Logical Markov Decision Programs

Kristian Kersting and **Luc De Raedt**
Machine Learning Lab,
University of Freiburg,
Georges-Koehler-Allee, Building 079,
79110 Freiburg, Germany

Abstract

Motivated by the interest in relational reinforcement learning, we introduce a novel representation formalism, called logical Markov decision programs (LOMDPs), that integrates Markov Decision Processes with Logic Programs. Using LOMDPs one can compactly and declaratively represent complex relational Markov decision processes. Within this framework we then develop a theory of reinforcement learning in which abstraction (of states and actions) plays a major role. The framework presented should provide a basis for further developments in relational reinforcement learning.

1 Introduction

In the past few years, there has been a lot of work on extending probabilistic and stochastic frameworks with abilities to handle objects, see e.g. [Anderson *et al.*, 2002; Džeroski *et al.*, 2001; Friedman *et al.*, 1999; Kersting and De Raedt, 2001; Kersting *et al.*, 2003; Muggleton, 1996]. From an inductive logic programming or relational learning point of view, these approaches are upgrades of propositional representations towards the use of relational or computational logic representations. Various successes in this direction have been reported. Indeed, [Friedman *et al.*, 1999] and [Kersting and De Raedt, 2001] upgrade Bayesian networks, [Muggleton, 1996] upgrades stochastic context free grammars, [Anderson *et al.*, 2002] and [Kersting *et al.*, 2003] upgrade (hidden) Markov models.

The first contribution of this paper is the introduction of a novel representation formalism, called *logical Markov decision programs* (LOMDPs), that combines *Markov decision processes* with computational logic. The result is a flexible and expressive framework for defining MDPs that are able to handle structured objects as well as relations and functors. For MDPs, such a framework grounded in computational logic, was still missing. Only [Boutilier *et al.*, 2001] report on combining MDPs with Reiter's situation calculus. However, as we argue in Section 7, it is more complex and model-free reinforcement learning techniques have yet not been addressed within this framework. LOMDPs share - with the other upgrades of propositional representations - two advantages. First, logical expressions (in the form of clauses, rules or transitions) may contain variables and as such make *abstraction* of many specific *grounded* rules or transitions. This allows one to compactly represent complex domains. Secondly, because of this abstraction, the number of parameters (such as rewards and probabilities) in the model is significantly reduced. This in turn allows - in principle - to speed up and simplify the learning because one can learn at the *abstract* level rather than at the *ground* level.

Many fascinating machine learning techniques have been developed under the name reinforcement learning (RL) in the context of MDPs over the last few decades, cf. [Sutton and Barto, 1998]. Recently, there has also been an increased attention for dealing with relational representations and objects in reinforcement learning, see e.g. [Džeroski *et al.*, 2001; Finney *et al.*, 2002] Many of these works have taken a practical perspective and have developed systems and experiments that operate in relational worlds. At the heart of these systems there is often a function approximator (often a logical decision tree) that is able to assign values to sets of states and to sets of state–action pairs. So far, however, a theory that explains why this approach works seems to be lacking. The second and most important contribution of this paper is a first step into the direction of such a theory. The theory is based on a notion of abstract states and abstract policies represented by logical expressions. An abstract state represents a set of concrete states and an abstract policy is then a function from abstract states to actions. All ground states represented by the same abstract state are essentially assigned the same action. This is akin to what happens with (relational) reinforcement learning using (logical) decision trees [Džeroski *et al.*, 2001], where each leaf of the decision tree represents an abstract state and where states classified in the same leaf obtain the same value or action. Within the LOMDP framework abstract policies can easily be represented. The abstract value function (assigning values to abstract states or state action pairs) is defined as the average values of the states or state action pairs they represent. We will show that these abstract value functions cannot in general be learned using traditional MDP methods. This in turn provides some new insights into relational reinforcement learning approaches.

