Relevance Grounding for Planning in Relational Domains

Tobias Lang and Marc Toussaint

TU Berlin, Franklinstrasse 28/29, 10587 Berlin, Germany,
{lang,mtoussai}@cs.tu-berlin.de

Abstract. Probabilistic relational models are an efficient way to learn and represent the dynamics in realistic environments consisting of many objects. Autonomous intelligent agents that ground this representation for all objects need to plan in exponentially large state spaces and large sets of stochastic actions. A key insight for computational efficiency is that successful planning typically involves only a small subset of relevant objects. In this paper, we introduce a probabilistic model to represent planning with subsets of objects and provide a definition of object relevance. Our definition is sufficient to prove consistency between repeated planning in partially grounded models restricted to relevant objects and planning in the fully grounded model. We propose an algorithm that exploits object relevance to plan efficiently in complex domains. Empirical results in a simulated 3D blocksworld with an articulated manipulator and realistic physics prove the effectiveness of our approach.

1 Introduction

Artificial Intelligence investigates systems that act autonomously in complex environments. Such systems need to be able to reason under time pressure about their world in order to derive plans of actions and achieve their goals. This requires probabilistic relational knowledge representations that can deal with the stochasticity of actions, cope with noise and generalize over object instances. Complex environments typically contain very many objects. Consider for example a household robot, that has to represent all kinds of furnitures, dishes, house inventary etc. together with their properties and relationships. Such realistic domains comprise state spaces that are exponential in the number of represented objects and large sets of stochastic actions. Research in A.I. over the last years has led to world models that describe the action effects and state transitions compactly in terms of abstract logical formulae and can be learned from experience. However, how to exploit this model compactness for planning remains a major challenge. Planning in the fully grounded representation is often a hopeless undertaking as the state space quickly grows for all but the smallest problems. This problem is often simply ignored by designing the domain carefully to only contain those domain aspects which are relevant for successful planning. For truly autonomous agents operating continuously with changing tasks, however, we require principled ways to make planning in complex environments tractable.

W. Buntine et al. (Eds.): ECML PKDD 2009, Part I, LNAI 5781, pp. 736–751, 2009.

Models of human cognition provide an inspiring idea of how one may plan in a highly complex world. Humans are often assumed to possess declarative world knowledge about the types of objects they encounter in daily life [1] which is presumably stored in the long-term memory. For instance, they know that piling dishes succeeds the better the more exactly aligned these dishes are. This abstract knowledge is independent of any concrete dish instances or other unrelated objects (such as lamps and cars) and is akin to the abstract probabilistic relational models in A.I.. When planning, human beings may reason about objects according to their abstract world knowledge [3], i.e., they ground their abstract world model with respect to these objects. Such reasoning is often assumed to take place in the working memory, a cognitive system functioning as a work-space in which recently acquired sensory information and information from long-term memory are processed for further action such as decision-making [2] [16]. This system has *limited capacity* and humans can only take some selected objects into account – those they deem relevant for the problem at hand. For example, when planning to prepare a cup of tea, they do not consider the frying pan in the shelf or a soccer ball in the garage. One can view this as grounding the abstract world knowledge only with respect to these relevant objects, thereby enabling tractable planning.

In this paper, we take up this idea and exploit the great advantage of abstract relational world models to be applicable to arbitrary subsets of objects. First, we define object relevance in terms of a graphical model. This allows us then to prove consistency between repeated planning in partially grounded models restricted to relevant objects and planning in the fully grounded model. Thereby, we reformulate the original intractable problem into tractable versions where we can apply any efficient planning method to solve our problem at hand, enabling real-time planning and planning with quickly changing goals. Empirical results in an extended simulated 3D blocksworld with realistic physics and an articulated manipulator using a learned world model show the effectiveness of our approach.

The remainder of this paper is organized as follows. In the next section, we present relational world models and discuss the difficulties of planning in the fully grounded representation. In Section 3, we introduce our approach of *Relevance Grounding*. In Section 4, we present our empirical results. In Section 5, we discuss related work before we conclude.

2 Background

2.1 Compact World Models

A relational domain is represented by a relational logic language $\mathcal{L}$: the set of logical predicates $\mathcal{P}$ and the set of logical functions $\mathcal{F}$ contain the relationships and properties that can hold for domain objects. The set of logical predicates $\mathcal{A}$ comprises the possible actions in the domain.

A concrete instantiation of a relational domain is made up of a finite set of objects $\mathcal{O}$. If the arguments of a predicate or function are all concrete, i.e. taken from $\mathcal{O}$, we call it *grounded*. A concrete world state s is fully described by all

grounded predicates and functions. Concrete actions a are described by positive grounded predicates from $\mathcal{A}$.

The arguments of predicates and functions can also be abstract logical variables which can represent any object. If a predicate or function has only abstract arguments, we call it *abstract*. 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 $\mathcal{O}$.

A relational model $\mathcal{T}$ of the transition dynamics specifies $P(s'|a,s)$, the probability of a successor state s' if action a is performed in state s. $\mathcal{T}$ is usually defined compactly in terms of abstract predicates and functions. This enables abstraction from object identities and concrete domain instantiations. For instance, the effects of trying to grab a cup may be defined by a single abstract model for all concrete cups. To apply $\mathcal{T}$ in a given world state, one needs to ground $\mathcal{T}$ with respect to some of the objects in the domain.

Examples of abstract transition models include relational probability trees for predicates and functions based on abstract logical formulae [6] and probabilistic relational rules, e.g. in the form of STRIPS-operators. An example of the latter are the *noisy indeterministic deictic (NID) rules* [15] which will be our running example in this paper and which we briefly review here. A NID rule r is given as follows

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

where $\mathcal{X}$ is a set of logic variables in the rule (which represent 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 state context Φ_r is fulfilled, and m_r+1 different outcomes with associated probabilities $p_{r,i} > 0$, $\sum_{i=0} p_{r,i} = 1$. Each outcome $\Omega_{r,i}(\mathcal{X})$ describes which predicates and functions 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 $\Omega_{r,i}$. The arguments of the action $a(\mathcal{X}_a)$ may be a true 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.

