over object types is crucial for any autonomous agent in realistic environments, but cannot be expressed in a propositional setting where every new object implies a new and therefore non-modeled situation.

The problem of exploration in stochastic relational worlds has so far received little attention. State-of-the-art relational reinforcement learning approaches (Džeroski et al., 2001; Driessens et al., 2006) are mostly model-free and use ε-greedy exploration which does not make use of relational knowledge. Exploiting the relational knowledge for exploration is the problem we address in the current paper. Applying existing, propositional exploration techniques is likely to fail: what in a propositional setting would be considered a novel situation and worth exploration may in the relational setting be an instance of a well-known abstract context in which exploitation is promising. In other terms, the key idea underlying our approach is: *The inherent generalization of learned knowledge in the relational representation has profound implications also on the exploration strategy.*

1.1 Our Approach

We first outline our approach so that we can better discuss related work afterwards. We present a general framework for model-based RL in relational domains. As is typical in model-based RL approaches like E^3 and R-MAX, our system will be composed of a *model learner*, a *planner* and a relational *exploration-exploitation strategy* that integrates both. We introduce a concrete instantiation in this setting where the learner and planner components are based on previous work: The learner is a relational learning algorithm of noisy indeterministic deictic (NID) rules (Pasula et al., 2007) which extracts a compact stochastic relational rule-based model from experience. As a planner we employ PRADA (Lang and Toussaint, 2010) which translates the learned relational model to a grounded dynamic Bayesian network (DBN) and uses approximate inference (a factored frontier) to estimate the expected return of sampled action sequences. Given a learner and a planner, the relational exploration-exploitation strategy needs to realize an estimation of novelty in the relational setting. The classical way to estimate state novelty is based on state (and action) visitation counts which, in an enumerated representation, directly reflect model certainty. We generalize the notion of state counts to relational count functions such that visitation of a single state increases this measure of knownness also for "related" states. What is considered related depends on the choice of features used to model these count functions: similar to density estimation with mixtures we assume count functions to be a mixture of basic (relational) features. The inherent generalization in these count functions thus depends on the choice of features. We propose several possible choices of such features in a relational setting, including one that exploits the specific relational context features that are implicitly learned by the relational rule learner.

Ideally, we would like the learner to fulfill guarantees in the KWIK (*knows what it knows*) framework (Li et al., 2011) and the planner to guarantee near-optimality (in our case, exact inference in the corresponding DBN). Clearly, our specific choices for the learner and planner do *not* fulfill these guarantees but target at being efficient in challenging applications as we demonstrate in the experimental section. Nevertheless, we will establish theoretical guarantees of our relational exploration-exploitation strategy under the assumption that we had a KWIK learner and near-optimal planning. This will allow us to draw clear connections *(i)* to the basic R-MAX and E^3 framework for exploration in reinforcement learning and *(ii)* to the pioneering work of Walsh (2010) on KWIK learning in a relational reinforcement learning setting.

Walsh's work proved the existence of a KWIK learning algorithm in a relational RL setting by nesting several KWIK algorithms for learning different parts of relational transition models. As Walsh points out himself, however, such an integrated KWIK learner, allowing provably efficient exploration, has never been realized or tested and would be "clearly approaching the edge of tractability" (Walsh, 2010). Further, his conceptual algorithm makes limiting assumptions on the model representation which are violated by the more general relational rule framework of Pasula et al. (2007). This is the reason why we choose a practically efficient but heuristic learner which has not been proven to be a KWIK learner. Similarly, it is clear that the computational complexity of *optimal* planning or exact inference in our corresponding DBN is exponential in the number of objects. Therefore, we choose a practically efficient approximate inference technique for planning in relational domains, as given by PRADA.

1.2 Related Work

The first studies on effective exploration in multi-state control problems developed a number of concepts for describing explorative behavior, including curiosity (Schmidhuber, 1991), seeking to minimize the variance of action value estimates (Kaelbling et al., 1996) and counters on the occurrences of states and actions (Thrun, 1992). Thereafter, efficient exploration solutions have been developed for propositional and continuous domains where the environment is represented as an enumerated or vector space. Bayesian reinforcement learning (Poupart et al., 2006) provides an optimal solution in a Bayesian framework by taking all potential models weighted by their posteriors into account at once. This solution is intractable in all but small problem settings, although there have been advances recently such as the near-Bayesian approach by Kolter and Ng (2009). An alternative approach to optimal exploration are algorithms studied in the probabilistically approximately correct (PAC) framework applied to Markov decision processes (MDPs) (so called PAC-MDP approaches). The seminal algorithms E^3 (Kearns and Singh, 2002) and R-MAX (Brafman and Tennenholtz, 2002; Strehl et al., 2009) execute near-optimal actions in all but a polynomial number of steps (or, in alternative problem formulations, only require a polynomial number of steps before they return a near-optimal policy for the current state). Despite the theoretical guarantees of these algorithms, in practice the required exploration steps are often unrealistically large. More importantly, these approaches are designed for enumerated representations of finite domains and thus difficult to apply in domains of everyday life involving many objects.

E^3 has been extended to parameter learning in factored propositional MDPs with a known structure (Kearns and Koller, 1999) and to Metric E^3 (Kakade et al., 2003) for state spaces where a metric allows to construct accurate local models. Both approaches assume the existence of efficient near-optimal planning algorithms. However, general algorithms with the required guarantees are not known. Supposedly, this is among the reasons why both approaches have not been empirically demonstrated: "it is thus unlikely that the Factored E^3 [for parameter learning] can ever be feasibly implemented" (Guestrin et al., 2002); similar statements can be made for Metric E^3. Therefore, Guestrin et al. (2002) propose an exploration strategy for factored propositional MDPs which is tailored towards a specific planning algorithm based on linear programming. The idea of characterizing relevant subspaces for exploration has been pursued in continuous domains using dimensionality reduction methods (Nouri and Littman, 2010). R-MAX has been extended to continuous MDPs with linearly parameterized dynamics (Strehl and Littman, 2007). All approaches discussed so far as well as other function approximation methods are propositional, that is, they do not generalize over object types. This can only be achieved by incorporating relational features (which subsume attributes of individual objects), which has not been pursued in the mentioned methods.

In recent years, there has been a growing interest in using expressive representations such as relational languages for reinforcement learning (RL) (Džeroski et al., 2001). While typical traditional RL approaches require explicit state and action enumeration, symbolic approaches seek to avoid explicit state and action enumeration through a symbolic representation of states and actions. Most work in this context has focused on model-free approaches (estimating a value function) and has not developed relational exploration strategies. Essentially, a number of relational regression algorithms have been developed for use in these relational RL systems such as relational regression trees (Džeroski et al., 2001) and Gaussian processes with graph kernels (Driessens et al., 2006). Kersting and Driessens (2008) introduce relational policy gradients. All of these approaches use some form of ϵ-greedy strategy to handle exploration; no special attention has been paid to the exploration-exploitation problem as done in the current paper. Driessens and Džeroski (2004) propose the use of "reasonable policies" in model-free relational RL to provide guidance, that is, to increase the chance to discover sparse rewards in large relational state spaces, also known as reward shaping. Sanner (2005, 2006) combines feature discovery with model-free relational reinforcement learning but does not discuss count function estimation for known states in the exploration-exploitation problem in general terms. Ramon et al. (2007) present an incremental relational regression tree algorithm that is capable of dealing with concept drift and showed that it enables a relational Q-learner to transfer knowledge from one task to another. They do not learn a model of the domain and again, relational exploration strategies were not developed. Generally, model-free RL approaches are not suited to realize the type of planned exploration as exemplified in R-MAX, E^3 or Bayesian RL. Croonenborghs et al. (2007) learn a relational world model online and additionally use lookahead trees to give the agent more informed Q-values by looking some steps into the future when selecting an action. Exploration is based on sampling random actions instead of informed exploration. Diuk (Diuk, 2010; Diuk et al., 2008) presents an algorithm for efficient exploration under certain assumptions in an alternative object-oriented representation of MDPs focused on object attributes. This representation does not account for noisy dynamics in realistic domains where actions may have a large number of low-probability effects. Efficient planning algorithms for this representation still need to be developed.

The pioneering work of Walsh (Walsh, 2010; Walsh et al., 2009; Walsh and Littman, 2008) provides the first principled investigation into the exploration-exploitation tradeoff in relational domains. His work lifts ideas from R-MAX and efficient RL algorithms for feature selection (such as the algorithms by Diuk et al., 2009) to relational domains. Walsh establishes sample complexity bounds for specific relational MDP learning problems which scale polynomially in the relational action operator descriptions and the number of objects—in contrast to the original R-MAX and E^3 which scale exponentially in the number of objects in relational domains due to the corresponding exponential state and action spaces. Walsh provides an evaluation for some of his algorithms in settings with 2-3 objects; however, his approaches for learning the effects of actions and, more importantly, for learning full action operators have not been demonstrated in practice. Despite their theoretical significance, it is uncertain whether an implementation of these algorithms is feasible; this might show an “inherent limitation of the KWIK-learning paradigm [Walsh’s learning framework, discussed below] and likely of online learning itself” (Walsh, 2010). Furthermore, to derive theoretical guarantees Walsh assumes a limited language to represent the learned model. Our approach will use the more expressive language of relational NID rules (Pasula et al., 2007), which are necessary to capture the dynamics in realistic domains, for which Walsh’s algorithms would not be applicable. We will detail the differences in representations in Section 3.4.

There is also an increasing number of (approximate) dynamic programming approaches for solving relational MDPs, see for example Boutilier et al. (2001), Kersting et al. (2004), Hölldobler et al. (2006), Wang et al. (2008) and Sanner and Boutilier (2009). In contrast to the current paper, however, their work assumes a given model of the world. Recently, Lang and Toussaint (2009) and Joshi et al. (2010) have shown that successful planning typically involves only a small subset of relevant objects or states. This can speed up symbolic dynamic programming significantly. A principled approach to exploration, however, has not been developed. Guestrin et al. (2003) calculate approximate value functions for relational MDPs from sampled (grounded) environments and provide guarantees for accurate planning in terms of the number of samples; they do not consider an agent which explores its environment step by step to learn a transition model.

The question of optimal exploration in model-based RL where we learn a transition model has similarities to the problem of active learning (Cohn et al., 1996). Epshteyn et al. (2008) investigate active RL in enumerated domains to focus the exploration on the regions of the state space to which the optimal policy is most sensitive. In statistical relational learning (Getoor and Taskar, 2007; de Raedt et al., 2008), which combines expressive knowledge representation formalisms with statistical approaches to perform probabilistic inference and learning in relational domains, active learning has only recently started to attract attention (Bilgic et al., 2010; Xu et al., 2010).

1.3 Contributions

The previous section outlined previous work on (mostly model-free) relational RL, which neglected explicit exploration, and the fundamental work by Walsh et al., which proved existence of KWIK learning in a model-based relational RL setting but falls short of practical applicability. The goal of this work is to propose a practically feasible online relational RL system that integrates efficient exploration in relational domains with fully unknown transition dynamics and learning complete action operators (including contexts, effects, and effect distributions).

Our approach extends Kearns and Singh's theoretically justified exploration technique E^3 and its successor R-MAX and outperforms existing non-relational techniques in a large number of relevant and challenging problems. More precisely, our contributions are the following:

- We introduce the problem of learning relational count functions which generalize the classical notion of state (action) visitation counts.

- We develop a general relational model-based reinforcement learning framework called REX (short for relational explorer), which lifts the concepts of E^3 and R-MAX to relational representations and uses learned relational count functions to estimate empirical model confidence in the relational RL setting.

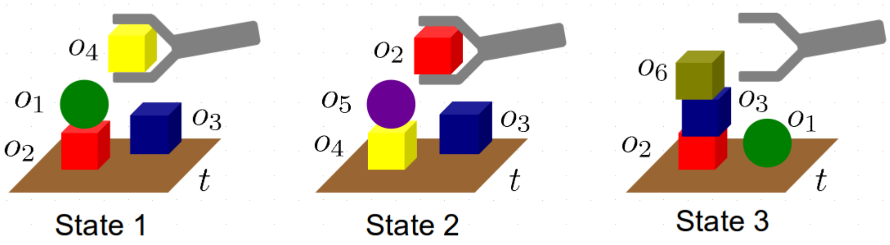
- We provide guarantees on the exploration efficiency of the general REX framework *under the assumption that we had* a relational KWIK learner and *were* capable of near-optimal planning in our domain.

- As a concrete instance of our REX framework, we integrate the state-of-the-art relational planner PRADA (Lang and Toussaint, 2010) and a learner for probabilistic relational rules (Pasula et al., 2007) into our framework. The resulting system is the first practically feasible efficient solution for relational domains with fully unknown transition dynamics.

Our extensive experimental evaluation in a 3D simulated complex robot manipulation environment with an articulated manipulator and realistic physics and in domains of the international planning competition (IPPC) shows that our methods can solve tasks in complex worlds where existing propositional methods fail. With these evaluations we also show that relational representations are a promising technique to formalize the idea of curriculum learning (Bengio et al., 2009). Our work has interesting parallels in cognitive science: Windridge and Kittler (2010) employ ideas of relational exploration for cognitive bootstrapping, that is, to progressively learn more abstract representations of an agent's environment on the basis of its action capabilities.

1.4 Outline

In the following section, we introduce previous work and background on which our methods build. In Section 3, we develop our general REX framework, including our model of relational count functions, and derive theoretical guarantees of this framework under the assumption that we had an ideal KWIK learner and near-exact inference. We then assume relational rule learning and PRADA as concrete learner and planner components for REX. In Section 4, we present the experimental evaluation of this overall system. Finally, we conclude in Section 5.



(a)

State	Enumerated	Factored	Relational
1	s_1	$on\_o_1\_o_2$, $on\_o_2\_t$, $on\_o_3\_t$, $inhand\_o_4$	$on(o_1,o_2)$, $on(o_2,t)$, $on(o_3,t)$, $inhand(o_4)$, $ball(o_1)$, $cube(o_2)$, $cube(o_3)$, $cube(o_4)$, $table(t)$
2	s_2	$on\_o_3\_t$, $on\_o_4\_t$, $on\_o_5\_o_4$, $inhand\_o_2$	$on(o_3,t)$, $on(o_4,t)$, $on(o_5,o_4)$, $inhand(o_2)$, $cube(o_2)$, $cube(o_3)$, $cube(o_4)$, $ball(o_5)$, $table(t)$
3	s_3	$on\_o_1\_t$, $on\_o_2\_t$, $on\_o_3\_o_2$, $on\_o_6\_o_3$	$on(o_1,t)$, $on(o_2,t)$, $on(o_3,o_2)$, $on(o_6,o_3)$, $cube(o_2)$, $cube(o_3)$, $cube(o_6)$, $ball(o_1)$, $table(t)$

(b)

Table 1: Illustration of three world representation types in a robot manipulation domain

2. Background on MDPs, Representations, Exploration and Transition Models

In this section, we set up the theoretical background for the relational exploration framework and algorithms presented later. First, we review briefly Markov decision processes (MDPs). Then, we describe different methods to represent states and actions in MDPs. Thereafter, we discuss exploration in MDPs including the algorithms E^3 and R-MAX. Finally, we discuss in detail compact relational transition models.

2.1 Markov Decision Processes

A Markov decision process (**MDP**) is a discrete-time stochastic control process used to model the interaction of an agent with its environment. At each time-step, the process is in one of a fixed set of discrete states S and the agent can choose an action from a set A. The transition model T specifies the conditional transition distribution $P(s' \,|\, s,a)$ over successor states s' when executing an action a in a given state s. The agent receives rewards in states according to a function $R : S \rightarrow \mathbb{R}_{\geq 0}$ (we assume non-negative rewards in this paper without loss of generality). The goal of planning in an MDP is to find a policy $\pi : S \rightarrow A$, specifying for each state the action to take, which maximizes the expected future rewards. For a discount factor $0 < \gamma < 1$, the value of a policy π for a state s is defined as the expected sum of discounted rewards $V^{\pi}(s) = E[\sum_t \gamma^t R(s_t) \,|\, s_0 = s, \pi]$. In our context, we face the problem of reinforcement learning (RL): we do not know the transition model T. Without loss of generality, we assume in this paper that the reward function R is given. We pursue a model-based approach and estimate T from our experiences. Based on our estimate $\hat{T}$ we compute (approximately) optimal policies.

2.2 State and Action Representations

The choice of representation for states S and actions A has important consequences for reinforcement learning techniques on the conceptual and the algorithmic level. Three different representation types dominate AI research on discrete representations: *(i)* unstructured enumerated representations, *(ii)* factored propositional representations, and *(iii)* relational representations. Table 1 presents three states in a robot manipulation domain together with their translations to the respective representations.

The simplest representation of states and actions is the **enumerated** (or flat) representation. States and actions are represented by single distinct symbols. Hence, the state and action spaces S and A are simply enumerated lists. In Table 1, the three states are represented by s_1, s_2 and s_3. This representation cannot capture the structure of states and does not provide a concept of objects. Therefore, it is impossible to express commonalities among states and actions. In the example, all three states appear equally different in this representation.