We proceed as follows. After introducing some mathematical preliminaries in Section 2, we present the LOMDP framework in Section 3. Section 4 defines abstract policies and shows how to compute the value of an abstract policy. This

results in the LQ learning algorithm presented in Section 5. The algorithm is experimentally evaluated in Section 6. Before concluding, we discuss related work.

2 Preliminaries

As logic programs and Markov decision processes will be used throughout this paper as the underlying mathematical concepts, we now briefly introduce their key concepts.

2.1 Logic

A *first-order alphabet* Σ is a set of relation symbols $\mathtt{r}$ with arity $m \geq 0$, and a set of functor symbols $\mathtt{f}$ with arity $n \geq 0$. If $n = 0$ then $\mathtt{f}$ is called a constant, if $m = 0$ then p is called a proposition. An *atom* $\mathtt{r}(\mathtt{t}_1, \ldots, \mathtt{t}_m)$ is a relation symbol $\mathtt{r}$ followed by a bracketed n-tuple of terms $\mathtt{t}_i$. A *term* is a variable $\mathtt{V}$ or a functor symbol immediately followed by a bracketed n-tuple of terms $\mathtt{t}_i$, i.e., $\mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$. A conjunction is a set of atoms. A conjunction A is said to be θ-subsumed by a conjunction B, denoted by $A \leq_\theta B$, if there exists a substitution θ such that $B\theta \subset A$. A term, atom or clause E is called *ground* when it contains no variables. The *most general unifier* (MGU) for atoms a and b is denoted by $\mathrm{mgu}(a, b)$. The *Herbrand base* of Σ, denoted as hb_Σ, is the set of all ground atoms constructed with the predicate and functor symbols in the alphabet Σ.

2.2 Notation

Atoms are written in lower case a, set of of atoms in upper case A, and sets of sets of atoms in bold, upper case $\mathbf{A}$. To highlight that a (resp. A and $\mathbf{A}$) may not be ground (i.e. it may contain variables), we will write $\mathbb{a}$ (resp. $\mathbb{A}$ and $\mathbb{A}$).

2.3 Markov Decision Processes

A Markov decision process (MDP) is a tuple $\mathbf{M} = (S, A, \mathbf{T}, \lambda)$. To avoid ambiguities, we will sometimes index the elements by $\mathbf{M}$. Here, S is a set of system states, i.e. propositions. The agent has available a finite set of actions $A(z) \subseteq A$ for each state $z \in S$ which cause stochastic state transitions. For each $z, z' \in S$ and $a \in A(z)$ there is a transition T in $\mathbf{T}$ which is an expression of the form $z' \xleftarrow{p:r:a} z$. The transition denotes that with probability $P(z, a, z') := p$ action a causes a transition to state z' when executed in state z. We have for each $z \in S$ and $a \in A(z)$ that $\sum_{z' \in S} P(z, a, z') = 1$. For a transition the agent gains an expected next reward $R(z, a, z') := r$. In case that the reward function R is probabilistic (mean value depends on the current state and action only) the MDP is called *nondeterministic*, otherwise *deterministic*. In this paper, we only consider MDPs with stationary transition probabilities and stationary, bounded rewards.

A (stationary) deterministic policy $\pi : S \mapsto A$ is a set of expressions of the form $a \leftarrow z$ for each $z \in S$ where $a \in A(s)$. It denotes a particular course of actions to be adopted by an agent, with $\pi(z) := a$ being the action to be executed whenever the agent finds itself in state z. We assume an infinite horizon and also that the agent accumulates the rewards associated with the states it enters. To compare policies, we use the expected total discounted reward as our optimality criterion, i.e., future rewards are discounted by $0 \leq \lambda < 1$. The value of a policy π can be shown to be $V_\pi(z) = \sum_{z' \xleftarrow{p:r:a} z \in \mathbf{T}} p \cdot [r + \lambda \cdot V_\pi(z')]$. The value of π at any initial state z can be computed by solving this system of linear equations. A policy π is optimal if $V_\pi(z) \geq V_{\pi'}(z)$ for all $z \in S$ and policies π'. A (stationary) nondeterministic policy π maps a state to a distribution over actions. The value of π is then the expectation according to this distribution.