As above, let σ denote a substitution that maps variables to constant objects, $\sigma : \mathcal{X} \rightarrow \mathcal{O}$. 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 $a = a_r$. By grounding NID rules, we can predict successor states for a given state. NID rules can be learned from experience triples (s,a,s') using a batch algorithm that trades off the likelihood of these triples with the complexity of the learned rule-set. For more details, we refer the reader to Pasula et al. [15].

2.2 Planning in Ground Representations

Our goal is to plan in the ground relational domain: find a "satisficing" action sequence that will lead with high probability to states with large rewards. While a relational transition model $\mathcal{T}$ provides a very compact description of the dynamics of the world, this compactness does not carry over to planning. Grounding the relational representation language $\mathcal{L}$ w.r.t. all domain objects $\mathcal{O}$ results in a state space that is exponential in $|\mathcal{O}|$. Thus, evaluating an action sequence is exponential in $|\mathcal{O}|$. Furthermore, the set of ground actions and thus the search space of plans scales with the number of objects. (Planning is even further complicated due to the stochasticity of actions.) Planning is only tractable in case $|\mathcal{O}|$ is very small. In most realistic scenarios, $|\mathcal{O}|$ is rather large, however. Fortunately, it often suffices to take only the objects that are relevant for the planning problem into account. In the next section, we will introduce *Relevance Grounding* which formalizes this idea in a systematic way. To plan in grounded models, we use the PRADA algorithm [14] in this paper. PRADA converts NID rules into dynamic Bayesian networks, predicts the effects of action sequences on states and rewards by means of approximate inference and samples action sequences in an informed way. PRADA has the crucial advantage to evaluate an action sequence very efficiently, in particular in time linear in its length.

3 Relevance Grounding

In the following, we introduce a probabilistic model which expresses the coupling between state sequences, action sequences, objects and rewards. This model will help to formalize what planning with subsets of objects implies. In particular, we will be able to derive results on planning with subsets of objects – which corresponds to conditioning on object-sets $\mathbf{o}$.

Assume our domain contains objects $\mathcal{O}$. By M we denote the model grounded for all objects, including the complete state and action space (all ground predicates and functions w.r.t. $\mathcal{O}$), which defines the state transition dynamics according to some given relational transition model $\mathcal{T}$. Let $\mathbf{a} = (a_1, \ldots, a_T)$ denote a *plan*, i.e., a sequence of actions. Let $\mathbf{s} = (s_1, \ldots, s_T)$ denote a sequence of encountered states. We assume that in a given trial $(\mathbf{s}, \mathbf{a})$ certain objects are relevant while others are not. For example, an object o is relevant if it is an argument of one of the actions in $\mathbf{a}$. We give a concrete definition of object relevance in Sec. 3.1. For now, we only assume that, in general, object relevance can be expressed by a conditional probability $P(\mathbf{o}|\mathbf{s}, \mathbf{a})$ where $\mathbf{o}$ is a random variable referring to a subset of $\mathcal{O}$. Let R denote the event of achieving a reward at the end of a trial[1]. We assume the following joint distribution over these random variables:

$$P(R, \mathbf{o}, \mathbf{s}, \mathbf{a}; M) = P(R \mid \mathbf{s}, \mathbf{a}; M)\, P(\mathbf{o} \mid \mathbf{s}, \mathbf{a}; M)\, P(\mathbf{s} \mid \mathbf{a}; M)\, P(\mathbf{a}; M) \quad (2)$$

[1] When we assume a geometric prior on the trial length, the expected reward is equivalent to the sum of discounted rewards when rewards are given in each time-step – see Toussaint et al. [18] for details.

Note that all conditional distributions depend on the model M. In absence of goals or rewards, we assume a uniform prior over plans $P(\mathbf{a}; M)$, discounted by their length T by a discount factor $0 < \gamma < 1$ (see footnote 1). In the following, we will eliminate $\mathbf{s}$, i.e., we consider

$$P(R, \mathbf{o}, \mathbf{a}; M) = \sum_{\mathbf{s}} P(R, \mathbf{o}, \mathbf{s}, \mathbf{a}; M) = P(R \mid \mathbf{o}, \mathbf{a}; M) P(\mathbf{o} \mid \mathbf{a}; M) P(\mathbf{a}; M) \,, \tag{3}$$

where R now conditionally depends on $\mathbf{o}$.

Generally, planning requires finding the maximizing argument $\mathbf{a}^*$ of the following distribution:

$$P(\mathbf{a} \mid R; M) \propto P(R \mid \mathbf{a}\,; M)\, P(\mathbf{a}; M)\,. \tag{4}$$

Finding plans with high $P(\mathbf{a} \mid R; M)$ is a difficult task for the following reasons: *(i)* the search space of $\mathbf{a}$ scales with the number of objects; *(ii)* evaluating $P(R \mid \mathbf{a}; M)$ is difficult as M's state space is exponential in the number of objects $\mathcal{O}$. To overcome this problem, we observe that we can decompose

$$P(\mathbf{a} \mid R; M) = \textstyle\sum_{\mathbf{o}} P(\mathbf{a} \mid \mathbf{o}, R; M) P(\mathbf{o} \mid R; M) \tag{5}$$

where $P(\mathbf{o} \mid R; M)$ is defined as

$$P(\mathbf{o} \mid R\,;\, M) \propto \sum_{\mathbf{a}} P(R \mid \mathbf{o}, \mathbf{a}\,;\, M) P(\mathbf{o} \mid \mathbf{a}; M) P(\mathbf{a}; M)\,. \tag{6}$$

$P(\mathbf{o} \mid R; M)$ is a measure for the relevance of object-sets with respect to the reward. Note that this posterior favors small object-sets due to the prior over plan lengths in $P(\mathbf{a}; M)$. If every successful plan makes use of object o, then for each $\mathbf{o}$ with $P(\mathbf{o} \mid R\,;\, M) > 0$ we have $o \in \mathbf{o}$. In this case, we call o necessary for R. Using this formalization of the relevance of object-sets, Eq. (5) provides us a way to decompose the above planning problem into two stages: *(i)* sampling of object-sets using $P(\mathbf{o} \mid R; M)$; *(ii)* finding plans with high $P(\mathbf{a} \mid \mathbf{o}, R; M)$ corresponding to planning conditioned on a set of relevant objects. The key idea is that the conditioning on $\mathbf{o}$ in stage *(ii)* may significantly reduce the cost of planning, as we will discuss below.