A **factored** propositional representation represents a state as a list of attributes. These attributes capture the state structure and hence commonalities among states. MDPs based on factored representations, called factored MDPs, have been investigated extensively in RL and planning research. The disadvantage of factored representations is their lack of a notion of objects. This makes it impossible to express commonalities among attributes. For instance, in Table 1 the attributes $on\_o_1\_o_2$ and $on\_o_5\_o_4$ are treated as completely different and therefore State 1 is perceived as equally different from State 2 as from State 3. Similar arguments hold for actions which are also represented by individual symbols.

Relational representations account for state structure and objects explicitly. The state space S is described by means of a relational vocabulary consisting of predicates $\mathcal{P}$ and functions $\mathcal{F}$, which yield the set of ground atoms with arguments taken from the set of domain objects $\mathcal{O}$. A state is defined by a list of true ground literals. The action space A is defined by atoms $\mathcal{A}$ with arguments from $\mathcal{O}$. In MDPs based on relational representations, called relational MDPs, the commonalities of state structures, actions and objects can be expressed. They enable compact representations since atoms containing logical variables allow for abstraction from concrete objects and situations. We will speak of grounding an abstract formula ψ if we apply a substitution σ that maps all of the variables appearing in ψ to objects in $\mathcal{O}$. In Table 1, abstract atoms capture the greater similarity of State 1 and State 2 in contrast to the one of State 1 and State 3: we can generalize State 1 and State 2 to the (semi-) abstract state $on(A,B), on(B,table), on(o_3,table), inhand(C)$, which is impossible for State 3.

The choice of representation determines the expressivity of models and functions in reinforcement learning. In particular, it influences the compactness and generalization of models and the efficiency of learning and exploration as we discuss next.

2.3 Exploration

A central challenge in reinforcement learning is the exploration-exploitation tradeoff. We need to ensure that we learn enough about the environment to accurately understand the domain and to be able to plan for high-value states (explore). At the same time, we have to ensure not to spend too much time in low-value parts of the state space (exploit). The discount factor γ of the MDP influences this tradeoff: if states are too far from the agent, the agent does not need to explore them as their potential rewards are strongly discounted; large values for γ necessitate more exploration.

The exploration efficiency of a RL algorithm can be measured in terms of its sample complexity: this is the number of time-steps it acts non-optimally, that is, without achieving near-optimal rewards. More formally, let $R_{max} > 0$ denote the maximal reward and m the number of unknown parameters of the MDP. m captures the complexity of the learning problem and corresponds to the number of parameters of the transition model T in our context. Let $V_t(s_t) = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,|\, s_0, a_0, r_0 \ldots, s_t]$ be the value function of the algorithm's policy (which is non-stationary and depends on its history), and V^* of the optimal policy. We define the sample complexity along the lines of Kakade (2003):

Definition 1 *Let $\varepsilon > 0$ be a prescribed accuracy and $\delta > 0$ be an allowed probability of failure. The expression $\eta(\varepsilon, \delta, m, \gamma, R_{max})$ is a **sample complexity** bound for the algorithm if independently of the choice of s_0, with probability at least $1 - \delta$, the number of timesteps such that $V_t(s_t) < V^*(s_t) - \varepsilon$ is at most $\eta(\varepsilon, \delta, m, \gamma, R_{max})$.*

An algorithm with a sample complexity polynomial in $1/\varepsilon$, $\log(1/\delta)$, m, $1/(1-\gamma)$ and R_{max} is called probably approximately correct in MDPs, **PAC-MDP** (Strehl et al., 2009).

The seminal approach R-MAX (Brafman and Tennenholtz, 2002) provides a PAC-MDP solution to the exploration-exploitation problem in unstructured enumerated state spaces: its sample complexity is polynomial in the number of states and actions (Strehl et al., 2009). R-MAX generalizes the fundamental approach E^3 (Explicit Explore or Exploit) (Kearns and Singh, 2002) for which a similar result has been established in a slightly different formulation: E^3 finds a near-optimal policy after a number of steps which is polynomial in the number of states and actions. Both E^3 and R-MAX focus on the concept of *known states* where all actions have been observed sufficiently often, defined in terms of a threshold ζ. For this purpose, they maintain state-action counts $\kappa(s,a)$ for all state-action pairs. E^3 (Algorithm 1) distinguishes explicitly between exploitation and exploration phases. If E^3 enters an *unknown* state, it takes the action it has tried the fewest times there (**direct exploration**). If it enters a *known* state, it tries to calculate a high-value policy within a model M_{exploit} including all known states with their sufficiently accurate model estimates and a special self-looping state $\tilde{s}$ with zero reward which absorbs unknown state-action pairs (assuming non-negative rewards in the MDP). If it finds a near-optimal policy in M_{exploit} this policy is executed (**exploitation**). Otherwise, E^3 plans in a different model M_{explore} which is equal to M_{exploit} except that all known states achieve zero reward and $\tilde{s}$ achieves maximal reward. This "optimism in the face of uncertainty" ensures that the agent explores unknown states efficiently (**planned exploration**). The value of the known-state threshold ζ depends on several factors: the discount factor γ, the maximum reward R_{max} and the complexity m of the MDP defined by the number of states and actions as well as the desired accuracy ε and confidence δ for the RL algorithm. The original formulation of E^3 assumes knowledge of the optimal value function V^* to decide for exploitation. Kearns and Singh discuss, however, how this decision can be made without that knowledge. In contrast to E^3, R-MAX decides implicitly for exploration or exploitation and maintains only one model $M_{\text{R-MAX}}$: it uses its model estimates for the known states, while unknown state-action transitions lead to the absorbing state $\tilde{s}$ with maximum reward R_{max}.

The theoretical guarantees of E^3 and R-MAX are strong. In practice, however, the number of exploratory actions becomes huge so that in case of the large state spaces of relational worlds, it is unrealistic to meet the theoretical thresholds of state visits. To address this drawback, variants of E^3 for factored but propositional MDP representations have been explored (Kearns and Koller, 1999; Guestrin et al., 2002). Our evaluations will include variants of factored exploration strategies where the factorization is based on grounded relational formulas. However, factored MDPs still do not enable generalization over objects. In this paper, we are investigating exploration strategies for relational representations and lift E^3 and R-MAX to relational domains. This may strongly improve

Algorithm 1 Sketch of E^3

```
Input: State $s$
Output: Action $a$
if $\forall a : \kappa(s,a) \geq \zeta$ then    ▷ State is known
    Plan in $M_{\text{exploit}}$ with zero-reward for $\tilde{s}$
    if resulting plan has value above some threshold then
        return first action of plan    ▷ Exploitation
    else
        Plan in $M_{\text{explore}}$ with maximum reward for $\tilde{s}$ and zero-reward for known states
        return first action of plan    ▷ Planned exploration
    end if
else    ▷ State is unknown
    return action $a = \text{argmin}_a \kappa(s,a)$    ▷ Direct exploration
end if
```

the exploration efficiency and the performance of a reinforcement learning agent. The key idea is to generalize the state-action counts $\kappa(s,a)$ of the original E^3 algorithm over states, actions and objects.

Sample complexity guarantees for more general MDPs can be developed within the **KWIK** (*knows what it knows*) framework (Li et al., 2011). In a model-based RL context, a KWIK learning algorithm can be used to estimate the unknown parts of the MDP. This learner accounts for its uncertainty explicitly: instead of being forced to make a prediction (for instance, of the probability of a successor state for a given state-action pair), it can instead signal its uncertainty about the prediction and return a unique symbol $\perp$. In this case, the (potentially noisy) outcome is provided to the learner which it can use for further learning to increase its certainty. For required model accuracy ε and confidence δ, a model class is called **KWIK-learnable** if there is a learner satisfying the following conditions: (1) if the learner does not predict $\perp$, its prediction is ε-accurate, and (2) the number of $\perp$-predictions is bounded by a polynomial function of the problem description (here, this is m in addition to ε and δ). The KWIK-R-MAX algorithm (Li, 2009) uses a KWIK learner $\mathcal{L}$ to learn the transition model T. The predictions of $\mathcal{L}$ define the known states in the sense of E^3 and R-MAX: a state is known if for all actions $\mathcal{L}$ makes ε-accurate predictions (with some failure probability) and unknown otherwise (where $\mathcal{L}$ predicts $\perp$). Where the learner is uncertain and predicts $\perp$, KWIK-R-MAX assumes a transition to $\tilde{s}$ with reward R_{max}. Li (2009) shows that if T can be efficiently KWIK-learned by $\mathcal{L}$, then KWIK-R-MAX using $\mathcal{L}$ is PAC-MDP. The overall accuracy ε for the sample complexity determines the required individual accuracies ε_T for the KWIK model learner and ε_P for the planner. Hence, one can derive an efficient RL exploration algorithm by developing an efficient KWIK learner $\mathcal{L}$ for the associated model learning problem.

2.4 Learning Generalizing Transition Models

The central learning task in model-based reinforcement learning is to estimate a transition model $\hat{T}$ from a set of experiences $\mathcal{E} = \{(s_t, a_t, s_{t+1})\}_{t=0}^{T-1}$ which can be used for decision-making and planning. (We assume the reward function is provided to the agent.) An example of $\mathcal{E}$ is given in Table 2. The learned model $\hat{T}$ defines a conditional distribution $P(s' \mid s,a)$. Generally, our view is that $\hat{T}$ does not have to be a precise map of the truth—the point of a model is to abstract and partition the space in such a way that this model is a good basis for accurate and efficient decision making.

$$
\begin{array}{lll}
\mathcal{E} = \{ & & \\
\quad grab(d): & & cube(a), cube(b), ball(c), ball(d), table(t), on(a,t), on(b,t), on(c,a), \underline{on(d,b)}\ldots \\
& \rightarrow & cube(a), cube(b), ball(c), ball(d), table(t), on(a,t), on(b,t), on(c,a), \underline{inhand(d)}\ldots \\
& & \\
\quad puton(t): & & cube(a), cube(b), ball(c), ball(d), table(t), on(a,t), on(b,t), on(c,a), \underline{inhand(d)}\ldots \\
& \rightarrow & cube(a), cube(b), ball(c), ball(d), table(t), on(a,t), on(b,t), on(c,a), \underline{on(d,t)}\ldots \\
& & \\
\quad grab(c): & & cube(a), cube(b), ball(c), ball(d), table(t), on(a,t), on(b,t), \underline{on(c,a)}, on(d,t)\ldots \\
& \rightarrow & cube(a), cube(b), ball(c), ball(d), table(t), on(a,t), on(b,t), on(d,t), \underline{inhand(c)}\ldots \\
\quad \vdots & & \vdots \\
\} & &
\end{array}
$$

Table 2: The reinforcement learning agent collects a series $\mathcal{E}$ of relational state transitions consisting of an action (on the left), a predecessor state (first line) and a successor state (second line after the arrow). The changing state features are underlined. The agent uses such experiences to learn a transition model resulting in a compression of the state transitions.

This can be achieved by compressing the experiences $\mathcal{E}$ in a compact model $\hat{T}$. Before describing a specific learning algorithm, we discuss conceptual points of learning generalizing models from experience. Compression of the experiences can exploit three opportunities:

- The *frame assumption* states that all state features which are not explicitly changed by an action persist over time. This simplifies the learning problem by deliberately ignoring large parts of the world.
- *Abstraction* allows to exploit the set of experiences efficiently by means of generalization. It can be achieved with relational representations.
- Assuming *uncertainty* in the observations is essential to fit a generalizing and regularized function. It allows to tradeoff the exact modeling of every observation (model likelihood) with generalization capabilities and to find low-complexity explanations of our experience: singleton events can be "explained away" as noise. In our point of view, the assumption of uncertainty is crucial to get relational representations working; it "unleashes" the model and opens the door for simplification, abstraction and compactification. Note that even in deterministic domains, it may be advantageous to learn a probabilistic model because it can be more abstract, more compact and neglect irrelevant details. In this sense, modeling uncertainty can also be understood as regularization.

The generalization capability of $\hat{T}$ and in turn the efficiency to learn it depend on the chosen representation. In an enumerated representation, we need to collect experiences for each relevant state-action pair (s,a) separately. In a factored propositional representation, one can—to some degree—generalize over states and actions by means of the structure imposed by the state attributes, but not over objects. For instance in Table 2, in a factored representation we need to learn the effects for $grab(d)$ and $grab(c)$ independently. In contrast, a *relational* representation enables compact transition models $P(s' \mid s,a)$ by using abstract formulas to generalize from concrete situations and object identities. From a statistical learning point of view, the purpose of such relational logic

$$
\begin{array}{ll}
grab(X): & on(X,Y),\ ball(X),\ cube(Y),\ table(Z) \\
& \rightarrow \quad \left\{ \begin{array}{lll} 0.7 & : & inhand(X),\ \neg on(X,Y) \\ 0.2 & : & on(X,Z),\ \neg on(X,Y) \\ 0.1 & : & \text{noise} \end{array} \right.
\end{array}
$$

Table 3: Example NID rule for a robot manipulation scenario, which models to try to grab a ball X. The cube Y is implicitly defined as the one below X (deictic referencing). X ends up in the robot's hand with high probability, but might also fall on the table. With a small probability something unpredictable happens. Refer to Figure 1 for an example application.

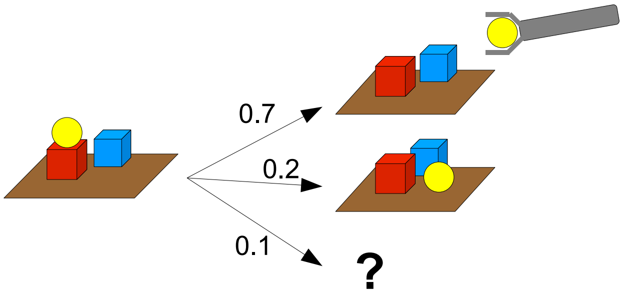


Figure 1: The NID rule defined in Table 3 can be used to predict the effects of action $grab(specific\_ball)$ in the situation on the left side. The right side shows the possible successor states as predicted by the rule. The noise outcome is indicated by a question mark and does not define a unique successor state.

descriptions is simply to provide complex feature descriptors. For instance in Table 2, both $grab(d)$ and $grab(c)$ are used to learn one general model for the abstract action $grab(X)$. In turn, the learned model also models situations with previously unseen objects (which is impossible in enumerated and factored propositional representations). In this sense, we view a relational transition model $\hat{T}$ as a (noisy) compressor of the experiences $\mathcal{E}$ whose compactness enables generalization.

Transition models which generalize over objects and states play a crucial role in our relational exploration algorithms. While our ideas work with any type of relational model that can be learned from experience, for illustration and to empirically evaluate our ideas we employ noisy indeterministic deictic (NID) rules which we present next.

2.4.1 NOISY INDETERMINISTIC DEICTIC RULES

An example of noisy indeterministic deictic (**NID**) rules (Pasula et al., 2007) is shown in Table 3 for our robot manipulation domain. Figure 1 depicts a situation where this rule can be used for prediction. Formally, a NID rule r is given as

$$a_r(\mathcal{X}):\ \phi_r(\mathcal{X}) \quad \rightarrow \quad \begin{cases} p_{r,1} & : \quad \Omega_{r,1}(\mathcal{X}) \\ & \vdots \\ p_{r,m_r} & : \quad \Omega_{r,m_r}(\mathcal{X}) \\ p_{r,0} & : \quad \Omega_{r,0} \end{cases} ,$$

where $\mathcal{X}$ is a set of logic variables in the rule (which represents a (sub-)set of abstract objects). The rule r consists of preconditions, namely that action a_r is applied on $\mathcal{X}$ and that the abstract state context ϕ_r is fulfilled, and m_r+1 different abstract outcomes with associated probabilities $p_{r,i} > 0$, $\sum_{i=0} p_{r,i} = 1$. Each outcome $\Omega_{r,i}(\mathcal{X})$ describes which literals are predicted to change when the rule is applied. The context $\phi_r(\mathcal{X})$ and outcomes $\Omega_{r,i}(\mathcal{X})$ are conjunctions of literals constructed from the predicates in $\mathcal{P}$ as well as equality statements comparing functions from $\mathcal{F}$ to constant values. The so-called *noise outcome* $\Omega_{r,0}$ subsumes all possible action outcomes which are not explicitly specified by one of the other $\Omega_{r,i}$. This includes rare and overly complex outcomes which are typical for noisy domains and which we do not want to cover explicitly for compactness and generalization reasons. For instance, in the context of the rule depicted in Figure 1 a potential, but highly improbable outcome is to grab the blue cube while pushing all other objects off the table: the noise outcome accounts for this without the burden of explicitly stating it. The arguments of the action $a(\mathcal{X}_a)$ may be a proper subset $\mathcal{X}_a \subset \mathcal{X}$ of the variables $\mathcal{X}$ of the rule. The remaining variables are called deictic references $DR = \mathcal{X} \setminus \mathcal{X}_a$ and denote objects relative to the agent or action being performed.