3 Logical Markov Decision Programs

The *logical component* of a MDP corresponds to a *finite state automaton*. This is essentially a propositional representation because the state and action symbols are flat, i.e. not structured. The key idea underlying *logical Markov decision programs* (LOMDPs) is to replace these flat symbols by abstract symbols.

Definition 1. *An abstract state is a conjunction $\mathbb{Z}$ of logical atoms, i.e., a logical query. In case of an empty conjuction, we write $\emptyset$.*

Abstract states represent sets of states. More formally, we have that a state Z is a (finite) conjunction of ground facts over the alphabet Σ, i.e. a logical interpretation, a subset of the Herbrand base. In the blocks world, one possible state Z is $\mathtt{on(a,b)}$, $\mathtt{on(b,fl)}, \mathtt{bl(a)}, \mathtt{bl(b)}, \mathtt{cl(a)}, \mathtt{cl(fl)}$ where $\mathtt{on(a,b)}$ denotes that object $\mathtt{a}$ is on $\mathtt{b}$, $\mathtt{cl(a)}$ states that $\mathtt{a}$ is clear, $\mathtt{bl(a)}$ denotes that $\mathtt{a}$ is a block, and $\mathtt{fl}$ refers to the floor. An abstract state $\mathbb{Z}$ is e.g. $\mathtt{on(X,Y)}$, $\mathtt{bl(Y)}, \mathtt{bl(X)}$. It represents all states (over the given alphabet Σ) that have two blocks on one another. Formally, speaking, we have that an abstract state $\mathbb{Z}$ represents all states Z for which there exists a substitution θ such that $\mathbb{Z}\theta \subseteq Z$. Let $S(\mathbb{Z})$ denote this set of states. The substitution in the previous example is $\{\mathtt{X/a}, \mathtt{Y/b}\}$. By now we are able to define abstract transitions.

Definition 2. *An* abstract transition T *is an expression of the form* $\mathbb{H} \xleftarrow{p:r:\mathbb{a}} \mathbb{B}$ *where* $\mathrm{P(T)} := p \in [0, 1]$, $\mathrm{R(T)} := r \in [0, 1]$, $\mathbb{a}$ *is an abstract action, and* $\mathrm{body(T)} := \mathbb{B}$ *and* $\mathrm{head(T)} := \mathbb{H}$ *are abstract states.*

We assume T to be range-restricted, i.e., $\mathrm{vars}(\mathbb{H}) \subseteq \mathrm{vars}(\mathbb{B})$, and $\mathrm{vars}(\mathbb{a}) \subseteq \mathrm{vars}(\mathbb{B})$, so that an abstract transition relies on the information encoded in the current state only. The semantics of an abstract transition[1] are:

> *If the agent is in a state Z, such that $\mathbb{B} \leq_\theta Z$, then it will go to the state $Z' := [Z \setminus \mathbb{B}\theta] \cup \mathbb{H}\theta$ with probability p when performing action $\mathbb{a}\theta$ receiving an expected next reward of r.*

For illustration purposes [2], consider the following abstract transition, which moves block $\mathtt{X}$ from $\mathtt{Y}$ to the floor with probability 0.9:

$$\mathtt{on(X,fl)}, \mathtt{cl(X)cl(Y)} \xleftarrow{0.9:-1:\mathtt{mv\_fl(X)}} \mathtt{on(X,Y)}, \mathtt{cl(X)}$$

[1] We implicitly assume that an abstract action has some preconditions

[2] Please note that we employ functor-free examples throughout the paper for the sake of simplicity. Abstract states $\mathbb{Z}$, actions $\mathbb{A}$, and transitions $\mathbb{T}$ can include functors. All proofs remain valid.