3.1 A Sufficient Definition of Relevance

We will now provide a definition of $P(\mathbf{o}|\mathbf{s}, \mathbf{a})$ which we have neglected thus far. For a given pair $(\mathbf{s}, \mathbf{a})$, we define the set Ω of relevant object-sets as

$$\begin{aligned} \Omega(\mathbf{s}, \mathbf{a}) = \{ \mathbf{o} \subseteq \mathcal{O} \mid \forall t, 0 \le t < T : P(s_{t+1} \mid s_t, a_t; M) = P(s_{t+1} \mid s_t, a_t; M_{\mathbf{o}}) \\ \wedge \quad \forall \mathbf{o}' \subset \mathbf{o}, \mathbf{o}' \neq \mathbf{o} \;\; \exists t, 0 \le t < T : P(s_{t+1} \mid s_t, a_t; M) \neq P(s_{t+1} \mid s_t, a_t; M_{\mathbf{o}'}) \} \end{aligned} \tag{7}$$

where $M_{\mathbf{o}}$ is the *reduced model* including only the objects $\mathbf{o}$ with their groundings of predicates and functions. $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t; M_{\mathbf{o}})$ is defined such that all predicates

and functions with at least one argument $o \notin \mathbf{o}$ persist from $\mathbf{s}_t$ and are ignored while calculating transition probabilities, and we define the action prior in the reduced model as $P(\mathbf{a}; M_\mathbf{o}) = P(\mathbf{a}|\mathbf{o}; M)$. $\Omega(\mathbf{s}, \mathbf{a})$ comprises minimal object-sets that are required to predict the state transitions correctly for a specific trial $(\mathbf{s}, \mathbf{a})$. Note that $|\Omega(\mathbf{s}, \mathbf{a})| \geq 1$ for all $(\mathbf{s}, \mathbf{a})$. We define $P(\mathbf{o}|\mathbf{s}, \mathbf{a})$ as

$$P(\mathbf{o} \mid \mathbf{s}, \mathbf{a}) = \frac{I(\mathbf{o} \in \Omega(\mathbf{s}, \mathbf{a}))}{|\Omega(\mathbf{s}, \mathbf{a})|} \,. \tag{8}$$

Intuitively, relevant object-sets are those that are taken into account to calculate the transition probabilities in $\mathbf{s}$ for a given $\mathbf{a}$. Clearly, they include the objects which are manipulated, i.e., whose properties or relationships change. We call these *actively relevant*. There are also *passively relevant* objects which are taken into account by the world dynamics model $\mathcal{T}$. For instance, imagine the task to go to the kitchen and prepare a cup of tea. The tea bag, the cup and the water heater are actively relevant objects. If the kitchen has two doors and one of them is locked, then the latter is passively relevant: we cannot manipulate, i.e. open, it, but it plays a role in planning as its being locked determines the other door to be necessary. A more technical example of object relevance is given below.

In general, there might be alternative interesting definitions of object relevance, e.g. where the transition probabilities in Eq.(7) only hold approximately. We chose the above definition because it is sufficient to a certain consistency for planning in reduced models:

Lemma 1. *When conditioning on a subset* $\mathbf{o}$ *of relevant objects, the following probabilities in the reduced model* $M_\mathbf{o}$ *are the same as in the full model* M*:*
(a) State sequences: $P(\mathbf{s} \mid \mathbf{o}, \mathbf{a}; M) = P(\mathbf{s} \mid \mathbf{a}; M_\mathbf{o})$
(b) Rewards: $P(R \mid \mathbf{o}, \mathbf{a}; M) = P(R \mid \mathbf{a}; M_\mathbf{o})$
(c) Action sequences: $P(\mathbf{a} \mid \mathbf{o}, R; M) = P(\mathbf{a} \mid R; M_\mathbf{o})$

Proof. If $\mathbf{o} \in \Omega(\mathbf{s}, \mathbf{a})$, we have:

$$P(\mathbf{s}|\mathbf{o}, \mathbf{a}; M) = \prod_{t=0}^{T-1} P(s_{t+1}|\mathbf{o}, s_t, a_t; M) = \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t; M_\mathbf{o}) = P(\mathbf{s}|\mathbf{a}; M_\mathbf{o}) \,. \tag{9}$$

If $\mathbf{o} \notin \Omega(\mathbf{s}, \mathbf{a})$, we have $P(\mathbf{s} \mid \mathbf{o}, \mathbf{a}; M) = 0$. Similarly, $\mathbf{s}$ cannot be predicted in $M_\mathbf{o}$ as only the irrelevant object-set $\mathbf{o}$ is available, so we get $P(\mathbf{s} \mid \mathbf{a}; M_\mathbf{o}) = 0$. Furthermore, we have:

$$P(R \mid \mathbf{o}, \mathbf{a}; M) = \sum_{\mathbf{s}} P(R, \mathbf{s} \mid \mathbf{o}, \mathbf{a}; M) = \sum_{\mathbf{s}} P(R|\mathbf{s}, \mathbf{o}, \mathbf{a}; M) P(\mathbf{s}|\mathbf{o}, \mathbf{a}; M) \tag{10}$$

$$= \sum_{\mathbf{s}} P(R \mid \mathbf{s}, \mathbf{a}; M)\, P(\mathbf{s} \mid \mathbf{a}; M_\mathbf{o})\, I(\mathbf{o} \in \Omega(\mathbf{s}, \mathbf{a})) \tag{11}$$

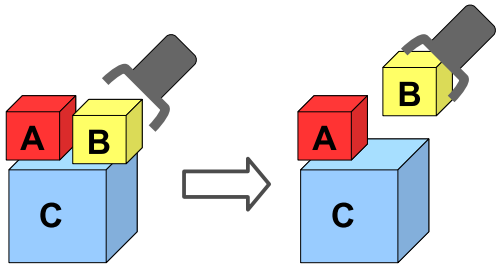
$$= \sum_{\mathbf{s}: \mathbf{o} \in \Omega(\mathbf{s}, \mathbf{a})} P(R \mid \mathbf{s}, \mathbf{a}; M) P(\mathbf{s} \mid \mathbf{a}; M_\mathbf{o}) = P(R \mid \mathbf{a}; M_\mathbf{o}) \tag{12}$$

Finally, we have:

$$P(\mathbf{a} \mid \mathbf{o}, R; M) \propto P(R \mid \mathbf{o}, \mathbf{a}; M)\; P(\mathbf{a} \mid \mathbf{o}; M) \tag{13}$$

$$= P(R \mid \mathbf{a}; M_\mathbf{o})\; P(\mathbf{a}; M_\mathbf{o}) \quad \propto \quad P(\mathbf{a} \mid R; M_\mathbf{o}) \quad \square \tag{14}$$

Table 1. Example of active and passive object relevance. Objects B and C are actively relevant as their properties are changed. A is passively relevant as it determines rule 2 to model the transition dynamics. If it was ignored, rule 1 would be used instead yielding wrong state transition probabilities.