So, how do we apply NID rules? Let σ denote a substitution that maps variables to constant objects, $\sigma : \mathcal{X} \rightarrow O$. Applying σ to an abstract rule $r(\mathcal{X})$ yields a *grounded rule* $r(\sigma(\mathcal{X}))$. We say a grounded rule r *covers* a state s and a ground action a if $s \models \phi_r$ and $a = a_r$. Let Γ be our set of ground rules and $\Gamma(s,a) \subset \Gamma$ the set of rules covering (s,a). If there is a unique covering rule $r_{s,a} \in \Gamma(s,a)$ with $|\Gamma(s,a)| = 1$, we use it to model the effects of action a in state s. We calculate $P(s' \,|\, s,a)$ by taking all outcomes of $r_{s,a}$ (omitting subscripts in the following) into account weighted by their respective probabilities,

$$P(s' \,|\, s,a) \;=\; P(s' \,|\, s,r) \;=\; \sum_{i=1}^{m_r} p_{r,i} P(s'|\Omega_{r,i},s) + p_{r,0} P(s' \,|\, \Omega_{r,0},s),$$

where, for $i > 0$, $P(s' \,|\, \Omega_{r,i},s) = I(s \wedge \Omega_{r,i} \models s')$ is a deterministic distribution that is one for the unique state constructed from s taking the changes of $\Omega_{r,i}$ into account. (The function $I(\cdot)$ maps logical truth values to 0 or 1.) The distribution given the noise outcome, $P(s' \,|\, \Omega_{r,0},s)$, is unknown and needs to be estimated. Pasula et al. use a worst-case constant bound $p_{min} \leq P(s'|\Omega_{r,0},s)$ to lower bound $P(s'|s,a)$. If a state-action pair (s,a) does *not* have a unique covering rule r (including the case that more than one rule covers the state-action pair, resulting in potentially conflicting predictions), we use a noisy default rule r_ν which predicts all effects as noise: $P(s'|s,r_\nu) = P(s' \,|\, \Omega_{r_\nu,0},s)$.

The ability to learn models of the environment from experience is a crucial requirement for autonomous agents. The problem of learning rule-sets is in general NP-hard, but with suitable assumptions and restrictions efficiency guarantees on the sample complexity can be given for many learning subtasks (Walsh, 2010). Pasula et al. (2007) have proposed a supervised batch learning algorithm for *complete* NID rules based on ideas developed in inductive logic programming (Nienhuys-Cheng and de Wolf, 1997). This algorithm learns the structure of rules (contexts and outcomes) as well as their parameters from experience triples $\mathcal{E} = \{(s_t, a_t, s_{t+1})_{t=0}^{T-1}\}$, relying on the frame assumption. Efficient exploration strategies to collect useful data $\mathcal{E}$ for learning, however, were not investigated by Pasula et al.—this will be achieved with our proposed methods. The learning algorithm performs a greedy search through the space of rule-sets, maintaining the thus far best performing rule-set. It optimizes the tradeoff between maximizing the likelihood of the experience triples and minimizing the complexity of the current hypothesis rule-set Γ by optimizing the scoring metric (Pasula et al., 2007)

$$S(\Gamma) = \sum_{(s,a,s')} \log P(s' \,|\, s, r_{s,a}) - \alpha \sum_{r \in \Gamma} \mathrm{PEN}(r) \,, \tag{1}$$

where $r_{s,a}$ is either the unique covering rule for (s,a) or the noisy default rule r_ν. α is a scaling parameter that controls the influence of regularization. $\mathrm{PEN}(r)$ penalizes the complexity of a rule and is defined as the total number of literals in r. The larger α is set, the more compact are the learned rule-sets—and thus, the more general, but potentially also more uncertain and inaccurate. For instance, if we set $\alpha = \infty$, the resulting model will consist of only the default rule, explaining all state transitions as noise. In contrast, if we set $\alpha = 0$, the resulting model explains each experience as accurately as possible, potentially overfitting and not generalizing the data. The learning algorithm is initialized with a rule-set comprising only the noisy default rule r_ν and then iteratively adds new rules or modifies existing ones using a set of search operators. More precisely, these search operators take the current rule-set and the set of experiences as input, repeatedly select individual rules, modify them with respect to the experiences, and thereby produce new rule-sets. If the best new rule-set scores better than the current rule-set according to Equation (1), it becomes the new rule-set maintained by the algorithm. For instance, the most complex search operator, called *ExplainExamples*, creates new rules from experiences which are modeled by the default rule thus far. First, it builds a complex specific rule for an experience; then, it tries to trim this rule to generalize it to other experiences. Other search operators work directly on existing rules: they add new literals to rule contexts or delete them from contexts, add new deictic references to rules in form of literal sets or delete them, delete complete rules from rule-sets, or generalize comparison literals involving function values in rule contexts.

The noise outcome of NID rules is crucial for learning: it permits the iterative formulation of the learning algorithm as the corresponding noisy default rule covers all experiences without unique covering rule so far; and more importantly, it avoids overfitting by refusing to model rare and overly complex experiences. This advantage has been demonstrated empirically by Pasula et al. (2007). Its drawback is that the successor state distribution $P(s' \,|\, \Omega_{r,0}, s)$ is unknown. To deal with this problem, the learning algorithm uses a lower bound p_{min} to approximate this distribution, as described above. In our experience, while the resulting rule-sets are robust to minor changes in the two parameters p_{min} and α, the concrete choice for their values is important for learning expressive rule-sets, and their mutual influence needs to be taken into account. These parameters, however, cannot be optimized by the rule-learning algorithm itself: the required degree of rule compactness is not only determined by the complexity of the modeled domain (in terms of the vocabulary size, stochasticity and context sensitivity of actions), but also by the purpose the rules have to serve (for example, the required accuracy for planning). Nonetheless, in our experiments no significant efforts were necessary to find appropriate values for p_{min} and α: preliminary testings of a small number of typical value-combinations were sufficient to set the parameters effectively for each domain. All other choices in the learning algorithm (in particular, the choice of search operators and their order) were the same across all reported experiments.

The rule learning algorithm of Pasula et al. uses greedy heuristics in its attempt to learn complete rules. Hence, one cannot give guarantees on its efficiency, correctness or convergence. It has been shown empirically, however, that learned NID rules provide accurate transition models in noisy robot manipulation domains (Pasula et al., 2007; Lang and Toussaint, 2010). In this paper, we show further that the transition dynamics of many domains of the international planning competition can be learned reliably with NID rules. In all our investigated domains, independent learning runs converged to the same or very similar rule-sets; in particular, the learned rule contexts are usually the same.

2.5 Planning with Probabilistic Relational Rules

Model-based RL requires efficient planning for exploitation and planned (directed) exploration. The semantics of NID rules allow one to find a "satisficing" action sequence in relational domains that will lead with high probability to states with large rewards. In this paper, we use the **PRADA** algorithm (Lang and Toussaint, 2010) for planning in grounded relational domains. Empirical results have shown that PRADA finds effective and reliable plans in difficult scenarios. PRADA grounds a given set of abstract NID rules with respect to the objects in the domain and converts the grounded rules to factored dynamic Bayesian networks (DBNs). The random variables (nodes) of the DBNs represent the state literals, actions, rules, rule contexts and outcomes and rewards at different time-steps; factors on these variables define the stochastic transition dynamics according to the NID rules. For planning, PRADA samples sequences of actions in an informed way taking into account the effects of previous actions. To evaluate an action sequence, it uses fast approximate inference to calculate posterior beliefs over states and rewards. If exact instead of approximate inference is used, PRADA is guaranteed to find the optimal plan with high probability given a sufficient number of samples (Lang, 2011). PRADA can plan for different types of rewards, including conjunctions of abstract and grounded literals. Thus, it can be used in model-based RL for both exploiting the learned model to plan for high-reward states as well as for exploring unknown states and actions using the R_{max} reward. The number of sampled action sequences trades off computation time and plan quality: the more samples are taken, the higher are the chances to find a good plan. In our experiments, we set this number sufficiently high to ensure that good plans are found with high probability. If we know the maximum reward, we can determine PRADA's planning horizon for a desired level of accuracy from the discount factor γ of the MDP: we can calculate after which time-step the remaining maximally possible rewards can be ignored.

3. Exploration in Relational Domains

Relational representations enable generalization of experiences over states, actions and objects. Our contribution in this paper are strategies to exploit this for exploration in model-based reinforcement learning. First, we discuss the implications of a relational knowledge representation for exploration on a conceptual level (Section 3.1). We show how to quantify the knowledge of states and actions by means of a generalized, relational notion of state-action counts, namely a *relational count function*. This opens the door to a large variety of possible exploration strategies. Thereafter, we propose a relational model-based reinforcement learning framework lifting the ideas of the algorithms E^3 and R-MAX to symbolic representations (Section 3.2). Then, we discuss theoretical guarantees concerning the exploration efficiency in our framework (Section 3.3). Finally, we present a concrete algorithm in this framework which uses some of the previously introduced exploration strategies (Section 3.4), including an illustrative example (Section 3.5).

3.1 Relational Count Functions for Known States and Actions

The theoretical derivations of the efficient non-relational exploration algorithms E^3 and R-MAX show that the concept of known states is crucial. On the one hand, the confidence in estimates in known states drives exploitation. On the other hand, exploration is guided by seeking for novel (yet unknown) states and actions. For instance, the direct exploration phase in E^3 chooses novel actions, which have been tried the fewest; the planned exploration phase seeks to visit novel states, which are labeled as yet unknown.

In the original E^3 and R-MAX algorithms operating in an enumerated state space, states and actions are considered known based directly on their **counts**: the number of times they have been visited. In relational domains, we should go beyond simply counting state-action visits to estimate the novelty of states and actions:

- The size of the state space is exponential in the number of objects. If we base our notion of known states directly on visitation counts, then the overwhelming majority of all states will be labeled yet-unknown and the exploration time required to meet the criteria for known states of E^3 and R-MAX even for a small relevant fraction of the state space becomes exponential in the number of objects.
- The key benefit of relational learning is the ability to generalize over yet unobserved instances of the world based on relational abstractions. This implies a fundamentally different perspective on what is novel and what is known and permits qualitatively different exploration strategies compared to the propositional view.

We propose to generalize the notion of counts to a *relational count function* that quantifies the degree to which states and actions are known. Similar to using mixtures (e.g., of Gaussians) for density modeling, we model a count function over the state space as a linear superposition of features f_k ("mixture components") such that visitation of a single state generalizes to "neighboring" states—where "neighboring" is defined by the structure of the features f_k. The only technical difference between count functions and mixture density models is that our count functions are not normalized.

Theoretical guarantees on the convergence and accuracy for this model are discussed in Section 3.3. Given sets of features f_k and mixture weights w_k, a count function over states s can be written as

$$\kappa(s) = \sum_k w_k f_k(s) \ .$$

The state features f_k can be arbitrary. An example for a relational feature are binary tests that have value 1 if some relational query is true for s; otherwise they have value 0. Estimating such mixture models is a type of unsupervised learning or clustering and involves two problems: finding the features f_k themselves (structure learning) and estimating the feature weights w_k (parameter learning).

While the use of relational features offers a great compactness and generalization it complicates the feature selection (structure learning) problem: there are no prior restrictions on the length and complexity of features and hence, we have essentially infinitely many features to choose from. The longer a relational feature is (e.g., a long conjunction of abstract literals) and the more variables it contains, the larger is the number of possible ways to bind the variables and the larger is the set of refined features that we potentially have to consider when evaluating a feature. In addition, the space of the instances s is very large in our case. Recall that even very small relational models can have hundreds of ground atoms and it would be impossible to represent all possible states. For instance, for just a single binary relation and 10 objects there would be 100 ground atoms and hence 2^{100} distinct states, which is clearly intractable. Thus, we need to focus on a compact, structured representation of the count functions. Furthermore, we only have a finite set of positive (that is, experienced) states from which we have to generalize to other states, while taking care not to overgeneralize wrongly. Note that in contrast the original E^3 and R-MAX algorithms for unstructured domains do not face this problem as they do not generalize experiences to other states. Thus, we essentially face the problem of structure learning from positive examples only (Muggleton, 1997). This is more difficult than the traditional setting considered in relational learning that additionally assumes that negative (impossible) examples are given.

If the features are given, however, the estimation of the count function becomes simpler: only the weights need to be learned. For instance, in 1-class SVMs for density estimation assumptions about the feature structure are embedded in the kernel function; in the mixture of Gaussians model, the functional form of Gaussians is given a-priori and provides the structure. We propose a "patch up" approach in this paper. We examine different choices of features in the relational setting whose weights can be estimated based on empirical counts. While they are only approximations, we then show that we can "patch up" and improve some of these approximations, namely the context-based features (see below) through learning NID rules. In our relational model-based RL algorithms as well as in our evaluation, we focus on context-based features. In other words, our methods implicitly solve both structure and parameter learning from positive examples only.

Let us now introduce different choices of features for the estimation of relational count functions. These imply different approaches to quantify known states and actions in a relational RL setting. We focus on a specific type of features, namely queries $q \in Q$. Queries are simply relational formulas, for instance conjunctions of ground or abstract literals, which evaluate to 0 or 1 for a given state s. We discuss different choices of Q in detail below including examples. Our count functions use queries q in combination with the set of experiences $\mathcal{E}$ of the agent and the estimated transition model $\hat{T}$. Given a set Q of queries we model the state count function as

$$\kappa_Q(s) \;=\; \textstyle\sum_{q \in Q} c_{\mathcal{E}}(q)\, I(\exists \sigma : s \models \sigma(q)) \tag{2}$$

$$\text{with} \quad c_{\mathcal{E}}(q) \;=\; \textstyle\sum_{(s_e, a_e, s'_e) \in \mathcal{E}} I(\exists \sigma : s_e \models \sigma(q)) \;. \tag{3}$$

This function combines the confidences of all queries $q \in Q$ which are fulfilled in state s. The second term in Equation (2) examines whether query q is fulfilled in s: the substitution σ is used to ground potentially abstract queries; the function $I(\cdot)$ maps logical statements to 1 if they are satisfied and to 0 otherwise. $c_{\mathcal{E}}(q)$, the first term in Equation (2), is an experience-count of query q: it quantifies the number of times query q held in previously experienced predecessor states in the agent's set of observed state transitions $\mathcal{E} = \{(s_t, a_t, s_{t+1})\}_{t=1}^{T-1}$. Overall, a state s has a high count $\kappa_Q(s)$ if it satisfies queries $q \in Q$ with large experience-counts $c_{\mathcal{E}}(q)$. This implies that all states with low $\kappa_Q(s)$ are considered novel and should be explored, as in E^3 and R-MAX. The model for state-action count functions is analogous. In the following, we discuss different choices for queries q and the accompanying count estimates which emphasize different aspects of relational data. We use the three states shown in Table 1 as our running example: we assume that our experiences consist of exactly State 1, that is $\mathcal{E} = \{(s_1, a_1, s'_1)\}$ (the action a_1 and the successor state s'_1 are ignored by $\kappa_Q(s)$), while State 2 and State 3 have not been experienced.

Enumerated: Let us first consider briefly the propositional enumerated setting. We have a finite enumerated state space S and action space A. The set of queries,

$$Q_{enum} = \{s \,|\, \exists (s_e, a_e, s'_e) \in \mathcal{E} : s_e = s\},$$

corresponds to predecessor states $s \in S$ which have been visited in $\mathcal{E}$. Thus, queries are conjunctions of positive and negated ground atoms which fully describe a state. This translates directly to the count function

$$\kappa_{enum}(s) = c_{\mathcal{E}}(s)\ , \quad \text{with } c_{\mathcal{E}}(s) = \textstyle\sum_{(s_e,a_e,s'_e)\in\mathcal{E}} I(s_e = s)\ .$$

The experience-count $c_{\mathcal{E}}(q)$ in the previous equation counts the number of occasions state s has been visited in $\mathcal{E}$ (in the spirit of Thrun, 1992). There is no generalization in this notion of known states. Similar arguments can be applied on the level of state-action counts $\kappa(s,a)$. As an illustration, in our running example of Table 1 both State 2 and State 3 are equally unknown and novel: both are not the experienced State 1.

Literal-based: Given a relational structure with the set of logical predicates $\mathcal{P}$, an alternative approach to describe what are known states is based on counting how often a (ground) literal (a potentially negated atom) has been observed true or false in the experiences $\mathcal{E}$ (all statements equally apply to functions $\mathcal{F}$, but we neglect this case here). A literal l for a predicate $P \in \mathcal{P}$ containing variables is abstract in that it represents the set of all corresponding ground, that is variable-free, literals for P. Ground literals then play the role of the traditional factors used in mixture models. First, we consider ground literals $l \in \mathcal{L}^G$ with arguments taken from the domain objects O. This leads to the set of queries $Q_{lit} = \mathcal{L}^G$ and in turn to the count function

$$\begin{aligned} \kappa_{lit}(s) \;&=\; \textstyle\sum_{l\in\mathcal{L}^G}\ c_{\mathcal{E}}(l)\, I(s \models l) \\ \text{with} \quad & c_{\mathcal{E}}(l) := \textstyle\sum_{(s_e,a_e,s'_e)\in\mathcal{E}} I(s_e \models l). \end{aligned}$$

Each query $l \in \mathcal{L}^G$ examines whether l has the same truth values in s as in experienced states. This implies that a state is considered familiar (with $\kappa_{lit}(s) > 0$) if a ground literal that is true (false) in this state has been observed true (false) before. Thus abstraction over states can be achieved by means of ground literals. We can follow the same approach for abstract literals $\mathcal{L}^A$ and set $Q_{lit} = \mathcal{L}^A$. For $l \in \mathcal{L}^A$ and a state s, we examine whether there are groundings of the logical variables in l such that s covers l. More formally, we replace $s \models l$ by $\exists\sigma : s \models \sigma(l)$. For instance, we may count how often a specific blue ball was on top of *some* other object. If this was rarely the case this implies a notion of novelty which guides exploration. For example, in Table 1 State 1 and State 3 share the ground literal $on(o_2,t)$ while this literal does not hold in State 2. Thus, if $Q_{lit} = \{on(o_2,t)\}$ and State 1 is the sole experienced state, then State 3 is perceived as better known than State 2. In contrast, if we use the abstract query $inhand(X)$ (expressing there was *something* held inhand) and set $Q_{lit} = \{inhand(X)\}$, then State 3 is perceived as more novel, since in both State 1 and State 2 some object was held inhand. Note that this second query abstracts from the identities of the inhand held objects.