Applied to state *Exp*

$$\texttt{on}(\texttt{a},\texttt{b}),\texttt{on}(\texttt{b},\texttt{fl}),\texttt{on}(\texttt{c},\texttt{fl}),$$
$$\texttt{cl}(\texttt{a}),\texttt{cl}(\texttt{c}),\texttt{bl}(\texttt{a}),\texttt{bl}(\texttt{b}),\texttt{bl}(\texttt{c})$$

the abstract transition tells us that when we execute $\texttt{mv\_fl}(\texttt{a})$ the successor state will be

$$\texttt{on}(\texttt{a},\texttt{fl}),\texttt{on}(\texttt{b},\texttt{fl}),\texttt{on}(\texttt{c},\texttt{fl}),$$
$$\texttt{cl}(\texttt{a}),\texttt{cl}(\texttt{b}),\texttt{cl}(\texttt{c}),\texttt{bl}(\texttt{a}),\texttt{bl}(\texttt{b}),\texttt{bl}(\texttt{c})$$

with probability 0.9 gaining a reward of -1. One can see that this implements a kind of first-order variant of probabilistic STRIPS operator, cf. [Hanks and McDermott, 1994].

As LOMDPs typically consist of a set $\mathbb{T}$ of multiple abstract transitions there are two constraints to be imposed in order to obtain meaningful LOMDPs. First, let $\mathbb{B}$ be the set of all bodies of abstract state transitions in the LOMDP (modulo variable renaming). For $\mathbb{B} \in \mathbb{B}$, let $\mathbb{A}(\mathbb{B})$ denote the set of all abstract actions $\mathbb{a}$ such that $\mathbb{H} \xleftarrow{p:r:\mathbb{a}} \mathbb{B}$ is in the LOMDP. We require

$$\forall \mathbb{B} \in \mathbb{B}, \forall \mathbb{a} \in \mathbb{A}(\mathbb{B}) \sum_{\substack{\mathrm{T}\in\mathbb{T},\\ \mathrm{body(T)}=\mathbb{B},\\ \mathrm{act(T)}=\mathbb{a}}} \mathrm{P(T)} = 1.0. \qquad (1)$$

This condition guarantees that all abstract successor states are specified when executing an abstract action in an abstract state and that their probabilities sum to 1. Secondly, we need a way to cope with contradicting transitions and rewards. Indeed, consider the two transitions $\texttt{e} \xleftarrow{1:-1:\texttt{a}} \texttt{d}$ and $\texttt{g} \xleftarrow{1:-2:\texttt{a}} \texttt{f}$, and state $Z = \{\texttt{d},\texttt{f}\}$. The problem with these transitions is that the first transition says that if we execute $\texttt{a}$ in Z we will go with probability 1 to state $Z' = \{\texttt{e},\texttt{f}\}$ whereas the second assigns a probability of 1 to state $Z'' = \{\texttt{d},\texttt{g}\}$. There are essentially two ways to deal with this situation. On the one hand, one might want to combine the two transitions and assign a probability of 0.5 to both Z' and Z'' for Z. On the other hand, one might want to have only one of rule of fire. In this paper, we take the second approach because this allows us to consider the transitions more independently of one another. This in turn will simplify learning and yields locally interpretable models. We assume a total order $\prec$ over all action-body pairs in $\mathbb{T}$ and do a forward search among the pairs stopping with the first matching body such as in Prolog[3]. From now on, we assume $\mathbb{B}$ to be ordered w.r.t. $\prec$. We will give an example after the next definition.

By now we are able to formally define logical Markov decision programs.

Definition 3. *A logical Markov decision process (LOMDP) is a tuple* $\mathbb{M} = (\Sigma, \mathbb{A}, \mathbb{T}, \lambda)$ *where* Σ *is a logical alphabet,* $\mathbb{A}$*, is a set of abstract actions,* $\mathbb{T}$ *is a finite set of abstract state transitions based on actions in* $\mathbb{A}$*, and* $0 \leq \lambda < 1$ *is a discount factor, such that* (1) *holds.*

[3]We chose a total order for the sake of simplicity. A partial order $\prec$ among the pairs s.t. the set of pairs is well-founded, i.e., every descending chain of elements w.r.t. $\prec$ is finite, actually suffices. Then, the conflict resolution strategy is to select only those abstract transitions whose action-body pair is minimal. An example is given in [Kersting *et al.*, 2003] where a kind of subsumption (or generality) relation among $\mathbb{B}$ is employed. All theorems can be adapted accordingly.