Action: $grab(B)$

Rule 1:
$grab(X): \quad on(X,Y), \neg on(Z,Y)$
$\rightarrow 1.0 : inhand(X),\ \neg on(X,Y)$

Rule 2:
$grab(X): \quad on(X,Y), on(Z,Y)$
$\rightarrow 0.8 : inhand(X),\ \neg on(X,Y)$
$\quad 0.2 : \neg on(X,Y)$

From Lemma 1 and Eq. (5) the following proposition follows directly:

Proposition 1. *Given the joint in Eq. (2) and the definition of* $P(\mathbf{o} \mid \mathbf{s}, \mathbf{a})$ *in Eq. (8), it holds:*

$$P(\mathbf{a} \mid R; M) = \sum_{\mathbf{o}} P(\mathbf{a} \mid R; M_{\mathbf{o}})\, P(\mathbf{o} \mid R; M) \tag{15}$$

An illustrative example of object relevance. The scenario in Table 1 illustrates active and passive object relevance. Two small blocks A and B are on top of a big block C. Our goal is to hold B inhand. This can be achieved by means of a plan consisting of a single action $grab(B)$. Our transition dynamics model contains two NID rules to model the *grab*-action. Rule 1 applies if the target block is the only block on top, in which case *grab* always succeeds. Rule 2 applies if the target block is not the only block on top. In this case, *grab* only succeeds with probability 0.8; otherwise with probability 0.2, grabbing fails due to lack of space and the target block is pushed off the big block instead. Clearly, in our situation we have to use rule 2. Blocks B and C are manipulated and thus are actively relevant, whereas A is passively relevant as it determines rule 2 to apply. If A was ignored, we would use rule 1 – yielding an erroneous higher success probability. Similar scenarios are typical in physical worlds: the probabilities of successful planning change when objects (e.g. potential obstacles) are added to or removed from the scene even when they are not actively manipulated.

3.2 Planning with Relevant Objects

Our definition of object relevance and the subsequent discussion led to a crucial observation: to find plans with high $P(\mathbf{a} \mid R; M)$, it is not necessary to use the full model M including all objects $\mathcal{O}$. As Proposition 1 shows, an alternative is to find plans in the reduced models $P(\mathbf{a} \mid R; M_{\mathbf{o}})$ for object-sets $\mathbf{o}$ with high

Algorithm 1. Relevance Grounding

Input: objects $\mathcal{O}$, goal τ, relational transition model $\mathcal{T}$, relevance distribution $q(\cdot)$, number of relevance groundings N_{rel}, number of verifications N_{ver}
Output: action sequence $\mathbf{a}$

for $i = 1$ to N_{rel} **do** ▷ Relevance grounding
  Sample object-set $\mathbf{o}_i \subset \mathcal{O}$ according to q
  Build reduced model $M_{\mathbf{o}_i}$
  $\mathbf{a}_i = plan(\tau; M_{\mathbf{o}_i}, \mathcal{T})$ ▷ Plan in reduced model
  $\psi(\mathbf{a}_i) = P(R \mid \mathbf{a}_i; M_{\mathbf{o}_i}, \mathcal{T})$ ▷ Value in reduced model
end for
for $i = 1$ to N_{ver} **do** ▷ Verifying in original model
  Let $\mathbf{a}$ denote plan with i-th largest ψ
  Calculate $\Psi(\mathbf{a}) = P(R \mid \mathbf{a}; M, \mathcal{T})$ ▷ Value in original model
end for
return $\text{argmax}_{\mathbf{a}} \Psi(\mathbf{a})$

relevance $P(\mathbf{o} \mid R; M)$. This makes planning more efficient due to the reduced state and action spaces in $M_{\mathbf{o}}$.

Obviously, we do not know $P(\mathbf{o} \mid R; M)$. If we knew the reward likelihoods for all plans, i.e., if we had already planned, we could calculate this quantity according to Eq. (6). However, planning is just the problem we are trying to solve. Thus, we have to estimate this quantity by some distribution $q(\cdot)$ over object-sets resulting in the approximate distribution $Q(\cdot)$ over plans defined as

$$Q(\mathbf{a}; R, M) = \sum_{\mathbf{o}} P(\mathbf{a} \mid R; M_{\mathbf{o}})\, q(\mathbf{o}; R, M) \approx P(\mathbf{a} \mid R; M)\,. \tag{16}$$

The quality of $Q(\cdot)$ depends on the quality of the approximate relevance distribution $q(\cdot)$. If $q(\cdot)$ is not exact, then a plan found in $M_{\mathbf{o}}$ may have lower success probability when planning in M instead (cf. Table 1). Therefore, it is a good idea to verify the quality of the proposed plan in the original model M or in a less reduced model $M_{\mathbf{o}'}$ with $\mathbf{o} \subset \mathbf{o}'$. This requires algorithms that can exploit the transition dynamics $\mathcal{T}$ to efficiently calculate $P(R \,|\mathbf{a})$ also in large models.

Algorithm 1 presents our complete *Relevance Grounding* method. Given an estimator for object-set relevance $q(\cdot)$, we can find plans with approximately high $P(\mathbf{a} \mid R; M)$ as follows: *(i)* we take samples $\mathbf{o}$ from $q(\cdot)$; *(ii)* we plan in the reduced models $M_{\mathbf{o}}$; *(iii)* we verify the resulting plans in the original or a less reduced model; *(iv)* we return the plan with the best verified value. In this paper, we employ NID rules as transition dynamics model $\mathcal{T}$ and use the PRADA algorithm for planning which is in particular appropriate for verification as it evaluates an action sequence in time linear in its length.