Context-based: Assume that we are given a finite set Φ of contexts, which are queries consisting of formulas over predicates and functions. While many relational knowledge representations have some notion of context or rule precondition, in our running example of NID rules these may correspond to the set of NID rule contexts $\{\phi_r\}$. These are learned from the experiences $\mathcal{E}$ and have specifically been optimized to be a compact context representation (cf. Section 2). Given a set Φ of such queries, setting $Q_\Phi = \Phi$ results in the count function

$$\kappa_\Phi(s) \;=\; \textstyle\sum_{\phi\in\Phi} c_{\mathcal{E}}(\phi)\, I(\exists\sigma : s \models \sigma(\phi))$$
$$\text{with} \quad c_{\mathcal{E}}(\phi) \;=\; \textstyle\sum_{(s_e,a_e,s'_e)\in\mathcal{E}} I(\exists\sigma : s_e \models \sigma(\phi)).$$

$c_{\mathcal{E}}(\phi)$ counts in how many experiences $\mathcal{E}$ the context respectively query ϕ was covered with arbitrary groundings. Intuitively, contexts may be understood as describing situation classes based on whether the same abstract prediction models can be applied. Taking this approach, states are considered novel if they are not covered by any existing context ($\kappa_\Phi(s) = 0$) or covered by a context that has rarely occurred in $\mathcal{E}$ ($\kappa_\Phi(s)$ is low). That is, the description of novelty which drives exploration is lifted to the level of abstraction of these relational contexts. Similarly, we estimate a count function for known state-action pairs based on state-action contexts. For instance, in the case of a set Γ of NID rules, each rule defines a state-action context, resulting in the count

$$\kappa_\Phi(s,a) \;=\; \textstyle\sum_{r\in\Gamma} c_{\mathcal{E}}(r)\, I(r = r_{s,a}), \quad \text{with} \quad c_{\mathcal{E}}(r) \;:=\; |\mathcal{E}(r)|,$$

which is based on counting how many experiences are covered by the unique covering rule $r_{s,a}$ for a in s. $\mathcal{E}(r)$ are the experiences in $\mathcal{E}$ covered by r, $\mathcal{E}(r) = \{(s,a,s') \in \mathcal{E} \,|\, r = r_{s,a}\}$. Thus, the number of experiences covered by the rule $r_{s,a}$ modeling (s,a) can be understood as a measure of confidence in $r_{s,a}$ and determines $\kappa_\Phi(s,a)$. We will use $\kappa_\Phi(s,a)$ in our proposed algorithm below. In the example of Table 1, assume in State 1 we perform $puton(o_3)$ successfully, that is, we put the inhand cube o_4 on o_3. From this experience, we learn a rule for the action $puton(X)$ with the context $\phi = clear(X) \wedge inhand(Y)$. Thereafter, State 3 is perceived as more novel than State 2: the effects of $puton(o_3)$ can be predicted in State 2, but not in State 3.

Similarity-based: Different methods to estimate the similarity of relational states exist. For instance, Driessens et al. (2006) and Halbritter and Geibel (2007) present relational reinforcement learning approaches which use relational graph kernels to estimate the similarity of relational states. These can be used to estimate relational counts which, when applied in our context, would readily lead to alternative notions of novelty and thereby exploration strategies. This count estimation technique bears similarities to Metric E^3 (Kakade et al., 2003). Applying such a method to model $\kappa(s)$ from $\mathcal{E}$ implies that states are considered novel (with low $\kappa(s)$) if they have a low kernel value (high "distance") to previous explored states. Let $k(\cdot,\cdot) \in [0,1]$ denote an appropriate kernel for the queries $q \in Q$, for instance based on relational graph kernels. We replace the hard indicator function $I(\exists\sigma : s \models \sigma(q))$ in Equation (2) by the kernel function, resulting in the more general kernel-based count function

$$\kappa_{k,Q}(s) \;=\; \textstyle\sum_{q\in Q} c_{\mathcal{E}}(q)\, k(s,q)\;.$$

If one sets $Q = \{s \,|\, s \in S\}$ to the set of all states, the previous count function measures the distance to all observed predecessor states multiplied by experience-counts. In the example of Table 1 with $\mathcal{E} = \{(s_1,a_1,s'_1)\}$, if we use relational graph kernels, the isomorphism of the graph representations of State 1 and State 2 leads to a large kernel estimate $k(s_1,s_2)$, while the different graph structure of State 3 causes $k(s_1,s_3)$ to be small, therefore $\kappa_{k,Q}(s_2) > \kappa_{k,Q}(s_3)$.

The above approaches are based on counts over the set of experiences $\mathcal{E}$. As a simple extension, we propose using the **variability** within $\mathcal{E}$. Consider two different series of experiences $\mathcal{E}_1$ and $\mathcal{E}_2$ both of size n. Imagine that $\mathcal{E}_1$ consists of n times the same experience, while the experiences in $\mathcal{E}_2$ differ. For instance, $\mathcal{E}_1$ might list repeatedly grabbing the same object in the same context, while $\mathcal{E}_2$ might list grabbing different objects in varying conditions. $\mathcal{E}_2$ has a higher variability. Although the experience-counts $c_{\mathcal{E}_1}(q)$ and $c_{\mathcal{E}_2}(q)$ as defined in Equation (3) are the same, one might be tempted to say that $\mathcal{E}_2$ confirms q better as the query q succeeded in more heterogeneous situations, supporting the claim of generalization. We formalize this by redefining the experience-counts in Equation (3) using a distance estimate $d(s,s') \in [0,1]$ for two states s and s' as

$$c^V_{\mathcal{E}}(q) = \sum_{(s_t,a_t,s'_t)\in\mathcal{E}} I(\exists\sigma : s_t \models \sigma(q)) \cdot \delta(s_t, \mathcal{E}^{(q,t)}) \,,$$

$$\text{with} \quad \mathcal{E}^{(q,t)} = \{(s_e,a_e,s'_e) \in \mathcal{E} \,|\, \exists\sigma : s_e \models \sigma(q) \wedge e < t\}$$

$$\text{and} \quad \delta(s,\mathcal{E}) = \min_{(s_e,a_e,s'_e)\in\mathcal{E}} d(s,s_e) \,.$$

The experience-count $c^V_{\mathcal{E}}(q)$ weights each experience based on its distance $\delta(s_t,\mathcal{E}^{(q,t)})$ to prior experiences (while the original $c_{\mathcal{E}}(q)$ assigns all experiences the same weight irrespective of other experiences). Here, the measure $d(s,s')$ computes the distance between two ground states, but calculations on partial or (partially) lifted states are likewise conceivable. To compute $d(s,s')$, a kernel $k(s,s') \in [0,1]$ as above might be employed, $d(s,s') \propto 1-k(s,s')$, for instance using a simple distance estimate based on least general unifiers (Ramon, 2002). For illustration, consider again the states in Table 1. Assume two series of experiences $\mathcal{E}_1 = \{(s_1,a_1,s'_1),(s_2,a_2,s'_2)\}$ and $\mathcal{E}_2 = \{(s_1,a_1,s'_1),(s_3,a_3,s'_3)\}$ and the query $q = on(X,Y) \wedge ball(X)$ (that there is a ball on-top of some other object). All State 1, State 2 and State 3 cover this query, therefore the standard counts for $\mathcal{E}_1$ and $\mathcal{E}_2$ according to Equation (3) are the same, $c_{\mathcal{E}_1}(q) = c_{\mathcal{E}_2}(q)$. As State 1 and State 2 share the same structure, however, the variability within $\mathcal{E}_1$ is smaller than within $\mathcal{E}_2$. Thus, $\mathcal{E}_2$ provides more heterogeneous evidence for q and therefore $c^V_{\mathcal{E}_2}(q) > c^V_{\mathcal{E}_1}(q)$.

3.2 Relational Exploration Framework

The approaches to estimate relational state and action counts which have been discussed above open a large variety of possibilities for concrete exploration strategies. In the following, we derive a relational model-based reinforcement learning framework we call REX (short for relational explorer) in which these strategies can be applied. This framework is presented in Algorithm 2. REX lifts E^3 and R-MAX to relational exploration.

At each time-step, the relational explorer performs the following general steps:

1. It adapts its estimated relational transition model $\hat{T}$ with the set of experiences $\mathcal{E}$.

2. Based on $\mathcal{E}$ and $\hat{T}$, it estimates the count function for known states and actions, $\kappa(s)$ and $\kappa(s,a)$. For instance, $\hat{T}$ might be used to provide formulas and contexts to estimate a specific relational count function.

3. The estimated count function is used to decide for the next action a_t based on the strategy of either E^3 or R-MAX.

   In the case of relational E^3, the phase-ordering is exactly the same as in the original E^3: if the current state s_t is known, exploitation is tried and if exploitation fails (the planner does not find

Algorithm 2 Relational Exploration (REX)

```
Input: Start state $s_0$, reward function $R$, confidence threshold $\zeta$
Set of experiences $\mathcal{E} = \emptyset$
for t=0,1,2 ... do
    Update transition model $\hat{T}$ according to $\mathcal{E}$
    Estimate $\kappa(s)$ and $\kappa(s,a)$ from $\mathcal{E}$ and $\hat{T}$    ▷ Relational representation enables generalization
    if $E^3$-exploration then    ▷ $E^3$-exploration
        if $\forall a \in \mathcal{A} : \kappa(s_t, a) \geq \zeta$ then    ▷ State is known → uses relational generalization
            Plan in $M_{\text{exploit}}$ with zero reward for $\tilde{s}$    ▷ uses relational generalization
            if resulting plan has value above some threshold then
                $a_t$ = first action of plan    ▷ Exploitation
            else
                Plan in $M_{\text{explore}}$ with maximum reward for $\tilde{s}$ and zero-reward for known states
                            ▷ uses relational generalization
                $a_t$ = first action of plan    ▷ Planned exploration
            end if
        else    ▷ State is unknown
            $a_t = \text{argmin}_a \kappa(s_t, a)$    ▷ Direct exploration → uses relational generalization
        end if
    else    ▷ R-MAX-exploration
        Plan in $M_{\text{R-MAX}}$ with maximum reward for $\tilde{s}$ and given reward $R$ for all known states
                    ▷ uses relational generalization
        $a_t$ = first action of plan
    end if
    Execute $a_t$
    Observe new state $s_{t+1}$
    Update set of experiences $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_t, a_t, s_{t+1})\}$
end for
```

a policy with a sufficiently high value), planned exploration to unknown states is undertaken; otherwise (if the current state s_t is not known), direct exploration is performed. Like in the original non-relational E^3, the decision for exploitation in relational E^3 assumes knowledge of the optimal values. To remove this assumption, the relational explorer can instead attempt planned exploration first along the lines described by Kearns and Singh for the original E^3.

In the case of relational R-MAX, a single model is built in which all unknown states according to the estimated counts lead to the absorbing special state $\tilde{s}$ with reward R_{max}.

4. Finally, the action a_t is executed, the resulting state s_{t+1} observed and added to the experiences $\mathcal{E}$, and the process repeated.

The estimation of relational count functions for known states and actions, $\kappa(s)$ and $\kappa(s,a)$, and the resulting generalization over states, actions and objects play a crucial role at several places in the algorithm; for instance, in the case of relational E^3,

- to decide whether the current state is considered known or novel;
- to determine the set of known states where to try to exploit;
- to determine the set of both novel and known states in planned exploration to decide for target states and plan-through states;

- and to determine which action to execute in direct exploration where the least known action with the lowest $\kappa(s,a)$ is chosen; this combines the original E^3 (choosing the action with the fewest "visits") with relational generalization (defining "visits" by means of state abstraction).

The parameter ζ in our relational exploration framework REX defines the threshold to decide whether states and actions are known or unknown. For the original E^3 and R-MAX algorithms for enumerated state and action spaces, a value for ζ can be derived such that efficient exploration in polynomial time can be guaranteed. This derivation is closely tied to the enumerated representation of state and action spaces, not straightforward to apply in practice and will lead to overly large thresholds (see Section 2.3). In the next subsection, we discuss how to develop theoretical guarantees on the sample complexity for relational domains within our REX framework.

3.3 Theoretical Guarantees on the Sample Complexity

The original R-MAX algorithm operating in enumerated state and action spaces is PAC-MDP: its sample complexity is polynomial in the number of states and actions (Strehl et al., 2009). A similar result for E^3 has been established in a slightly different formulation: E^3 finds a near-optimal policy after a number of steps which is polynomial in the number of states and actions. In the following, we briefly establish conditions for which similar guarantees can be made in our relational exploration framework REX using count functions: we state simple conditions for a KWIK learner to "match" with the used count function such that REX using this learner is PAC-MDP. This clarifies also a relation to Walsh's assumptions about mutually exclusive contexts in his KWIK learning setup.

Let $\mathcal{E}_{\kappa(s,a)} \subseteq \mathcal{E}$ be the subset of experiences such that any experience $(s_t, a_t, s_{t+1}) \in \mathcal{E}_{\kappa(s,a)}$ would lead to $\kappa(s,a) > 0$ given $\kappa(s,a) = 0$ before. Recall that m is the number of unknown parameters of the (relational) transition model T. As described in Section 2.3, we distinguish two different accuracy levels ε_P and ε_T which both influence the overall accuracy ε in the sample complexity of a PAC-MDP reinforcement learning algorithm (Definition 1). An ε_P -accurate planner finds plans a whose value differs by at most ε_P from the value of the optimal plan a*: $Q(s,\mathrm{a}) > Q(s,\mathrm{a}^*) - \varepsilon_P$. An ε_T-accurate model learner $\mathcal{L}$ makes predictions $\hat{y} = \mathcal{L}(x)$ for input x which differ by at most ε_T from the true value y: $|\hat{y} - y| < \varepsilon_T$. We get the following lemma on the exploration efficiency of REX:

Lemma 2 *Assume* REX *uses an ε_P-accurate planner and a KWIK learner $\mathcal{L}$ to learn the transition model $\hat{T}$. If for a given probability of failure $\delta \in (0,1)$* REX *employs a count function $\kappa(s,a)$ such that it holds with confidence at least $1-\delta$ that*

$$\forall s,a: \qquad \kappa(s,a) \geq \zeta \;\Rightarrow\; \mathcal{L}(s,a)\,\textit{is}\ \varepsilon_T\textit{-accurate and}\ \mathcal{L}(s,a) \neq \perp \quad \textit{and} \tag{4}$$

$$|\mathcal{E}_{\kappa(s,a)}| \geq \textit{polynomial}(m) \;\Rightarrow\; \kappa(s,a) \geq \zeta\,, \tag{5}$$

then REX *in the* R-MAX *variant is PAC-MDP.*

Proof First assume we had equivalence $\iff$ in Equation (4). Then, for the R-MAX option in Algorithm 2, this corollary is a direct implication of the original results for KWIK-R-MAX (Li, 2009): in line 19 of Algorithm 2 the condition $\kappa(s,a) \geq \zeta$, using the count function to decide knownness of states and actions, is identical to using the KWIK learner condition $\mathcal{L}(s,a) \neq \perp$. To show the lemma with one-sided implication in Equation (4), we define a second learner $\mathcal{L}'$ with $\kappa(s,a) < \zeta \Rightarrow \mathcal{L}'(s,a) = \perp$ and $\mathcal{L}'(s,a) \neq \perp \Rightarrow \mathcal{L}'(s,a) = \mathcal{L}(s,a)$ for all s,a. $\mathcal{L}'$ is a KWIK learner due to the condition in Equation (5) and for $\mathcal{L}'$ equivalence in Equation (4) holds. ■

The condition $\kappa(s,a) \geq \zeta \;\Rightarrow\; L(s,a) \neq \perp$ describes a "matching" between the KWIK learner and the definition of the count function. Motivated by Walsh's concrete relational KWIK learner and our concrete REX instance described below, we mention another, more special case condition for a KWIK learner to match with a count function.