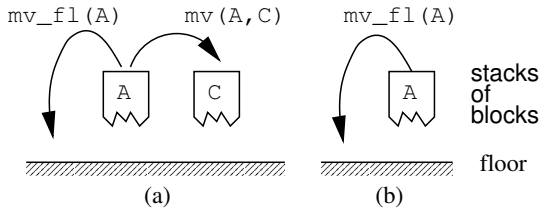

Figure 1: The two underlying patterns of the blocks world. Figure (a) shows the situation that there are at least two stacks of height > 0. Figure (b) shows the situation that there is only one stack left. The serrated cuts indicate that `A` (resp. `C`) can be on top of some other block or on the floor.

Before giving the semantics of LOMDPs, let us also illustrate LOMDPs on the *stack* example from the blocks world:

1:	absorb	$\xleftarrow{1.0:0.0:\texttt{absorb}}$	absorb.
2:	$\texttt{on}(\texttt{A},\texttt{fl}),\texttt{cl}(\texttt{A}),$ $\texttt{on}(\texttt{C},\texttt{D}),\texttt{cl}(\texttt{C}),$ $\texttt{cl}(\texttt{B})$	$\xleftarrow{0.9:-1:\texttt{mv\_fl(A)}}$	$\texttt{on}(\texttt{A},\texttt{B}),\texttt{cl}(\texttt{A}),$ $\texttt{on}(\texttt{C},\texttt{D}),\texttt{cl}(\texttt{C}).$
3:	$\texttt{on}(\texttt{A},\texttt{C}),\texttt{cl}(\texttt{A}),$ $\texttt{on}(\texttt{C},\texttt{D}),\texttt{cl}(\texttt{C}),$ $\texttt{cl}(\texttt{B})$	$\xleftarrow{0.9:-1:\texttt{mv(A,C)}}$	$\texttt{on}(\texttt{A},\texttt{B}),\texttt{cl}(\texttt{A}),$ $\texttt{on}(\texttt{C},\texttt{D}),\texttt{cl}(\texttt{C}).$
4:	absorb	$\xleftarrow{1.0:20:\texttt{stop}}$	$\texttt{on}(\texttt{A},\texttt{B}),\texttt{cl}(\texttt{A}),$ $\texttt{bl}(\texttt{B}).$

If the transition probabilities do not sum to 1.0 for an abstract action then there is an additional abstract transition for staying in the current abstract state. In order to understand the LOMDP *stack*, one has to understand the abstract states that govern the underlying patterns of the blocks world, cf. Figure 1. Two abstract states (the artificial `absorb` state excluded) together with the order in which they are presented cover all possible state action patterns because we can take advantage of symmetry in the blocks world. Transition 1 encodes the absorbing state. Transitions 2 and 3 cover the cases in which there are (at least) two stacks. Finally, transition 4 encodes the situation that there is only one stack, i.e. our goal state *stack*. Here, $\texttt{on}(\texttt{A},\texttt{B}),\texttt{cl}(\texttt{A}),\texttt{bl}(\texttt{B})$ are only used to describe the preconditions of $\texttt{mv}(\texttt{A},\texttt{B})$: the floor cannot be moved. When performing action $\texttt{mv}(\texttt{a},\texttt{b})$ in state *Exp* (see above) only abstract transitions 4 is firing. Similar, we can easily encode the *unstack* goal.

Note that we have not specified the number of blocks. The LOMDP represents all possible blocks worlds using only 6 abstract transitions, i.e. 12 probability and reward parameters, whereas the number of parameters of a propositional system explodes: for 4 blocks there are 73 states, for 7 blocks 37.663 states, and for 10 blocks $58.941.091$ states, resulting in an even higher number of transitions.

The semantics of LOMDPs are as follows.