3.3 Learning Object Relevance

A crucial part in our proposed method is the relevance estimator of object-sets. Learning such an estimator is a novel and interesting machine learning problem.

As we are using relational representations, we can generalize over object identities in our planning goals and transfer the knowledge gained in previous planning trials to new, but similar problems. For a given goal τ, we can use our world dynamics model to create training instances $(\sigma_0, \tau, \mathbf{o}, P(\mathbf{o} \mid R))$. This enables us to learn object relevance based on nothing more than internal simulation (in contrast to "real" experiences) – akin to human reflection about a problem. σ_0 is a description of the start state s_0 and may involve all types of information, such as discrete, relational and continuous features. We can employ any regressor that can make use of the chosen features to learn a function $q(\mathbf{o}; s_0, \tau) \to \mathbb{R}$. A full approach to learning object relevance is beyond the scope of this paper, but in our first experiment (cf. Section 4.1) we will present an example of how to learn object relevance in a straightforward way, based purely on internal simulation.

4 Experiments

We test our *Relevance Grounding* approach in an extended simulated blocks world where a robot manipulates blocks and also balls scattered on a table. We use a 3D rigid-body dynamics simulator (ODE) that enables a realistic behavior of the 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). Object classes 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 and try to put them on top of other objects or on the table. Its actions are affected by noise so that resulting object piles are not straight-aligned. We 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 the data with predicates $block(X)$, $ball(X)$, $table(X)$, $on(X, Y)$, $out(X)$, $inhand(X)$, $upright(X)$, $clear(X) \equiv \forall Y.\neg on(Y, X)$ and functions $size(X), color(X)$ for state descriptions and $puton(X)$, $grab(X)$ and $doNothing()$ for actions. If there are o objects and f different object sizes and colors, the action space contains $2o+1$ actions while the state space is huge with $f^{2o}2^{o^2+6o}$ different states (not excluding states one would classify as impossible given some intuition about real world physics).

We use NID rules described in Sec. 2.1 to model the state transitions. We employ the rule learning algorithm of Pasula et al. [15] with the same parameter settings to learn three different sets of fully abstract NID rules from independent training sets of 500 experience triples each. Training data to learn rules are generated in a world of ten objects (six blocks, four balls) of two different sizes by performing random actions with a slight bias to build high piles. The resulting rule-sets contain 11, 12 and 12 rules respectively. We use the PRADA algorithm [14] for planning. We test our approach in worlds with varying numbers of blocks and balls of two different sizes. Thus, we transfer the knowledge gained in the training world to different, but similar worlds by using abstract NID rules. In each experiment, for each object number we create five start situations with different objects. Per rule-set and start situation, we perform three independent

runs with different random seeds. In all scenarios, we make the assumption that relevant object-sets contain 5 objects. We investigate different estimators to determine these 5 objects. To evaluate each approach, we compute the planning times and the mean performance over the fixed (but randomly generated) set of 45 test scenarios (3 *learned* rule-sets, 5 situations, 3 seeds).

4.1 Building High Piles

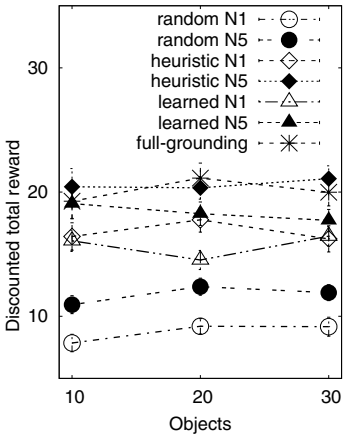
In our first experiment, we repeat the experiment of Pasula et al. which investigates building high piles. Our starting situations are chosen such that all objects have height 0 (are on the table) and our reward is the total change in object heights. We let the algorithm run for 10 time-steps. We set PRADA's planning horizon to $d = 6$ and use a discount factor of $\gamma = 0.95$. If the world was deterministic and objects could be stacked perfectly (such that objects could also be stacked on balls), the optimal discounted total reward would be 37.04.

We investigate three different relevance estimators to determine the sets of relevant objects. The *random* estimator samples objects randomly and independently. The *hand-made heuristic* assigns high probability to big blocks (since these are best to build with) and to objects that are either part of a high pile or on the table (in order to build higher piles). Once it has sampled an object, it assigns high probability to objects within the same pile as these might be required for deictic referencing in the NID rules (passive object relevance).

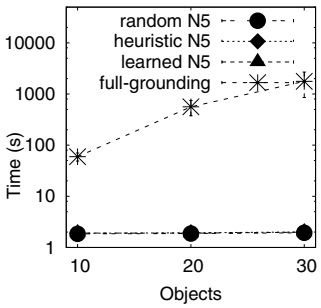
Furthermore, we investigate a simple *learned* estimator of object relevance from which we sample objects independently. We use linear regression to learn from discrete and logical object features, namely object size, type, color, height and clearedness. Training data are generated solely by internal simulation with the PRADA algorithm (in contrast to using the "real" ODE simulator) as follows: for a given situation, we randomly sample 5 objects and derive a plan in the partially grounded network; this plan is then evaluated in the full network and the resulting value is used as relevance estimate for these 5 objects. Note that this procedure does not require real experiences as it is fully based on internal reasoning about which features make an object relevant according to the learned world model (the NID rules in our case). The resulting learned estimator ignores object color as expected, but takes all other features into account, favoring clear big blocks at high heights. We compare these three relevance estimators to the *full-grounding* baseline which plans in the fully grounded model.

Table 2 presents our results. The mean performance of the heuristic is comparable to the full-grounding baseline. The performance of the learned estimator is comparable or only slightly worse than the heuristic, depending on the number of objects and the number N_{rel} of partially grounded models, but always significantly better than the random estimator. The performance of all estimators improves with increasing N_{rel}, but this effect diminishes if N_{rel} is large. Planning in the fully grounded model is hopelessly inefficient, in particular for large worlds. The same performance levels can be achieved by means of Relevance Grounding in only tiny fractions of this planning time, which is independent of the object number and thus constant over all investigated domain sizes.