Corollary 3 *Assume* REX *uses an* ε_P*-accurate planner and a KWIK learner* $\mathcal{L}$ *to learn the transition model* $\hat{T}$*. If it employs a count function* $\kappa_Q(s,a)$ *defined as in Equation (2) indirectly via a set* $Q = \{q_1,..,q_n\}$ *of queries where the queries are mutually exclusive (*$q_i(s,a) = true \rightarrow \forall k \neq i : q_k(s,a) = false$*) and with confidence at least* $1-\delta$

$$\begin{aligned} \forall q \in Q: \qquad & c_{\mathcal{E}}(q) \geq \zeta \;\Rightarrow\; \forall s,a \textit{ with } q(s,a) = true: \\ & \qquad\qquad\qquad \mathcal{L}(s,a) \textit{ is } \varepsilon_T\textit{-accurate and } \mathcal{L}(s,a) \neq \perp \\ \textit{and } \; & c_{\mathcal{E}}(q) \geq polynomial(m) \;\Rightarrow\; c_{\mathcal{E}}(q) \geq \zeta\,, \end{aligned}$$

then REX *in the* R-MAX *variant is PAC-MDP.*

The corollary follows directly from the previous lemma since if $\kappa_Q(s,a) \geq \zeta$, then there is a $q \in Q$ with $\exists \sigma : s \wedge a \models \sigma(q)$ and $c_{\mathcal{E}}(q) \geq \zeta$.

This special case is interesting since Walsh assumes such mutually exclusive contexts in his learning algorithms. Similarly, the learning algorithm of Pasula et al. tries to ensure that experienced states map to unique contexts (thus, heuristic disjoint queries). Therefore, in those special cases of mutually exclusive queries, a KWIK learner that matches a count function has to ensure to respond only a polynomial number of times with $\perp$ for each abstract context.

The above results are limited to REX in the R-MAX variant. Analogous to KWIK-R-MAX, one may conceive KWIK-E^3 which uses a KWIK learner $\mathcal{L}$ to estimate a transition model $\hat{T}$, relating to REX in the E^3 variant. This generalizes the original E^3 by replacing the counts on states and actions with the certainty estimate of $\mathcal{L}$: if $\mathcal{L}(s,a) = \perp$, then this state-action pair is considered unknown with a transition to the special state $\tilde{s}$ (which has reward 0 in M_{exploit} and reward R_{max} in M_{explore}). While in our experiments we investigate both REX in the R-MAX and E^3 variant, a proof that KWIK-E^3 is PAC-MDP is beyond the scope of this paper. To our knowledge, plain E^3 has not been shown to be strictly PAC-MDP in the sense of Definition 1, either. It is informative to see why we cannot make a proof analogous to the one of Li (2009) for KWIK-R-MAX. Li's proof is based on a theorem of Strehl et al. (2009) tailored to R-MAX-style exploration. For a given exploration algorithm, this theorem defines sufficient conditions on the value function of the algorithm's policy which ensure that the algorithm is PAC-MDP: the "optimism" condition states that this value function is lower bounded by the optimal value function in the true MDP—hence, the algorithm never misses the opportunity to harvest large rewards; the "accuracy" condition states that the value function is upper bounded by the value in a model with the true transitions for the known states—hence, the transitions need to be well estimated by the algorithm. These sufficient conditions cannot be transferred directly to E^3 since in E^3 there is not a *single* value function consistent with the policy that would fulfill these upper and lower bounds. In the exploration mode, E^3 plans more aggressively than R-MAX for unknown states from a known state and ignores rewards in the known states, thus potentially underestimating the expected value instead of overestimating it as R-MAX does. Nonetheless, it appears likely that KWIK-E^3 is PAC-MDP, but a proof has to be made along different lines than for KWIK-R-MAX.

In this subsection, we have established general conditions under which REX is a PAC-MDP reinforcement learning algorithm. In the next subsection, we use these conditions to take a closer look at the exploration efficiency of the concrete REX instance we investigate in our experiments.

3.4 A Model-Based Reinforcement Learner for Relational Domains with Fully Unknown Transition Dynamics

Our relational exploration framework REX is independent of the concrete choices for the transition model representation and learner, the planning algorithm and the relational count function. Here, we propose a concrete instance which we will use in our evaluation. To our knowledge, this instance of REX is the first empirically evaluated relational model-based reinforcement learning algorithm which learns and exploits full-fledged models of transition dynamics. It uses NID rules to learn a transition model $\hat{T}$, plans with the PRADA algorithm and uses the contexts of learned NID rules to estimate the count functions $\kappa_\Phi(s)$ and $\kappa_\Phi(s,a)$. The rule contexts are learned from experience and provide compact descriptions of situation classes. Thus, a state is known if all actions in this state are known. An action is known if there is sufficient confidence in its covering rule. The confidence of a rule depends on the number of experiences in $\mathcal{E}$ it explains, as described above. Our context-based approach to estimate the count function is simple, but empirically effective. We are certain that more elaborate and efficient exploration strategies can be derived from the above principles in the future.

Given these concrete instantiations of the learner, planner and count function components we reconsider briefly the theoretical guarantees. The following corollary shows under which assumptions this REX instance is PAC-MDP.

Corollary 4 *If PRADA uses exact inference and the unknown true transition model T can be modeled by means of NID rules with mutually exclusive contexts, without deictic references and without the special noise outcomes, then there exists a learner $\mathcal{L}$ for such NID rules such that* REX *in the* R-MAX *variant with PRADA, $\mathcal{L}$ and a context-based count function $\kappa_\Phi(s,a)$ is PAC-MDP.*

Proof To learn such NID rules, REX can use a KWIK learner built from a combination of the KWIK learners for the individual rule components proposed by Walsh (2010). If we assume exact inference, PRADA can be adapted to produce ε_P-accurate plans. Then, the corollary follows directly from Corollary 3. ■

Corollary 4 assures us that the exploration strategy of our REX instance is theoretically justified. However, the practicability of our REX instance relies on properties of the planner and learner components which violate the assumptions of the corollary. In relational domains, we easily have to deal with thousands if not millions of random variables. Unfortunately, exact propositional inference has exponential cost in the treewidth of the ground model, making it infeasible for most real-world applications. Even recent advances in lifted exact inference approaches—handling whole sets of indistinguishable variables and factors together—are still extremely complex, generally do not scale to realistic domains, and have only been applied to very small artificial problems. Thus, using approximate inference in PRADA is crucial for the efficiency of planning. While this makes it difficult to provide optimality guarantees, in practice PRADA produces reliably effective plans in difficult noisy settings if its number of action-sequence samples is sufficiently high (Lang and

Toussaint, 2010). Using approximate inference also implies that our REX instance plans through partially unknown states. In general, it is unclear how to efficiently build and exclusively use an MDP of known relational states. However, in each state PRADA, and thus REX, only takes known actions for planning into account: PRADA only considers those actions in a state for which it has a confident unique covering rule.

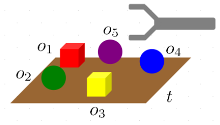
Furthermore, concerning the learning component of REX, deictic references and noise outcomes are expressive language constructs which are essential for learning and modeling the dynamics in practical applications (Pasula et al., 2007). There are, however, no KWIK-learners or other algorithms with theoretical guarantees for learning full NID rules. As noted by Walsh (2010) (p. 64), aside from not covering deictic references, his learning algorithms "would be inappropriate in a schema that allowed for 'noise' or 'miscellaneous' outcomes. [...] In such a situation, a heuristic scoring function that tries to minimize the weights on the noise term is likely a better solution." The heuristic learning algorithm for NID rules by Pasula et al. (2007) which we use in our evaluation cannot guarantee that the learned rules, and thus our learned count functions of known states and actions, converge to near-optimal models. As only contexts of rules with sufficient empirical evidence are used to estimate the count functions, however, the decisions for knowing states and actions are stable in practice. In our investigated domains, independent learning runs always converged to the same or very similar rule-sets and hence to the same learned count functions.

3.5 Illustrative Example

In Table 4, we present an example of an agent using the instance of our REX framework described in Section 3.4 with the E^3 strategy in a robot manipulation domain. In the beginning, the robot is given a relational vocabulary to describe the world on a symbolic level. It has the ability to convert its perceptions into the corresponding symbolic representation. Furthermore, it can execute two different types of motor primitives for grabbing objects and putting them on other objects. It can trigger these motor primitives by the symbolic actions $grab(X)$ and $puton(X)$. These actions are always executed and their effects depend on the context. Thus, in the example scenario with objects $O = \{o_1, o_2, o_3, o_4, o_5, t\}$ the action space of the robot consists of the actions $A = \{\ puton(t)$, $puton(o_1)$, $puton(o_2)$, $puton(o_3)$, $puton(o_4)$, $puton(o_5)$, $grab(o_1)$, $grab(o_2)$, $grab(o_3)$, $grab(o_4)$, $grab(o_5)\ \}$. See Toussaint et al. (2010) for a corresponding real-world robotic setup.

The robot is told to build a tower from the objects on the table. The robot does not know, however, how its actions change the state of the world. It has to learn this from experience and use its insights to achieve its goal—a prototypical model-based reinforcement learning situation. The robot will apply relational exploration to learn as much as possible about the dynamics of its world in as little time as possible. It uses NID rules to learn and represent the transition dynamics. Based on the contexts of these NID rules $Q = \{\Phi_i\}_i$ and its experiences $\mathcal{E}$, it estimates the counts of known states and actions as described above. Here, we assume the robot is confident about a rule if this rule explains at least two of its experiences and hence set $\zeta = 2$.

At $\boldsymbol{t = 0}$, the robot starts with zero knowledge about the transition dynamics. As s_0 is unknown, it performs a direct exploration action. All actions are equally unknown so it chooses randomly to grab cube o_1 and learns rule r_1 from the experience (s_0, a_0, s_1). At $\boldsymbol{t = 1}$, although all actions in s_1 and thus s_1 itself are unknown, the robot is less uncertain about the $grab(\cdot)$ actions for the objects lying on the table: these are covered by the rule r_1 which explains its single experience. It

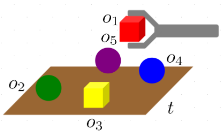


$\boldsymbol{t=0}$

No rule learned yet.

Direct exploration

Next action: $a_0 = \mathit{grab}(o_1)$



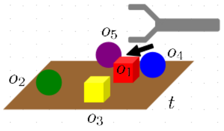
$\boldsymbol{t=1}$

Learned rule r_1

$$\begin{array}{l} \mathit{grab}(X):\quad \mathit{on}(X,Y) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{inhand}(X), \neg \mathit{on}(X,Y) \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Direct exploration

Next action: $a_1 = \mathit{puton}(o_4)$



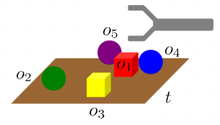
$\boldsymbol{t=2}$

Learned rule r_2

$$\begin{array}{l} \mathit{puton}(X):\quad \mathit{inhand}(Y), \mathit{table}(Z) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{on}(Y,Z), \neg \mathit{inhand}(Y) \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Direct exploration

Next action: $a_2 = \mathit{puton}(o_3)$



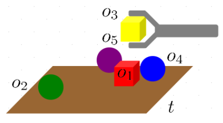
$\boldsymbol{t=3}$

Learned rule r_3

$$\begin{array}{l} \mathit{puton}(X):\quad \mathit{inhandNil}() \\ \rightarrow \left\{ \begin{array}{l} 1.0: - \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Direct exploration

Next action: $a_3 = \mathit{grab}(o_3)$



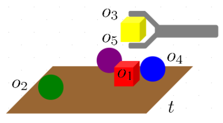
$\boldsymbol{t=4}$

Confirmed rule r_1

$$\begin{array}{l} \mathit{grab}(X):\quad \mathit{on}(X,Y) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{inhand}(X), \neg \mathit{on}(X,Y) \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Direct exploration

Next action: $a_4 = \mathit{grab}(o_3)$



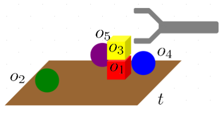
$\boldsymbol{t=5}$

Learned rule r_4

$$\begin{array}{l} \mathit{grab}(X):\quad \mathit{inhand}(X) \\ \rightarrow \left\{ \begin{array}{l} 1.0: - \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Direct exploration

Next action: $a_5 = \mathit{puton}(o_1)$



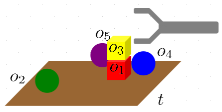
$\boldsymbol{t=6}$

Learned rules r_5 and r_6

$$\begin{array}{l} \mathit{puton}(X):\quad \neg \mathit{ball}(X), \mathit{inhand}(Y) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{on}(Y,X), \neg \mathit{inhand}(Y) \\ 0.0: \text{noise} \end{array} \right. \\ \mathit{puton}(X): \mathit{ball}(X), \mathit{inhand}(Y), \\ \qquad\qquad \mathit{table}(Z) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{on}(Y,Z), \neg \mathit{inhand}(Y) \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Direct exploration

Next action: $a_6 = \mathit{puton}(o_2)$



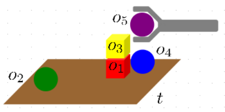
$\boldsymbol{t=7}$

Confirmed rule r_3

$$\begin{array}{l} \mathit{puton}(X):\quad \mathit{inhandNil}() \\ \rightarrow \left\{ \begin{array}{l} 1.0: - \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Planned exploration

Next action: $a_7 = \mathit{grab}(o_5)$



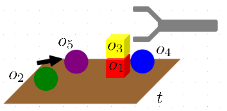
$\boldsymbol{t=8}$

Confirmed rule r_1

$$\begin{array}{l} \mathit{grab}(X):\quad \mathit{on}(X,Y) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{inhand}(X), \neg \mathit{on}(X,Y) \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Planned exploration

Next action: $a_8 = \mathit{puton}(o_2)$



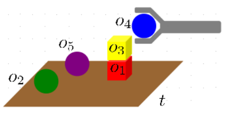
$\boldsymbol{t=9}$

Confirmed rule r_6

$$\begin{array}{l} \mathit{puton}(X): \mathit{ball}(X), \mathit{inhand}(Y), \mathit{table}(Z) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{on}(Y,Z), \neg \mathit{inhand}(Y) \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Planned exploration

Next action: $a_9 = \mathit{grab}(o_4)$



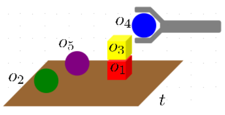
$\boldsymbol{t=10}$

Confirmed rule r_1

$$\begin{array}{l} \mathit{grab}(X):\quad \mathit{on}(X,Y) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{inhand}(X), \neg \mathit{on}(X,Y) \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Planned exploration

Next action: $a_{10} = \mathit{grab}(o_4)$



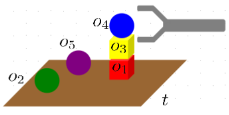
$\boldsymbol{t=11}$

Confirmed rule r_4

$$\begin{array}{l} \mathit{grab}(X):\quad \mathit{inhand}(X) \\ \rightarrow \left\{ \begin{array}{l} 1.0: - \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Planned exploration

Next action: $a_{10} = \mathit{puton}(o_3)$



$\boldsymbol{t=12}$

Confirmed rule r_5

$$\begin{array}{l} \mathit{puton}(X):\quad \neg \mathit{ball}(X), \mathit{inhand}(Y) \\ \rightarrow \left\{ \begin{array}{l} 1.0: \mathit{on}(Y,X), \neg \mathit{inhand}(Y) \\ 0.0: \text{noise} \end{array} \right. \end{array}$$

Done.

Table 4: *Example of Relational* E^3. A robot manipulates objects scattered on a table by means of motor primitives triggered by symbolic actions $\mathit{puton}(\cdot)$ and $\mathit{grab}(\cdot)$. The robot is told to build a tower. It starts with zero knowledge ($\mathcal{E} = \emptyset$) and learns from experience how its actions change the state of the world. The arrows at $t = 2$ and $t = 9$ indicate that the manipulated objects have fallen off balls.

is most uncertain about grabbing the inhand object (which is not covered by r_1) and all $puton(\cdot)$ actions which are not covered by any rule. The robot chooses $puton(o_4)$ randomly among these most unknown actions. As o_4 is a ball, the inhand-held cube o_1 falls from o_4 on the table t, resulting in state s_2. The robot generalizes this experience in form of the rule r_2 that $puton(X)$ will lead to putting the inhand-object on the table. This false generalization is uncertain, however, as it is only covered by one experience.

At $\boldsymbol{t = 2}$, the robot is most uncertain about the $puton(\cdot)$ actions which are not covered by any of its learned rules (the previously learned rule r_2 requires $inhand(Y)$ in its context which is not fulfilled in s_2). Therefore, the robot chooses randomly among them and performs $puton(o_3)$. It observes no effects and learns the rule r_3, predicting *puton* actions in contexts where nothing is held inhand. At $\boldsymbol{t = 3}$, the robot is equally uncertain about all actions (all actions are covered by rules which explain exactly one experience). It chooses randomly $grab(o_3)$. The experience (s_3, a_3, s_4) confirms the rule r_1 about which it is certain by now as r_1 explains two experiences. At $\boldsymbol{t = 4}$, $grab(o_3)$ is the only action which is not covered by any rule and therefore performed. The resulting state s_5 is not different from s_4, resulting in the learned rule r_4.