Theorem 1. *Every LOMDP* $\mathbb{M} = (\Sigma, \mathbb{A}, \mathbb{T}, \lambda)$ *specifies a discrete MDP* $\mathbf{M}(\mathbb{M}) = (S, A, \mathbf{T}, \lambda)$.

Proof sketch: Let $\mathrm{hb}^{\mathrm{s}}_{\Sigma} \subset \mathrm{hb}_{\Sigma}$ be the set of all ground atoms built over abstract **s**tates predicates, and let $\mathrm{hb}^{a}_{\Sigma} \subset$

hb_Σ be the set of all ground atoms built over abstract **a**ction names. Now, construct $\mathbf{M}(\mathbb{M})$ from $\mathbb{M}$ as follows. The countable state set S consists of all finite subsets of hb^s_Σ. The set of actions $\mathbf{A}(Z)$ for state $Z \in S$ is given by $\mathbf{A}(Z) = \{\mathtt{a}\theta | \mathbb{H} \xleftarrow{p:r:\mathtt{a}} \mathbb{B} \in \mathbb{T}$ minimal (w.r.t. $\prec$) , $\mathbb{B} \leq_\theta Z\}$. We have that $|\mathbf{A}(Z)| < \infty$ holds. The probability $P(Z, a, Z')$ of a transition in $\mathbf{T}$ from Z to another state Z' after performing an action a is the probability value p associated to the unique abstract transition matching Z, a, and Z' normalized by the number of transitions of the form $Z'' \xleftarrow{a} Z$ in $\mathbb{T}$. If there is no abstract transition connecting Z and Z', the probability is zero. The bounded rewards are constructed in a similar way but are not normalized. □

From Theorem 1 and [Puterman, 1994, Theorem 6.2.5] it follows that:

Corollary 1. *For every LOMDP, there exists an optimal policy (for the ground states).*

Finally, LOMDPs generalize finite MDPs.

Proposition 1. *Every* finite *MDP is a propositional LOMDP in which all relation symbols have arity* 0*.*

4 Abstract Policies

Theorem 1 states that every LOMDP $\mathbb{M}$ specifies a discrete MDP $\mathbf{M}(\mathbb{M})$. Furthermore, Corollary 1 guarantees that there exists an optimal policy π for MDP $\mathbf{M}(\mathbb{M})$. Of course, this policy is extensional or propositional in the sense that it specifies for each ground state separately which action to execute. Specifying such policies for LOMDPs with large state spaces is cumbersome and learning them will require much effort. Therefore, we introduce *abstract policies* $\boldsymbol{\pi}$ which intentionally specify the action to take for an abstract state (or sets of states).

Definition 4. *An abstract policy* $\boldsymbol{\pi}$ *over* Σ *is a finite set of decision rules of the form* $\mathtt{a} \leftarrow \mathbb{L}$ *where* $\mathtt{a}$ *is an abstract action and* $\mathbb{L}$ *is an abstract state*[4].

The meaning of a decision rule $\mathtt{a} \leftarrow \mathbb{L}$ is that

> *if the agent is in a state* Z *such that* $\mathbb{L} \leq_\theta Z$ *then the agent will perform action* $\mathtt{a}\theta$*, denoted by* $\boldsymbol{\pi}(Z)$*.*

Usually, $\boldsymbol{\pi}$ consists of multiple decision rules. We apply the same conflict resolution technique as for abstract transitions, i.e. we use a total order $\prec$ among the decision rules. Let $\mathbb{L} = \{\mathbb{L}_1, \ldots, \mathbb{L}_m\}$ be the set of bodies in $\boldsymbol{\pi}$ (ordered w.r.t. $\prec$). We call $\mathbb{L}$ the *abstraction level* of $\boldsymbol{\pi}$. We assume that $\mathbb{L}$ covers all possible states of the LOMDP. This together with the total order guarantees that $\mathbb{L}$ forms a partition of the states. The equivalence classes $[\mathbb{L}_1], \ldots, [\mathbb{L}_m]$ induced by $\mathbb{L}$ are inductively defined by $[\mathbb{L}_1] = S(\mathbb{L}_1)$, and for $i \geq 2$, $[\mathbb{L}_i] = S(\mathbb{L}_i) \setminus \bigcup_{j=1}^{i-1} [\mathbb{L}_j]$. Because $\mathbb{L}$ generally does not coincide with $\mathbb{B}$ the following proposition holds.