Table 2. *Building high piles problem*: (a) Mean rewards (changes in tower heights), (b) planning times, (c) details over 45 runs (3 rule-sets, 5 start situations, 3 seeds). Error bars for the rewards give the std. dev. of the mean estimator. N_{rel} denotes the number of relevant reduced models (cf. Algorithm 1). Performing no actions gives a reward of 0.



(a)



(b)

Obj.	Config	Reward	Time
10	random $N_{rel}=1$	7.85 ± 0.70	**0.37 ± 0.03**
	random $N_{rel}=5$	10.94 ± 0.73	1.84 ± 0.18
	random $N_{rel}=10$	14.59 ± 1.20	3.63 ± 0.26
	heuristic $N_{rel}=1$	16.42 ± 1.10	0.38 ± 0.02
	heuristic $N_{rel}=5$	20.43 ± 1.47	1.89 ± 0.11
	heuristic $N_{rel}=10$	**20.83 ± 1.32**	3.78 ± 0.23
	learned $N_{rel}=1$	16.07 ± 0.85	0.39 ± 0.02
	learned $N_{rel}=5$	19.07 ± 1.23	1.91 ± 0.11
	learned $N_{rel}=10$	18.63 ± 1.12	3.76 ± 0.29
	full-grounding	19.26 ± 1.38	59.51 ± 10.72
20	random $N_{rel}=1$	9.20 ± 0.62	0.39 ± 0.03
	random $N_{rel}=5$	12.38 ± 0.69	1.87 ± 0.15
	random $N_{rel}=10$	14.93 ± 0.74	3.78 ± 0.21
	heuristic $N_{rel}=1$	17.77 ± 0.99	**0.39 ± 0.02**
	heuristic $N_{rel}=5$	20.34 ± 0.91	1.93 ± 0.11
	heuristic $N_{rel}=10$	20.69 ± 1.16	3.85 ± 0.15
	learned $N_{rel}=1$	14.53 ± 0.77	0.42 ± 0.03
	learned $N_{rel}=5$	18.26 ± 0.94	1.90 ± 0.13
	learned $N_{rel}=10$	18.34 ± 0.82	3.81 ± 0.11
	full-grounding	**21.12 ± 1.21**	561.78 ± 186.76
30	random $N_{rel}=1$	9.16 ± 0.76	**0.38 ± 0.03**
	random $N_{rel}=5$	11.90 ± 0.64	1.93 ± 0.12
	random $N_{rel}=10$	14.00 ± 0.69	3.84 ± 0.25
	heuristic $N_{rel}=1$	16.23 ± 1.05	0.39 ± 0.02
	heuristic $N_{rel}=5$	**21.08 ± 1.03**	2.01 ± 0.13
	heuristic $N_{rel}=10$	20.21 ± 1.10	3.84 ± 0.16
	learned $N_{rel}=1$	16.45 ± 0.77	0.42 ± 0.04
	learned $N_{rel}=5$	17.72 ± 0.88	1.99 ± 0.04
	learned $N_{rel}=10$	18.44 ± 0.75	3.78 ± 0.24
	full-grounding	19.99 ± 1.11	1770.55 ± 916.44

(c)

4.2 Desktop Clearance

The goal in our second experiment is to clear up the desktop (see Fig. 1). Objects are lying splattered all over the desktop. An object is cleared if it is part of a pile containing all other objects of the same class, which can be defined as

$$cleared(X) \equiv \forall Y : sameClass(X,Y) \rightarrow samePile(X,Y)\ . \tag{17}$$

A class is defined in terms of color and size, but not type so that a class contains both blocks and balls. In our experiments, classes are made up of 2-4 objects with at most 1 ball (in order to enable successful piling). Our starting situations contain some piles, but only with objects of different classes. We let the algorithm run for 30 time-steps. For planning, we set PRADA's planning horizon

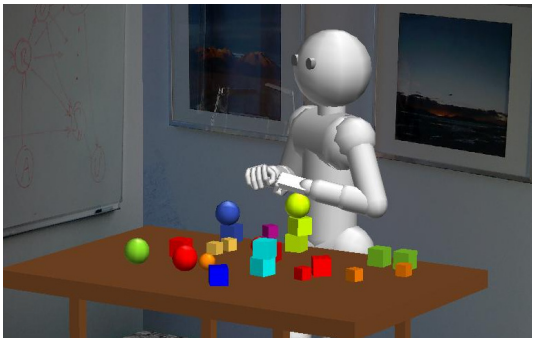


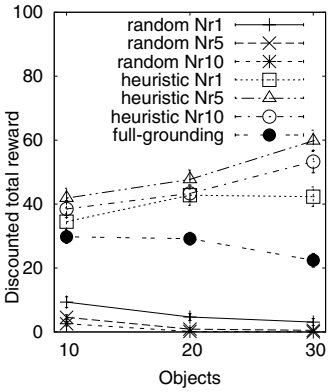
Fig. 1. *Clearance task.* The robot has to clear up the desktop by piling objects of the same size and color.

to $d = 20$ and use a discount factor of $\gamma = 0.95$. If the world was deterministic and objects could be stacked perfectly, the optimal values would be 86.91 for worlds with 10 objects and 110.85 for worlds with 20 and 30 objects. We investigate two relevance estimators. The random estimator samples randomly and independently among all objects. The heuristic estimator chooses randomly among the objects which are not cleared yet and then takes all other objects of the same class into account. Nearest neighbors are used to fill up the object-set. While this heuristic is hand-made, its idea can be derived from the logical reward description in Eq. (17) which states the importance of classes in relevant object-sets on the left side of the implication. How this can be done in principled ways is a major direction of future work.

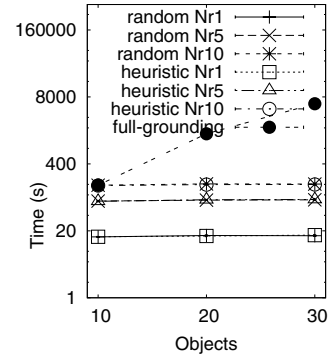
Table 3 presents our results. The random estimator performs poorly since its reduced models contain mostly only single instances of a class. This is disadvantageous as planning requires at least a second object of the same class and singleton instances are always cleared within reduced models which are thus a bad approximation of the full model. The mean performance of the heuristic estimator is significantly better than the full-grounding baseline, in particular in worlds with many objects. Note that the full-grounding baseline cannot find an optimal solution due to the huge search space. In contrast to planning in the fully ground model, the relevance grounding planning approaches are independent of the number of objects and thus several orders of magnitude faster.