At $\boldsymbol{t = 5}$, the most uncertain actions, $grab(o_3)$ and all *puton*-actions, are covered by rules with confidence one (that is, explaining one experience). The robot chooses randomly $puton(o_1)$. This provides an insightful experience: by comparing the experiences (s_1, a_1, s_2) and (s_5, a_5, s_6), the robot “understands” that rule r_2 is a false generalization and instead learns r_5 for putting on cubes and the table and r_6 putting on balls. At $\boldsymbol{t = 6}$, all *grab*-actions are known due to the confidence in rule r_1: the robot can predict their effects with confidence. In contrast, it is uncertain about all *puton*-actions as rule r_3 explains only one experience, so it randomly chooses $puton(o_2)$. The resulting experience confirms r_3 about which it is certain then.

At $\boldsymbol{t = 7}$, the robot can predict the effects of all actions with certainty. Therefore, it tries to exploit its knowledge about known states to plan for its goal to build a tower. The rule r_5 (modeling *puton*-actions for cubes) required for tower building, however, is still uncertain and thus the corresponding actions and states are unknown and cannot be considered in planning. Therefore, exploitation fails and the robot performs planned exploration instead: it plans for unknown states in which its rules r_4, r_5 and r_6 can be tested. Such states are reached by *grab*-actions and the robot chooses randomly to perform $grab(o_5)$. At $\boldsymbol{t = 8}$, the planned exploration of the last time-step allows to confirm several rules by performing one of the unknown *puton* actions or the likewise unknown $grab(o_5)$. The robot chooses randomly $puton(o_2)$. The resulting state s_9 confirms r_6 about which it is certain by now. At $\boldsymbol{t = 9}$, like in s_7 the robot performs planned exploration and grabs a random object, namely $grab(o_4)$.

At $\boldsymbol{t = 10}$, the unknown actions are $grab(o_4)$, $puton(o_1)$, $puton(o_3)$ and $puton(t)$ whose covering rules explain only one experience. The robot chooses randomly $grab(o_4)$ whose outcome in s_{11} confirms the rule r_4. At $\boldsymbol{t = 11}$, from the remaining three unknown actions, it chooses randomly $puton(o_3)$, confirming the rule r_5. At $\boldsymbol{t = 12}$, the highest possible tower has been built. Hence, the robot cannot exploit its knowledge to build an even higher tower. Similarly, planned exploration for unknown states fails and the robot concludes that it is done.

In this example, there have been no exploitation steps as in the last time-step achieved by direct and planned exploration, the highest possible reward has already been achieved. If there were more cubes, however, the robot could successfully exploit its knowledge to achieve even more rewarding states from s_{12}.

To keep this illustrative example short, we simplified in some respects: First, the robot makes most often the correct generalizations, even if competing false generalizations have the same statistical evidence. For instance, from experience (s_2, a_2, s_3) the robot could also generalize that $puton(X)$ does not have effects if X is a cube (instead of if its hand is empty). A second simplification in our example is our neglect of noise in the actions of the robot. Note however that our algorithms account for stochastic actions and our experimental evaluation is performed in intrinsically noisy domains. The third simplification is that we determined the "random" choices of the robot to be informative actions. For instance, if the robot had chosen $puton(o_1)$ in s_{11}, it might have had to revise its rule r_5 to incorporate $clear(X)$ and to come up with a rule for putting on objects which are not clear, leading to a more accurate model requiring more exploration steps.

4. Evaluation

We demonstrate that integrating our approach REX with NID rules and the planner PRADA as described in Section 3.4 leads to a practical exploration system for relational RL which outperforms established non-relational techniques on a large number of relevant problems. Our results show that REX explores efficiently worlds with many objects and transfers learned knowledge to new situations, objects and, in contrast to model-free relational RL techniques, even to new tasks. To our knowledge, this is the first evaluation of a model-based reinforcement learning agent which learns both the complete structure as well as the parameters of relational transition models.

We compare five methods inspired by E^3 and R-MAX based on different state and action representations:

- **flat E^3**: learning and exploration on an enumerated representation;
- **factored E^3**: learning and exploration on a factored propositional representation;
- **relational ε-greedy**: learning on a relational representation, ε-greedy exploration;
- **relational E^3**: learning and exploration on a relational representation; and
- **relational R-MAX**: learning and exploration on a relational representation.

It is not our goal to explicitly compare E^3 and R-MAX. We present results for both, relational E^3 and relational R-MAX, to demonstrate that REX is a practical, effective approach with either strategy. For the non-relational baselines, we have arbitrarily chosen E^3, but we might have likewise used R-MAX with similar results to expect.

We use NID rules to learn and represent transition models T in *all* investigated methods for two reasons: *(i)* there is an effective learning algorithm for NID rules, and *(ii)* we can express and learn transition models on all representation levels with NID rules which allows for a consistent comparison. Relational E^3, R-MAX and ε-greedy learn full-fledged abstract NID rules as described in Section 2.4.1. Our factored E^3 learns propositional (that is, grounded) NID rules using a slightly modified learning algorithm; ground literals imitate the propositional state attributes. (The factored E^3 algorithm of Kearns and Koller (1999) cannot learn the model structure and thus cannot be used as an alternative baseline.) Flat E^3 uses pseudo propositional NID rules where the rule context describes a complete ground relational state; hence, a rule is only applicable in a specific state and this approach corresponds to the original E^3. Exemplary rules for all representations are shown in

$$grab(X): \quad on(X,Y),\ ball(X),\ cube(Y),\ table(Z) \\ \rightarrow \begin{cases} 0.7 & : \ inhand(X),\ \neg on(X,Y) \\ 0.2 & : \ on(X,Z),\ \neg on(X,Y) \\ 0.1 & : \ \text{noise} \end{cases}$$

(a) Abstract NID rule

$$grab(d): \quad on(d,b) \\ \rightarrow \begin{cases} 0.7 & : \ inhand(d),\ \neg on(d,b) \\ 0.2 & : \ on(d,t),\ \neg on(d,b) \\ 0.1 & : \ \text{noise} \end{cases}$$

(b) Factored propositional NID rule

$$grab(d): \quad on(a,t), on(b,t), on(c,a), on(d,b), \neg on(a,b), \neg on(a,c) \ldots, \neg inhand(a), \ldots \\ \rightarrow \begin{cases} 0.7 & : \ inhand(d),\ \neg on(d,b) \\ 0.2 & : \ on(d,t),\ \neg on(d,b) \\ 0.1 & : \ \text{noise} \end{cases}$$

(c) Flat “NID rule”

Table 5: *Illustration of NID rules on different representation levels.* (a) The abstract rule uses variables to generalize over objects (same as in Table 3). (b) The factored propositional rule imitates propositional state attributes by using only ground literals. (c) The flat rule specifies a complete state in its context. Note that the propositional rules (b) and (c) can renounce on typing predicates such as $cube(\cdot)$ as they do not abstract from object identities. The presented rules might have been learned from the grab experiences shown in Table 2.

Table 5. In each domain, the parameters for the rule learning algorithm, α and p_{min}, were chosen by a preliminary coarse heuristic search; the rule mutation operators and their order were the same across all experiments and exploration algorithms.

We use our novel exploration approach REX introduced in Section 3.2 and Section 3.4 as a concrete instance of relational E^3 and R-MAX and a factored propositional variant of REX as factored propositional E^3. Thus, we learn (abstract or propositional) NID rules for the last action after each new observation from scratch using the algorithm of Pasula et al. (2007) (appropriately modified for the propositional representations) and employ PRADA (Lang and Toussaint, 2010) for exploitation or planned exploration. The reward function is not learned, but provided to the agent. PRADA plans in the grounded relational representation and therefore can deal with rules on all abstraction levels. PRADA's planning horizon and its number of plan samples were both set to large values across all experiments to ensure that the planning performance of PRADA is reliable and has negligible influence on the reported results. To estimate a count function for known states and actions, REX and similarly its propositional counterparts use the contexts of rules $Q = \{\phi_r\}$. Similarly as done in the evaluation of KWIK-R-MAX (Li, 2009), we set the threshold ζ for known states and actions

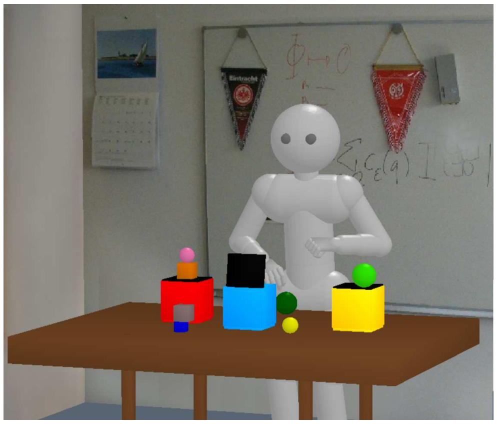
(Section 2.3) heuristically as we cannot derive it from theory due to the heuristic nature of the learning algorithm for NID rules. For all investigated E^3 and R-MAX algorithms, we set $\zeta = 2$; hence, an action is considered known in a state if its covering rule explains two experiences in $\mathcal{E}$. Since our investigated domains have mostly comparable levels of stochasticity, we did not optimize ζ for the individual domains. Although $\zeta = 2$ may not be optimal in each scenario, it allows the agent to explore the environments of our experiments within a reasonable number of actions (< 100), while providing some confidence in the estimate. We have implemented REX and its propositional counterparts, the learning algorithm for NID rules and the planning algorithm PRADA in C++.[1]

We compare our approach REX to relational ε-greedy which is the established exploration technique in relational RL approaches (see the discussion in Section 1.2). Relational ε-greedy is not a simple baseline, but an effective technique which learns abstract relational transition models (in our case NID rules) and uses them for exploitation—thereby employing the same set of known states as REX (relational E^3 and R-MAX). In contrast to REX, it performs a random action for exploration. Relational ε-greedy often profits from its optimism to try to exploit: in contrast to REX, it does not require to fully know a state before it can start exploitation there. In our experiments, we used decaying schemes to set ε dynamically, enabling a high exploration rate in the beginning and more exploitation with an increasing number of time-steps.

We emphasize that we are not aware of any other relational exploration approach for learning full transition models apart from ε-greedy which we could use as a baseline in our evaluation. As discussed in detail in Section 1.2, the conceptual approach of Walsh (2010) based on a relational R-MAX framework has not been demonstrated in practice and is supposedly intractable in large domains like ours. More importantly, it is limited to a relational language which is less expressive than the one required for a large fraction of our investigated scenarios.

Our first test domain, called *Robot manipulation domain*, is a simulated complex 3D desktop environment where a robot manipulates cubes, balls and boxes scattered on a table (Table 6). (In the following, we refer to this simulated robot when speaking of “the robot”.) This is a relevant scenario in robotics: “competent pick and place operations may provide a sufficient functional basis for the manipulation requirements of a many of the targeted applications [of modern robotics]” (Christensen, 2009, p. 56). Any autonomous system which wants to deliberately manipulate objects has to master some sort of abstract reasoning at a symbolic level. In current state-of-the-art robotics, however, this problem is largely unresolved and perceived as a bottleneck. Likewise, the ability to generalize from experiences is viewed as a key desired capability of robots: “It is largely perception and machine learning that distinguish a robot from an ordinary machine” (Christensen, 2009, p. 78). We will see that our approach REX provides a practical solution to these problems. We use a 3D rigid-body dynamics simulator (ODE) that enables a realistic behavior of the manipulated objects. For instance, piles of objects may topple over or objects may even fall off the table (in which case they become out of reach for the robot). Depending on their type, objects show different characteristics. For example, it is almost impossible to successfully put an object on top of a ball, and building piles with small objects is more difficult. The robot can grab objects, try to put them on top of other objects, in a box or on the table. Boxes have a lid; special actions may open or close the lid; taking an object out of a box or putting it into it is possible only when the box is opened. *The actions of the robot are affected by noise* so that resulting object piles are not straight-aligned. We

[1] The website `http://userpage.fu-berlin.de/tlang/explore/` provides our code of PRADA, the learning algorithm of NID rules and the robot manipulation simulator as well as videos of exemplary rounds in the robot manipulation domain.



Given to the robot:

- Symbolic vocabulary to describe states ($on(X,Y)$, $inhand(X)$, $clear(X)$, $\dots$)
- Ability to convert continuous perceptions into a symbolic representation
- Symbolic actions triggering noisy motor primitives (always executed, effects depend on contexts) ($grab(X)$, $puton(X) \dots$)
- Reward function

Not given to the robot:

- Transition model for symbolic actions
  "In which contexts do the motor primitives have which effects?"

Table 6: In our robot manipulation domain, a simulated robot has to explore a 3D desktop environment with cubes, balls and boxes of different sizes and colors to master various tasks.

assume full observability of triples (s,a,s') that specify how the world changed when an action was executed in a certain state. We represent states with predicates $cube(X)$, $ball(X)$, $box(X)$, $table(X)$, $on(X,Y)$, $contains(X,Y)$, $out(X)$, $inhand(X)$, $upright(X)$, $closed(X)$, $clear(X) \equiv \forall Y.\neg on(Y,X)$, $inhandNil() \equiv \neg\exists X.inhand(X)$ and functions $size(X)$, $color(X)$. These symbols are obtained by querying the state of the simulator and translating it according to simple hand-made guidelines, thereby sidestepping the difficult problem of converting the agent's observations into an internal representation. For instance, $on(a,b)$ holds if a and b exert friction forces on each other and a's z-coordinate is greater than the one of b, while their x- and y-coordinates are similar. If there are o objects and f different object sizes and colors in a world, the state space is huge with $f^{2o}2^{2o^2+8o}$ different states (not excluding states one would classify as "impossible" given some intuition about real world physics); in our scenarios, we usually have much more than 10^{50} states. This points at the potential of using abstract relational knowledge for exploration. We define five different types of actions, denoted by different predicate symbols. These actions correspond to motor primitives whose effects and contexts we want to explore, learn and exploit. The $grab(X)$ action triggers the robot to open its hand, move its hand next to X, let it grab X and raise the robot arm again. The execution of this action is not influenced by any further factors. For example, if a different object Y has been held in the hand before, it will fall down on either the table or a third object just below Y; if there are objects on top of X, these are very likely to fall down. The $puton(X)$ action centers the robot's hand at a certain distance above X, opens it and raises the hand again. For instance, if there is an object Z on X, the object Y that was potentially inhand may end up on Z or Z might fall off X. The $openBox(X)$ and $closeBox(X)$ actions only apply to boxes. $openBox(X)$ triggers the robot to move its arm next to a box X and try to open its lid; this is only successful if there is no object on top of this box. If the box is already open, the position of the lid does not change. Similarly, $closeBox(X)$ triggers the robot to move its arm next to the lid of the box X and close it; if the box is already closed, the lid is not moved. The $doNothing()$ action triggers no movement of the robot's arm. The robot might choose this action if it thinks that any other action could be harmful with respect to its expected reward. We emphasize again that actions always execute, regardless of the state of the world. Also, actions which are rather unintuitive for humans such as trying to grab the table or to put an object on top of itself are carried out. The robot has to learn by itself the effects and contexts of such motor primitives.

Our second set of test domains, called *IPPC* in the following, are domains taken from the international planning competition in 2008 (IPPC, 2008). While these domains are defined in the probabilistic planning domain definition language (PPDDL), the transition dynamics of many domains can be represented by NID rules (Lang and Toussaint, 2010). In our experiments, we show that these representations can be explored and learned efficiently. These experiments also demonstrate that our exploration approaches perform equally well when using a restricted representation language, namely without deictic references and noise outcomes which are not allowed by PPDDL.

We perform multiple experiments in our test domains where we pursue the same, similar or different tasks over several rounds:

- **Series 1 – Robot Manipulation Domain**:
  - Experiment 1: Simple task in unchanging worlds (Figure 2, p. 3723)
  - Experiment 2: Generalization to new worlds (Figure 3, p. 3724)
  - Experiment 3: Advanced task in unchanging worlds (Figure 4, p. 3725)
  - Experiment 4: Generalization to new tasks (Figure 5, p. 3726)
- **Series 2 – IPPC Domains**:
  - Experiment 5: Exploding Blocksworld (Figure 6, p. 3727)
  - Experiment 6: Triangle-tireworld (Figure 7, p. 3728)
  - Experiment 7: Search-and-rescue (Figure 8, p. 3729)

In all experiments *the robot starts from zero knowledge ($\mathcal{E} = \emptyset$) in the first round* and carries over experiences to the next rounds. Each round is limited by a maximum number of actions (100 in the robot manipulation domain, varying numbers in the IPPC domains). If the task is not solved by then, the round fails.

We report the success rates and the action numbers (failed trials contribute to the action number with the maximum number of actions). Both are direct measures of the goal-directedness and acting performance of an autonomous agent. Likewise these measures also evaluate the learning performance of the agent: the goal-directed performance depends on the learned rules and the active exploration on the learned count functions. Thus, high success rates and low action numbers indicate a good performance in learning rules and count functions—“good” in the sense of enabling goal-directed behavior.

4.1 Series 1 – Robot Manipulation Domain

In this first series of experiments, we compare our relational exploration approach REX in both variants (E^3 and R-MAX) to relational ε-greedy and to flat and factored E^3 approaches in successively more complex problem settings.