Proposition 2. *Any abstract policy* $\boldsymbol{\pi}$ *specifies a* nondeterministic *policy* π *at the level of ground states.*

Let $\mathbb{M}$ be a LOMDP and let $\mathbf{M}(\mathbb{M})$ be the induced MDP. We define the expected reward of $\mathbb{L} \in \mathbb{L}$ to be the expected reward taken over all states in $[\mathbb{L}]$. Therefore, the expected discounted reward, if abstract policy $\boldsymbol{\pi}$ is used and the system is in abstract state $\mathbb{L}$, is defined to be

$$V_{\boldsymbol{\pi}}(\mathbb{L}) = \lim_{N \to \infty} E_{[\mathbb{L}]} \left[E_\pi \left\{ \sum_{k=1}^{N} \lambda^k r_{t+k} | Z_t = Z \right\} \right] \qquad (2)$$

where r_i denotes the value at time i of the reward received w.r.t. $\mathbf{M}(\mathbb{M})$ when following the ground level policy π induced by $\boldsymbol{\pi}$. The inner expectation E_π is conditioned on the system being in state $Z \in S$ at time t, denoted by $Z_t = Z$. The outer expectation $E_{[\mathbb{L}]}$ runs over all elements of $[\mathbb{L}]$. The series in (2) converges absolutely for the same reasons as for MDPs. Thus, the limit and the expectations are interchangeable in (2):

$$V_{\boldsymbol{\pi}}(\mathbb{L}) = E_{[\mathbb{L}]} \left[E_\pi \left\{ \sum_{k=1}^{\infty} \lambda^k r_{t+k} | Z_t = Z \right\} \right] . \qquad (3)$$

The abstract Q function is defined analogously. Now, an abstract policy $\boldsymbol{\pi}$ is *discount optimal at abstraction level* $\mathbb{L}$ for fixed λ whenever $V_{\boldsymbol{\pi}}(\mathbb{L}) \geq V_{\boldsymbol{\pi}'}(\mathbb{L})$ for all $\mathbb{L} \in \mathbb{L}$ and abstract policies $\boldsymbol{\pi}'$ at abstraction level $\mathbb{L}$. Note, that optimality at abstraction level $\mathbb{L}$ does not imply optimality at the level of ground states. This is because an abstract policy specifies the expected behaviour of a set of ground states. The problem is now to compute the value function $V_{\boldsymbol{\pi}}$.

Let $\mathbb{M} = (\Sigma, \mathbb{A}, \mathbb{T}, \lambda)$ be a LOMDP, and let $\boldsymbol{\pi}$ be an abstract policy at abstraction level $\mathbb{L} = \{\mathbb{L}_1, \ldots, \mathbb{L}_m\}$. Consider the finite MDP $\mathbf{L} = (\{l_1, \ldots, l_m\}, A_{\mathbf{L}}, \mathbf{T}_{\mathbf{L}}, \lambda)$ wich is constructed as follows.

Construction: Both $\mathbb{L}$ and $\mathbb{B}$ (the set of bodies in $\mathbb{T}$) induce partitions $\{[\mathbb{L}_1], \ldots, [\mathbb{L}_m]\}$ (resp. $\{[\mathbb{B}_1], \ldots, [\mathbb{B}_n]\}$) of $S_{\mathbf{M}(\mathbb{M})}$ because both are ordered. The state l_i corresponds to $[\mathbb{L}_i]$. Furthermore, all ground states belonging to $[\mathbb{L}_i] \cap [\mathbb{B}_k]$ have the same set of possible transitions. In other words, $[\mathbb{L}_i] \cap [\mathbb{B}_k]$ forms an equivalence class. Now, there is a transition $T \in \mathbf{T}_{\mathbf{L}}$ from state l_i to l_j when doing action a with probability

$$P(l_i, a, l_j) := \sum_{\mathbb{H} \xleftarrow{p:r:\mathtt{a}} \mathbb{B} \in \mathbb{T}} \mu([\mathbb{B}] | [\mathbb{L}_i]) \cdot p \cdot \mu([\mathbb{L}_j] | S(\mathbb{H}))$$