We also investigate the use of verification of the plans found in the reduced models (cf. Algorithm 1). We evaluate the $N_{ver} = 3$ best reduced-model plans in a less reduced model containing 10 objects where the missing 5 slots are filled in by nearest neighbors. Thereby, information about objects within the same piles may be taken into account. Our results show that this improves the mean performance for both relevance estimators significantly at only a small increase in computational cost. In particular, this greatly increases the performance of the random estimator in worlds with 10 objects. In larger worlds, the random estimator almost never finds good plans in which case verification cannot help.

Table 3. *Clearance problem*: (a) Mean rewards, (b) planning times, (c) details over 45 runs (3 rule-sets, 5 start situations, 3 seeds). Error bars for the rewards give the std. dev. of the mean estimator. N_r denotes the number of relevant reduced models N_{rel}, N_v the number of partial plans N_{ver} that are verified in a less reduced model (cf. Algorithm 1). Performing no actions gives a reward of 0.



(a)



(b)

Obj.	Config	Reward	Time
10	random N_r=1	9.33 ± 1.67	**15.26 ± 0.21**
	rand. N_r=5	4.59 ± 0.78	75.73 ± 1.53
	rand. N_r=10	2.58 ± 0.73	153.86 ± 3.37
	rand. N_r=5, N_v=3	17.54 ± 1.92	84.63 ± 2.42
	rand. N_r=10, N_v=3	14.76 ± 1.85	162.25 ± 7.60
	heuristic N_r=1	34.50 ± 2.44	15.31 ± 0.27
	heur. N_r=5	41.88 ± 3.08	75.02 ± 1.64
	heur. N_r=10	38.48 ± 2.67	151.92 ± 2.94
	heur. N_r=5, N_v=3	46.46 ± 3.12	84.98 ± 1.83
	heur. N_r=10, N_{vr}=3	**49.15 ± 2.81**	161.79 ± 3.63
	full-grounding	29.77 ± 2.02	153.93 ± 13.51
20	random N_r=1	4.68 ± 1.07	16.24 ± 0.44
	rand. N_r=5	0.88 ± 0.45	78.92 ± 1.08
	rand. N_r=10	0.16 ± 0.11	163.21 ± 4.93
	rand. N_r=5, N_v=3	2.62 ± 0.90	90.25 ± 1.74
	rand. N_r=10, N_v=3	0.72 ± 0.58	168.91 ± 3.31
	heuristic N_r=1	42.77 ± 3.19	**15.89 ± 0.48**
	heur. N_r=5	47.72 ± 2.96	80.24 ± 1.38
	heur. N_r=10	43.34 ± 2.60	158.53 ± 2.06
	heur. N_r=5, N_v=3	51.03 ± 3.12	88.94 ± 1.99
	heur. N_r=10, N_v=3	**56.76 ± 2.68**	172.27 ± 5.01
	full-grounding	29.19 ± 1.98	1537.37 ± 225.96
30	random N_r=1	3.09 ± 1.09	**16.06 ± 0.41**
	rand. N_r=5	0.52 ± 0.26	80.11 ± 1.47
	rand. N_r=10	0.02 ± 0.02	162.17 ± 6.34
	rand. N_r=5, N_v=3	1.16 ± 0.51	92.17 ± 2.52
	rand. N_r=10, N_v=3	0.16 ± 0.10	178.96 ± 3.56
	heuristic N_r=1	42.31 ± 3.06	16.47 ± 0.43
	heur. N_r=5	59.79 ± 3.33	81.34 ± 3.87
	heur. N_r=10	53.29 ± 3.50	159.01 ± 3.39
	heur. N_r=5, N_v=3	55.00 ± 3.54	90.26 ± 3.46
	heur. N_r=10, N_v=3	**58.01 ± 3.42**	168.51 ± 5.16
	full-grounding	22.42 ± 2.14	5893.81 ± 1006.56

(c)

5 Related Work

The problem of planning in stochastic relational domains has been approached in quite different ways. The field of Relational Reinforcement Learning (RRL) [19] investigates value functions and Q-functions that are defined over all possible ground states and actions of a relational domain. The idea is to describe important world features in terms of abstract logical formulas enabling generalization over objects and situations. Examples of model-free approaches employ relational regression trees [8] or instance-based regression using distance metrices between relational states such as graph kernels [7] to learn Q-functions. Model-free approaches have the disadvantage to be inflexible as they enable planning only for the specific problem type used in the training examples. In contrast, model-based RRL approaches first learn a relational world model from the state

transition experiences, for example in form of relational probability trees for individual state properties [6] or SVMs using graph kernels [12]. One way to make use of the resulting model is to sample look-ahead trees of state transitions in Q-learning, i.e., to work with ground states. All approaches discussed thus far make use of ground states and actions and may well profit from our relevance grounding approach. Consider for example the instance-based approaches where relevance grounding will lead to tractable instance representations.

A promising alternative is to compute abstract value functions by working in the "lifted" abstract representation without grounding or referring to particular problem instances. This requires the learned (or prespecified) model to be complete. Symbolic Dynamic Programming [5] investigates exact solution methods for relational MDPs. The idea is to construct minimal logical partitions of the state space required to make all necessary value function distinctions. For example, Kersting et al. [13] present an exact value iteration for relational MDPs. Sanner et al. [17] exploit factored transition models of first-order MPDs to approximate the value function based on linear combinations of abstract first-order value functions. Their work shows that under certain assumptions (such as additive rewards), it is possible to derive efficient solution techniques. Nonetheless, this promising line of research is only in its beginnings and is confronted with serious technical challenges. It requires complex theorem proving to keep the logical formulas that represent sets of underlying states manageable.