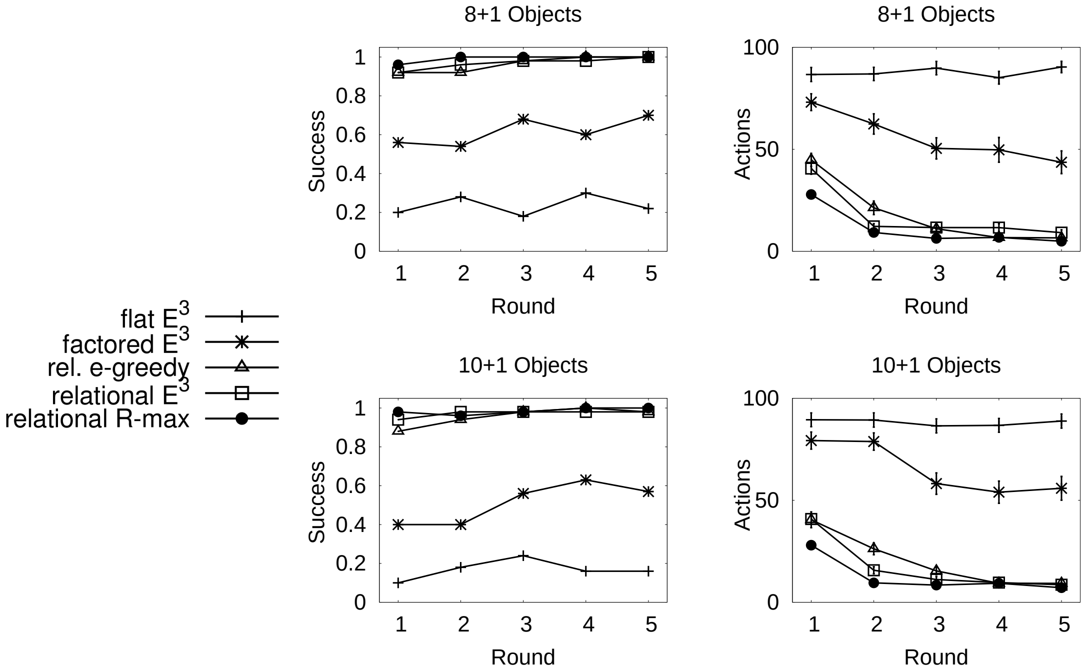


Figure 2: *Experiment 1 (Robot manipulation domain): Simple task in unchanging worlds.* A run consists of 5 subsequent rounds with the same start situations and goal objects. The robot starts with no knowledge in the first round. The success rate and the mean number of actions with standard deviations over 50 runs are shown (5 start situations, 10 seeds).

4.1.1 EXPERIMENT 1: SIMPLE TASK IN UNCHANGING WORLDS

The goal in each round is to stack two specific objects, $on(o_1, o_2)$. To collect statistics we investigate worlds of varying numbers of objects (cubes, balls, boxes and table): for each object number, we create five worlds with different objects. For each such world, we perform 10 independent runs with different random seeds. Each run consists of 5 rounds with the same goal instance and the same start situation. In some worlds, goal objects are put inside boxes in the beginning, necessitating more intense exploration to learn how to deal with boxes. Figure 2 shows that the relational explorers have superior success rates, require significantly fewer actions and reuse their learned knowledge effectively in subsequent rounds. The propositional explorers are overburdened with large numbers of objects—their learned count functions of known states and actions fail to recognize similar situations. In contrast, the relational explorers scale well with increasing numbers of objects. In the larger worlds, our REX approaches, that is relational E^3 and R-MAX, require less rounds than ε-greedy to solve the tasks with few actions: the learned count functions permit effective exploration.

4.1.2 EXPERIMENT 2: GENERALIZATION TO NEW WORLDS

In this series of experiments, the objects, their total numbers and the specific goal instances are different in each round (worlds of 7, 9 and 11 objects). We create 10 problem sequences (each with 5 rounds) and perform 10 trials for each sequence with different random seeds. As Figure 3 shows

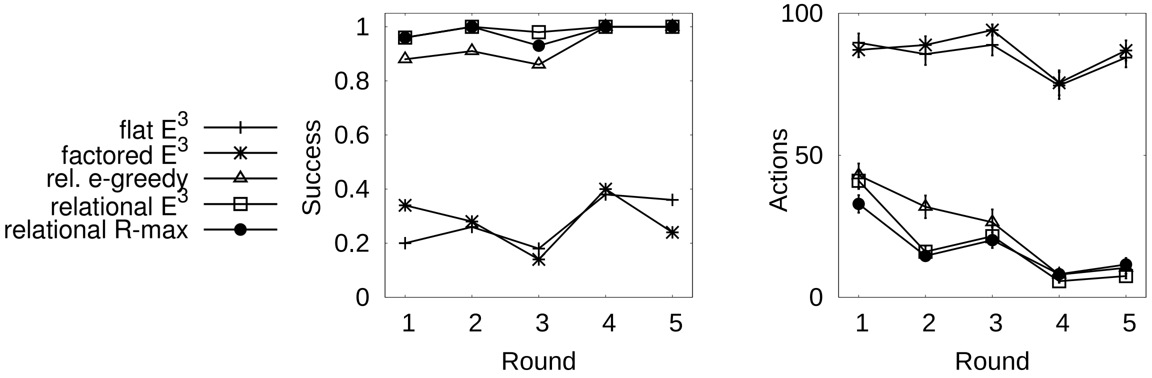


Figure 3: *Experiment 2 (Robot manipulation domain): Generalization to new worlds.* A run consists of a sequence of 5 subsequent problems (corresponding to rounds) with different objects, numbers of objects (6 - 10 cubes/balls/boxes + table) and start situations in each problem. The robot starts with no knowledge in the first round. The success rate and the mean number of actions with standard deviations over 100 runs are shown (10 sequences, 10 seeds).

the performance of the relational explorers is good from the beginning. Their learned rules and count functions for known states and actions enable a stable performance of relational E^3 and R-MAX at a near-optimal level after already 2 rounds, while relational ε-greedy requires 3-4 rounds. This experiment shows that the relational explorers can transfer their learned knowledge to new situations and objects. In contrast, the propositional explorers cannot transfer their knowledge to different worlds due to the limits of their learned rules and count functions and thus neither their success rates nor their action numbers improve in subsequent rounds.

4.1.3 EXPERIMENT 3: ADVANCED TASK IN UNCHANGING WORLDS

The goal in each round is to “clear” the table. 5 movable objects (balls and cubes) of different colors are distributed over the table. To clear up such a movable object, it needs to be put into a box of the same color. We perform 10 independent runs with different random seeds in 5 different start situations. Each run consists of 5 rounds with the same start situation. In start situations, balls and cubes may form piles, lie on top of closed boxes or be contained in boxes of a different color. Figure 4 presents our results. The non-relational exploration approaches fail to solve this task and are omitted in the figure. Relational E^3 and R-MAX outperform relational ε-greedy .

4.1.4 EXPERIMENT 4: GENERALIZATION TO NEW TASKS

Finally, we perform three tasks of increasing difficulty in succession: (i) piling two specific objects in simple worlds with cubes and balls, (ii) in worlds extended by boxes (as in Exp. 1 and 2), and (iii) clearing up the desktop by putting all movable objects into boxes of the same color where the required objects may be partially contained in wrong boxes in the beginning (as in Exp. 3). Each task is performed for three rounds in changing worlds with different goal objects. The results are presented in Figure 5. Non-relational explorers cannot generalize over objects and fail in such

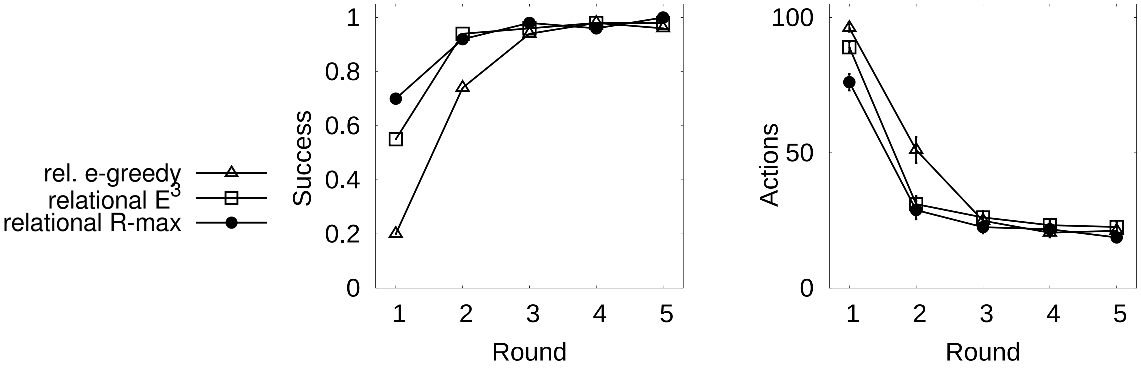


Figure 4: *Experiment 3 (Robot manipulation domain): Advanced task in unchanging worlds*. The goal is to put colored balls and cubes into boxes of the same color. A run consists of 5 subsequent rounds with the same start situation. The robot starts with zero knowledge in the first round. The success rate and the mean estimators of action numbers with standard deviations over 50 runs are shown (5 start situations, 10 seeds).

tasks (see Exp. 2 and 3) so we omit their results here. The results demonstrate that the relational explorers are not only able to generalize over different situations and objects, but transfer the learned knowledge from simple to difficult tasks in the sense of curriculum learning (Bengio et al., 2009). This is shown in the bottom row of Figure 5 where the results of relational E^3 are compared to restarting the learning procedure at the beginning of each new task (that is, in rounds 4 and 7) (the corresponding graphs for relational R-MAX and relational ε-greedy are similar). Furthermore, both relational E^3 and R-MAX benefit from their learned count functions for active exploration and outperform relational ε-greedy clearly.

4.1.5 SUMMARY

The experiments in the robot manipulation domain show that our relational exploration approach REX is a practical and effective full-system approach for exploration in challenging realistic problem settings. Our results confirm the intuitive expectation that relational knowledge improves learning transition models and learning count functions for known states and actions and thus exploration. Experiment 1 shows that relational explorers scale better with the number of objects than propositional explorers. The more challenging Experiments 2 and 3 demonstrate that the principled relational E^3 and R-MAX exploration of REX outperforms the established ε-greedy exploration: the learned count functions permit effective active exploration. Furthermore, these experiments show that REX transfers its knowledge efficiently to new situations and objects—while propositional explorers fail to do so. Finally, Experiment 4 demonstrates that REX is able to transfer its learned knowledge even to new tasks—in contrast to model-free relational approaches.

4.2 Series 2 – IPPC

In the second series of experiments, we demonstrate the effectiveness of REX in domains of the international planning competition. We choose three domains whose transition dynamics can be

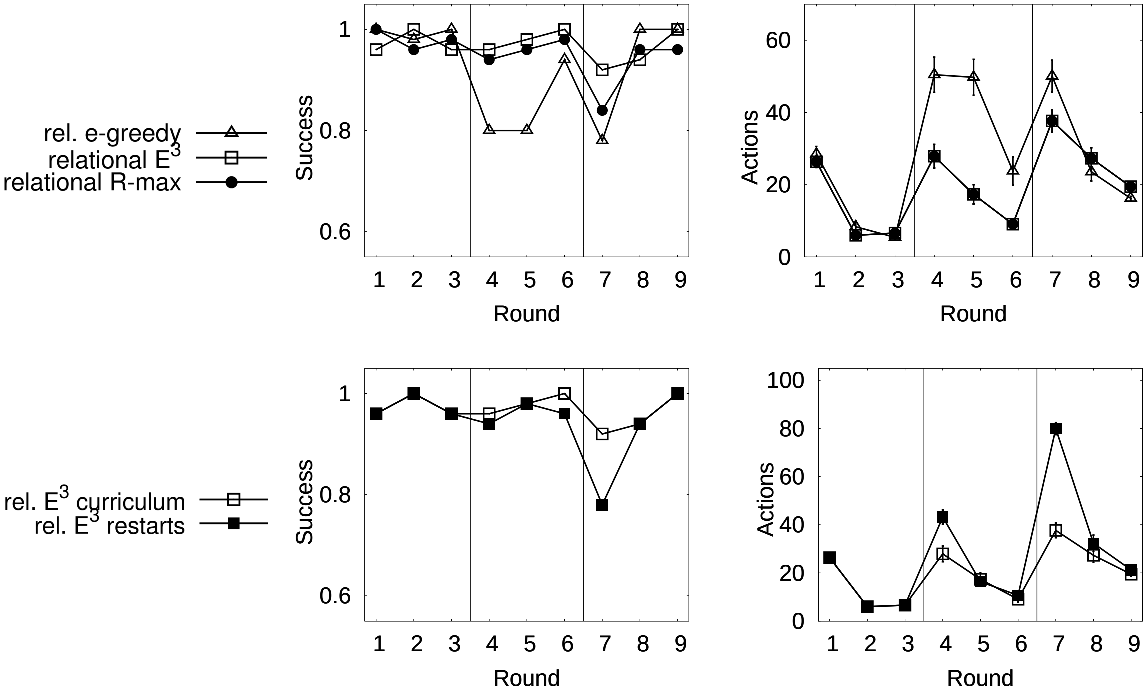


Figure 5: *Experiment 4 (Robot manipulation domain): Generalization to new tasks.* A run consists of a sequence of 9 problems (corresponding to rounds) with different objects, numbers of objects (6 - 10 cubes/balls/boxes + table) and start situations in each problem. The tasks are changed between rounds 3 and 4 and rounds 6 and 7 to more difficult tasks. The robot starts with no knowledge in the first round. The success rate and the mean number of actions with standard deviations over 50 runs are shown (10 sequences, 5 seeds). The *top row* presents the results of different relational exploration approaches. The *bottom row* compares the full curriculum relational E^3 which transfers learned knowledge to new tasks (same as in the top row) with restarting relational E^3 with each new task.

modeled by NID rules. We convert the PPDDL definitions of these domains manually into sets of NID rules and use these as the model of the true world dynamics. The agent tries to estimate the rules from its experiences. As PPDDL does not allow for deictic references and noise outcomes, this series of experiments demonstrates that REX performs also well with restricted relational languages.

For each domain, we present results on representative problem instances. To collect statistics, we perform 50 trials on the same problem instance with different random seeds. Each trial consists of 5 or 10 subsequent rounds (worlds do not change). In the IPPC domains, most actions do not have effects in a given state: in addition to specifying contexts to distinguish different effect sets, the PPDDL action operators define also restrictive action preconditions in the IPPC domains. This is in contrast to the intrinsically noisy robot manipulation domain where actions almost always have an effect. For instance, in the latter domain it is always possible to try to grab an object, be it clear, at the bottom of a pile or in a box. NID rules don't distinguish between contexts and preconditions. To focus on the context learning part, we introduce a restriction for exploration in *all* (relational and

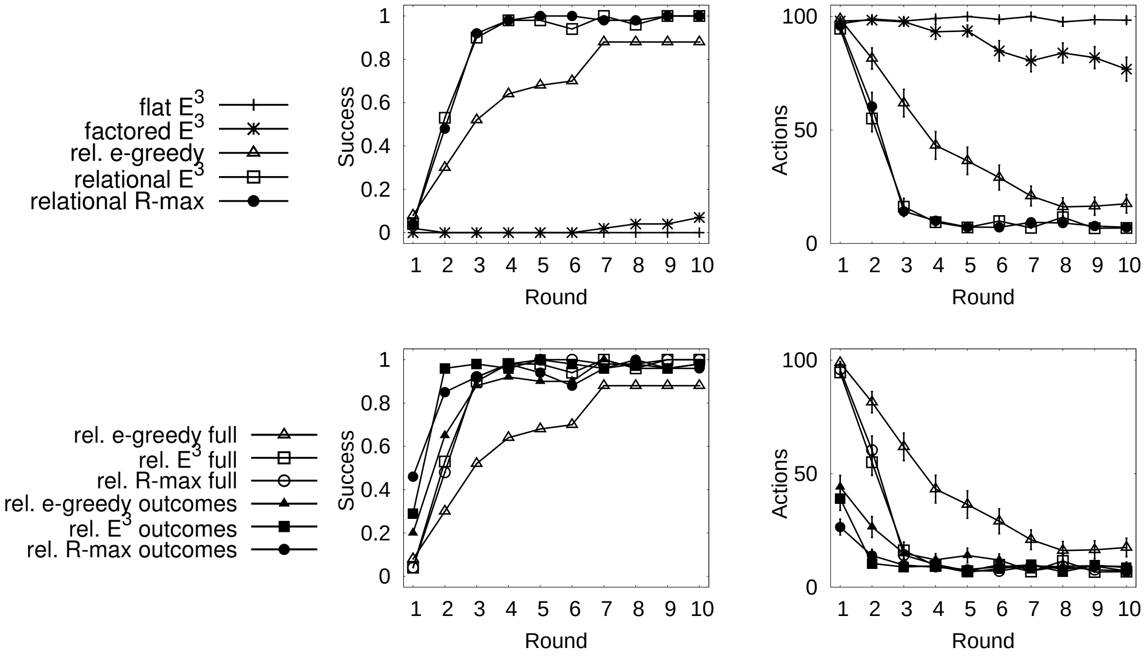


Figure 6: *Experiment 5 (IPPC): Exploding Blocksworld (problem instance 5).* A run consists of 10 subsequent rounds with the same start situation and goal objects. The agent starts with no knowledge in the first round. The success rate and the mean number of actions with standard deviations over 50 runs based on different random seeds are shown. The *top row* presents the results for learning the full transition models. The *bottom row* compares learning the full transition models (same as in the top row) with learning the outcomes and their distributions only when rule contexts are provided a-priori.

non-relational) investigated approaches: actions which have been executed without effects (so that supposedly their preconditions were violated) are forbidden until the next state change.