All of the above approaches compute full policies over complete state and action spaces. Instead, one may restrict oneself to deriving plans for a given start state. When grounding the full model, one might in principle use any of the traditional A.I. planning methods used for propositional representations, see [20] and [4]. An interesting strategy to work in a grounded model in a principled way is to consider only a small relevant subset of the state space which is derived from the start state and the planning goal. In contrast to our approach, the resulting subspace still represents all objects, thus the action space size is not decreased. A straight-forward way to create such a subspace are look-ahead trees for the start state that estimate the value of an action by taking samples of the corresponding successor state distribution [15]. Another idea is to maintain an *envelope* of states, a high-utility subset of the state space [9] which can be used to define a relational MDP. This envelope can be further refined by incorporating nearby states in order to improve planning quality. A crucial part of this approach is the initialization of the envelope which is based on an initial straight-line path from the start state to a goal state using a heuristic forward planner (e.g., by making this planning problem deterministic by only considering the most-probable successor state of an action). The envelope-based approach depends strongly on the efficiency and quality of this initial planner which is still faced with the complexity of the action space and its dependence on the number of objects, thus being applicable only for rather small planning horizons.

Action space complexity can be decreased by noting that if the identities of objects do not matter but only their relationships, then different equivalent actions may lead to equivalent successor states [10]. These are states where the

same relationships hold, but not necessarily with the same objects. Relevance grounding accounts for this idea by defining different object subsets to be relevant for the planning problem at hand. Action equivalence can be exploited during planning by only considering one sampled action per action equivalence class which significantly reduces the search space. If identity matters for a large number of objects, however, then this approach does not yield significant improvements. Another way to reduce the state space complexity is to look only at a subset of the logical vocabulary, i.e., ignore certain predicates and functions [11]. This helps when combined with the action equivalence approach as state descriptions become shorter and more approximate and the number of state equivalences increases. All these methods just discussed are complementary to our approach and when applied in a reduced grounded model within the relevance grounding framework might yield a strong way to plan efficiently in highly complex domains.

6 Discussion and Conclusions

In this paper, we have presented an approach for efficient planning in stochastic relational worlds based on exploiting object relevance. We define object relevance in terms of a graphical model. We have derived a systematic framework to plan in partially grounded models which we have proven to be consistent with planning in the fully grounded model. Empirical results show our approach to be effective in complex relational environments. Also, we have argued that our approach has interesting analogies to human cognition. Our framework is independent of the concrete planning algorithm used within the reduced models. In particular, it can well be combined with other approaches to increase planning efficiency in stochastic relational domains that have recently been introduced, such as envelope-based methods [9].

A key part for our framework and our major direction of future research is the estimator of object relevance. We have provided a successful example of how relevance can be learned from object features by means of nothing more than internal simulation, but this is clearly only preliminary. In our point of view, learning to estimate the relevance of objects is a formidable problem for machine learning, as a huge variety of methods using discrete, continuous, and logical features can be applied. Clearly, this is a difficult problem, bearing in mind that human beings often take a long time until they master certain types of planning problems. However, estimating object relevance appears to us to be a crucial prerequisite to be able to plan in the highly complex real world.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by the German Research Foundation (DFG), Emmy Noether fellowship TO 409/1-3.

References

1. Anderson, J.: Rules of the mind. Lawrence Erlbaum, Hillsdale (1993)
2. Baddeley, A.: The episodic buffer: a new component of working memory? Trends in Cognitive Sciences 4(11), 417–423 (1999)
3. Botvinick, M.M., An, J.: Goal-directed decision making in prefrontal cortex: a computational framework. In: Advances in Neural Information Processing Systems, NIPS (2009)
4. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Artificial Intelligence Research 11, 1–94 (1999)
5. Boutilier, C., Reiter, R., Price, B.: Symbolic dynamic programming for first-order MDPs. In: IJCAI, pp. 690–700. Morgan Kaufmann, San Francisco (2001)
6. Croonenborghs, T., Ramon, J., Blockeel, H., Bruynooghe, M.: Online learning and exploiting relational models in reinforcement learning. In: Proc. of the Int. Conf. on Artificial Intelligence, IJCAI (2007)
7. Driessens, K., Ramon, J., Gärtner, T.: Graph kernels and Gaussian processes for relational reinforcement learning. Machine Learning (2006)
8. Dzeroski, S., de Raedt, L., Driessens, K.: Relational reinforcement learning. Machine Learning 43, 7–52 (2001)
9. Gardiol, N.H., Kaelbling, L.P.: Envelope-based planning in relational MDPs. In: Proc. of the Conf. on Neural Information Processing Systems, NIPS (2003)
10. Gardiol, N.H., Kaelbling, L.P.: Action-space partitioning for planning. In: Proc. of the National Conference on Artificial Intelligence (AAAI), Vancouver, Canada (2007)
11. Gardiol, N.H., Kaelbling, L.P.: Adaptive envelope MDPs for relational equivalence-based planning. Technical Report MIT-CSAIL-TR-2008-050, MIT CS & AI Lab, Cambridge, MA (2008)
12. Halbritter, F., Geibel, P.: Learning models of relational MDPs using graph kernels. In: Proc. of the Mexican Conference on Artificial Intelligence (2007)
13. Kersting, K., Otterlo, M.V., de Raedt, L.: Bellman goes relational. In: Proc. of the Int. Conf. on Machine Learning, ICML (2004)
14. Lang, T., Toussaint, M.: Approximate inference for planning in stochastic relational worlds. In: Proc. of the Int. Conf. on Machine Learning, ICML (2009)
15. Pasula, H.M., Zettlemoyer, L.S., Kaelbling, L.P.: Learning symbolic models of stochastic domains. Artificial Intelligence Research 29 (2007)
16. Ruchkin, D.S., Grafman, J., Cameron, K., Berndt, R.S.: Working memory retention systems: a state of activated long-term memory. Behavioral and Brain Sciences 26, 709–777 (2003)
17. Sanner, S., Boutilier, C.: Approximate solution techniques for factored first-order MDPs. In: Proc. of the Int. Conf. on Automated Planning and Scheduling, ICAPS (2007)
18. Toussaint, M., Storkey, A.: Probabilistic inference for solving discrete and continuous state Markov decision processes. In: Proc. of the Int. Conf. on Machine Learning, ICML (2006)
19. van Otterlo, M.: The Logic of Adaptive Behavior. IOS Press, Amsterdam (2009)
20. Weld, D.S.: Recent advances in AI planning. AI Magazine 20(2), 93–123 (1999)