4.2.1 EXPERIMENT 5: EXPLODING BLOCKSWORLD

The results for problem instance 5 of this domain are displayed in the top row of Figure 6. They show that the propositional explorers almost always fail. Their performance is hampered in particular by the fact that most actions do not have effects in a given state. This is hazardous if one cannot generalize one's experiences over objects, resulting in barely useful estimated count functions for known states and actions: the propositional explorers spend too much time in each state exploring actions without effects. In contrast, the relational explorers learn quickly to solve the task. Both relational E^3 and R-MAX clearly outperform relational ε-greedy in both the success rate as well as the number of required actions, indicating the usefulness of the learned count functions for active exploration.

In the bottom row of Figure 6, the results of the relational approaches are compared to the scenario where the contexts of rules are known a-priori and only the outcomes of rules and their

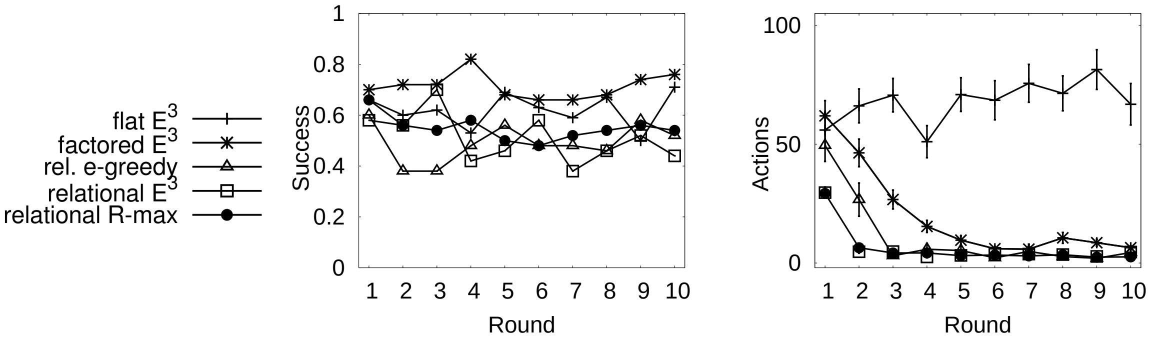


Figure 7: *Experiment 6 (IPPC): Triangle Tireworld (problem instance 1).* A run consists of 10 subsequent rounds with the same start situation and goal specification. The agent starts with no knowledge in the first round. The success rate and the mean estimators of the action numbers with standard deviations over 50 runs based on different random seeds are shown.

probabilities need to be learned. As expected, all relational approaches learn to solve the tasks much faster in this simpler scenario. Relational ε-greedy is still clearly inferior to the principled REX approaches.

4.2.2 EXPERIMENT 6: TRIANGLE TIREWORLD

The results presented in Figure 7 for problem instance 1 show that the success rates of all methods fluctuate in similar value ranges with factored E^3 performing best. In contrast, the smaller action numbers of relational E^3 and R-MAX indicate the advantage of learning and using expressive count functions for active exploration. The difficulty of this domain lies in dead-lock situations (where the agent has a flat tire, but no spare tire is available). The specific contexts for landing in such dead-locks are hard to learn given the limited number of available relevant experiences. Here, our general choice of the threshold for knowing states and actions, $\zeta = 2$, (which we did not optimize for individual domains) is too small: the explorers are too early confident about their learned model. This hurts the relational explorers in particular as their models generalize the most and hence their potentially still inaccurate predictions get applied wrongly more often.

4.2.3 EXPERIMENT 7: SEARCH AND RESCUE

In this domain, the agent can collect intermediate and final rewards in addition to solving the task. We present the total rewards with the success rates and the action numbers for problem instance 8 in Figure 8. Overall, relational R-MAX performs best with respect to all measures; relational E^3 has similar success rates and action numbers, but collects less rewards. While flat E^3 performs worst, here factored E^3 most often outperforms relational ε-greedy. This shows the benefit of a principled exploration strategy based on learned count functions of known states and actions.

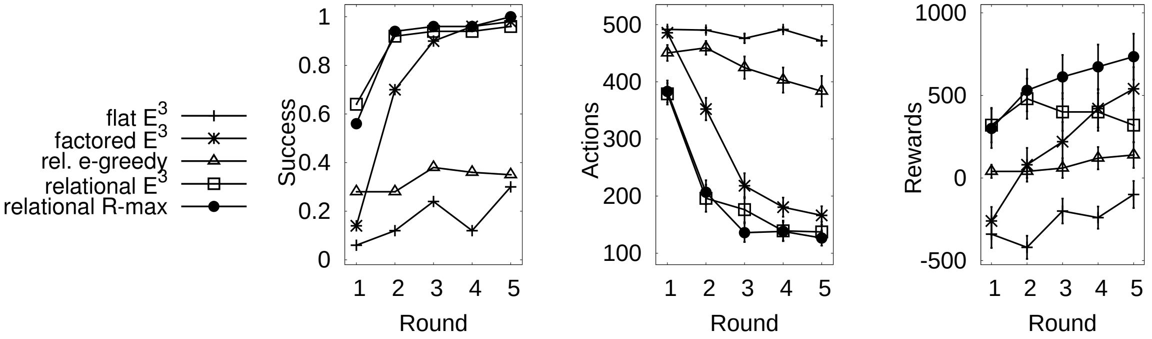


Figure 8: *Experiment 7 (IPPC): Search and Rescue (problem instance 8).* A run consists of 5 subsequent rounds with the same start situation and goal specification. The agent starts with no knowledge in the first round. The success rate and the mean estimators of the action numbers and the rewards with standard deviations over 50 runs based on different random seeds are shown.

4.2.4 SUMMARY

The results in the IPPC domains confirm our experimental findings in the robot manipulation domain: using either E^3 or R-MAX action selection strategies, our exploration approach REX is able to efficiently explore relational domains with fully unknown transition dynamics and different characteristics. All our experiments show that learning relational count functions of known states and actions and exploiting them in a principled exploration strategy outperforms both principled propositional exploration methods as well as the established technique for relational domains, relational ε-greedy.

5. Conclusion

Efficient exploration in relational worlds is an interesting problem that is fundamental to many real-life decision-theoretic planning problems, but has received little attention so far. We have approached this problem by proposing relational exploration strategies that borrow ideas from efficient techniques for propositional representations. The key step in going from propositional to relational representations is a new definition of the concept of the novelty of states and actions. We have introduced a framework of relational count functions to estimate empirical counts of relational data. In general this is a difficult problem since it requires learning from positive data only: the agent wants to generalize the experienced state transitions to other states, but has no negative examples and thus runs the risk of overgeneralization. We have introduced a relational exploration framework called REX where such count functions are learned from experience and drive exploration in relational domains. We have provided guarantees on the exploration efficiency of REX under certain assumptions on the model learner and planner.

We have proposed an instantiation of the REX framework by integrating the relational planner PRADA and a NID rule learner. This combination results in the first relational model-based reinforcement learner with a principled exploration strategy for domains with fully unknown transition dynamics whose effectiveness has been empirically demonstrated on a large number of difficult tasks in complex environments with many objects. Our experimental results show a significant improvement over established non-relational techniques for solving relevant, difficult and highly stochastic planning tasks in a 3D simulated robot environment and in domains of the international planning competition. Our results demonstrate that relational exploration, driven by estimated relational count functions, does not only improve the exploration performance, but also enables the transfer of learned knowledge to new situations and objects and even in a curriculum learning setting where different tasks have to be solved one after the other.

On a more general level, our work shows that it is promising to investigate the combination of statistical machine learning methods with expressive symbolic representations for developing intelligent agents. On the one hand, the symbolic representation provides abstract relational features which are key for generalization. On the other hand, an autonomous agent needs to adapt to its environment from experience and thus statistical techniques are required to actively learn compact symbolic representations.

5.1 Future Work

There are several interesting avenues for future work. One should start to explore statistical relational reasoning and learning techniques for the relational count function estimation problem implicit in exploring relational worlds. Interesting candidates include relational variants of kernel-based methods (Driessens et al., 2006), regression and cluster trees (Blockeel and de Raedt, 1998) as well as boosted relational dependency networks (Neville and Jensen, 2007; Natarajan et al., 2010) and their extension to the online learning setting. We believe our framework opens the door to a large variety of possible exploration strategies—our specific choices have only served as a first proof of concept. Another avenue of future work is to investigate incremental learning of transition models. For instance, how can a set of probabilistic relational rules be modified efficiently with additional experiences? The resulting algorithms might in turn provide new relational exploration strategies. Future work should also explore the connection between relational exploration and transfer learning. As it is hopeless to explore the whole state of a non-trivial world, investigating the relevance of objects or object classes for the task at hand is a further challenge for future research. Finally, examining our approach in other problem scenarios is appealing, for instance, applying it to large-scale applications such as the web or to geometrical reasoning of robots.

Acknowledgments

We thank the anonymous reviewers whose careful and thorough comments have helped greatly to improve the paper. TL, MT and KK were supported by the German Research Foundation (DFG, TO 409/1-3 and SPP 1527-5). KK was supported by the European Commission under contract number FP7-248258-First-MM and the Fraunhofer ATTRACT Fellowship STREAM.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 41–48, 2009.

Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active learning for networked data. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2010.

Hendrik Blockeel and Luc de Raedt. Top-down induction of first order local decision trees. *Artificial Intelligence Journal*, 101:185–297, 1998.

Craig Boutilier, Ray Reiter, and Bob Price. Symbolic dynamic programming for first-order MDPs. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 690–700, 2001.

Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 3:213–231, 2002.

Henrik Christensen. From internet to robotics – a roadmap for US robotics, May 2009. `http://www.us-robotics.us/reports/CCC\%20Report.pdf`.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research (JAIR)*, 4(1):129–145, 1996.

Tom Croonenborghs, Jan Ramon, Hendrik Blockeel, and Maurice Bruynooghe. Online learning and exploiting relational models in reinforcement learning. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 726–731, 2007.

Luc de Raedt, P. Frasconi, Kristian Kersting, and S.H. Muggleton, editors. *Probabilistic Inductive Logic Programming*, volume 4911 of *Lecture Notes in Computer Science*. Springer, 2008.

Carlos Diuk. *An Object-Oriented Representation for Efficient Reinforcement Learning*. PhD thesis, Rutgers, The State University of New Jersey, New Brunswick, NJ, 2010.

Carlos Diuk, Andre Cohen, and Michael Littman. An object-oriented representation for efficient reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2008.

Carlos Diuk, Lihong Li, and Bethany R. Leffler. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2009.

Kurt Driessens and Sašo Džeroski. Integrating guidance into relational reinforcement learning. *Machine Learning Journal*, 57(3):271–304, 2004.

Kurt Driessens, Jan Ramon, and Thomas Gärtner. Graph kernels and Gaussian processes for relational reinforcement learning. *Machine Learning Journal*, 64(1-3):91–119, 2006.

Sašo Džeroski, L. de Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine Learning Journal*, 43:7–52, 2001.

Arkady Epshteyn, Adam Vogel, and Gerald DeJong. Active reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 296–303, 2008.

Lise Getoor and Ben Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

Carlos Guestrin, Relu Patrascu, and Dale Schuurmans. Algorithm-directed exploration for model-based reinforcement learning in factored MDPs. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 235–242, 2002.

Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational MDPs. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1003–1010, 2003.

Florian Halbritter and Peter Geibel. Learning models of relational MDPs using graph kernels. In *Proc. of the Mexican Conf. on AI (MICAI)*, pages 409–419, 2007.

Steffen Hölldobler, Eldar Karabaev, and Olga Skvortsova. FluCaP: a heuristic search planner for first-order MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 27:419–439, 2006.

IPPC. Sixth International Planning Competition, Uncertainty Part, 2008. URL `http://ippc-2008.loria.fr/wiki/index.php/Main\_Page`.

Saket Joshi, Kristian Kersting, and Roni Khardon. Self-taught decision theoretic planning with first order decision diagrams. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2010.

Leslie Pack Kaelbling, Michael Littman, and Andrew Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research (JAIR)*, 4:237–285, 1996.

Sham Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

Sham Kakade, Michael Kearns, and John Langford. Exploration in metric state spaces. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2003.

Michael Kearns and Daphne Koller. Efficient reinforcement learning in factored MDPs. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 740–747, 1999.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning Journal*, 49(2-3):209–232, 2002.

Kristian Kersting and Kurt Driessens. Non–parametric policy gradients: A unified treatment of propositional and relational domains. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2008.

Kristian Kersting, Martijn van Otterlo, and Luc de Raedt. Bellman goes relational. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 465–472, 2004.

J. Zico Kolter and Andrew Ng. Near-Bayesian exploration in polynomial time. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 513–520, 2009.

Tobias Lang. *Planning and Exploration in Stochastic Relational Worlds*. PhD thesis, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 2011.

Tobias Lang and Marc Toussaint. Relevance grounding for planning in relational domains. In *Proc. of the European Conf. on Machine Learning (ECML)*, 2009.

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

Lihong Li. *A Unifying Framework for Computational Reinforcement Learning Theory*. PhD thesis, Department of Computer Science, Rutgers University, New Brunswick, NJ, USA, 2009.

Lihong Li, Michael Littman, Thomas Walsh, and Alexander Strehl. Knows what it knows: a framework for self-aware learning. *Machine Learning Journal*, 82(3):568–575, 2011.

Stephen Muggleton. Learning from positive data. In *Selected Papers from the 6th International Workshop on Inductive Logic Programming*, pages 358–376, London, UK, 1997. Springer-Verlag.

Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude Shavlik. Boosting relational dependency networks. In *Proc. of the Int. Conf. on Inductive Logic Programming (ILP)*, 2010.

Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research (JMLR)*, 8:653–692, 2007.

Shan-Hwei Nienhuys-Cheng and Ronald de Wolf, editors. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Computer Science*. Springer, 1997.

Ali Nouri and Michael L. Littman. Dimension reduction and its application to model-based exploration in continuous spaces. *Machine Learning Journal*, 81:85–98, October 2010.

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

Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 697–704, 2006.

Jan Ramon. *Clustering and Instance-Based Learning in First Order Logic*. PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2002.

Jan Ramon, Kurt Driessens, and T. Croonenborghs. Transfer learning in reinforcement learning problems through partial policy recycling. In *Proc. of the European Conf. on Machine Learning (ECML)*, pages 699–707, 2007.

Scott Sanner. Simultaneous learning of structure and value in relational reinforcement learning. In *Proc. of the ICML-05 Workshop on "Rich Representations for Relational Reinforcement Learning"*, 2005.

Scott Sanner. Online feature discovery in relational reinforcement learning. In *Proc. of the ICML-06 Workshop on "Open Problems in Statistical Relational Learning"*, 2006.

Scott Sanner and Craig Boutilier. Practical solution techniques for first-order MDPs. *Artificial Intelligence Journal*, 173(5-6):748–788, 2009.

Jürgen Schmidhuber. Curious model-building control systems. In *Proc. of Int. Joint Conf. on Neural Networks*, volume 2, pages 1458–1463, 1991.

Alexander L. Strehl and Michael L. Littman. Online linear regression and its application to model-based reinforcement learning. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 737–744, 2007.

Alexander L. Strehl, Lihong Li, and Michael Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research (JMLR)*, 2009.

Sebastian Thrun. The role of exploration in learning control. In *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, 1992.

Marc Toussaint, Nils Plath, Tobias Lang, and Nikolay Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

Thomas J. Walsh. *Efficient Learning of Relational Models for Sequential Decision Making*. PhD thesis, Rutgers, The State University of New Jersey, New Brunswick, NJ, 2010.

Thomas J. Walsh and Michael L. Littman. Efficient learning of action schemas and web-service descriptions. In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI)*, pages 714–719, 2008.

Thomas J. Walsh, Istvan Szita, Carlos Diuk, and Michael L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2009.

Chenggang Wang, Saket Joshi, and Roni Khardon. First order decision diagrams for relational MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 31:431–472, 2008.

David Windridge and Josef Kittler. Perception-action learning as an epistemologically-consistent model for self-updating cognitive representation. *Brain Inspired Cognitive Systems (special issue), Advances in Experimental Medicine and Biology*, 657, 2010.

Zhao Xu, Kristian Kersting, and Thorsten Joachims. Fast active exploration for link–based preference learning using Gaussian processes. In *Proc. of the European Conf. on Machine Learning (ECML)*, 2010.