Here, $\mu(X|Y)$ is a probability function. The value $\mu(X|Y)$ for $X, Y \subset S_{\mathbf{M}(\mathbb{M})}$ is the probability that a randomly selected ground state in Y is an element of X. Because $\mathbf{M}(\mathbb{M})$ induces a unique probability distribution over all ground states, μ is uniquely specified. This follows from Theorem 1. Clearly,

$$\sum_{l_j} P(l_i, a, l_j) = 1 .$$

The intuition behind $P(l_i, a, l_j)$ is that it specifies $P(\mathbb{L}_i, \mathtt{a}, \mathbb{L}_j)$ for the corresponding abstract states. The probabilistic reward $R(l_i, a, l_j)$ depends only on l_i and A, and can

[4] We assume that $\mathtt{a}$ is applicable in $\mathbb{L}$.

be chosen [5] s.t. its mean value equals

$$R(l_i, a) := \sum_{l_j} P(l_i, a, l_j) \cdot R(l_i, a, l_j) \ . \quad \square$$

As the underlying MDP $\mathbf{M}(\mathbb{M})$ is not known, the problem specified by $\mathbf{L}$ appears to a learner to have a non-Markovian nature. Consider the following LOMDP $\mathbb{M}$

$$\begin{array}{lcll} 1: & \mathtt{q} & \xleftarrow{1.0:0.0:\mathtt{a}} & \mathtt{p}, \mathtt{q}. \\ 2: & \emptyset & \xleftarrow{1.0:1.0:\mathtt{a}} & \mathtt{p}. \\ 3: & \mathtt{p} & \xleftarrow{1.0:0.0:\mathtt{a}} & \emptyset. \end{array}$$

and the abstraction level $\mathbb{L} = \{\mathtt{p}, \mathtt{q}, \emptyset\}$. The induced MDP $\mathbf{L}$ will assign the same probabilities and rewards to the transitions from l_2 to l_1 and from l_3 to l_1. Consequently, the values for l_2 and l_3 are the same in $\mathbf{L}$ as the next state is the same namely l_1, but $\mathbb{M}$ assigns different values to both.

The example shows that a learner following $\mathbf{L}$ has imperfect and incomplete perception of the states of $\mathbf{M}(\mathbb{M})$. This is interesting because $\mathbf{L}$ corresponds to leafs of a first order decision tree used in relational reinforcement learning [Džeroski *et al.*, 2001]. Unfortunately, complete observability is necessary for learning methods based on MDPs. Thus in general, we must use techniques for solving *partially observable* MDPs, see e.g. [Kaelbling *et al.*, 1996]. In the present paper, we follow the most naive approach to deal with partially observability, namely ignoring it. That is, we treat the induced MDP $\mathbf{L}$ as if it would be the correct underlying MDP.

5 LQ-Learning

In principle, any known algorithm for computing an optimal policy for $\mathbf{L}$ can be used. There are only two complications. First, the probability function μ is not given. This problem can however be solved using stochastic iterative dynamic programming, i.e. model-free approaches. Second, we do not want to construct $\mathbf{L}$. Instead, we directly want to use $\mathbb{L}$. Below, we sketch LQ learning, which learns the Q function of $\mathbf{L}$ using this idea combined with traditional Q learning. Similar, other methods such as MC, SARSA and actor-critic methods can be adapted.

Logical Q Learning
Let $\mathbb{L}$ be an abstraction level
Initialize $\widehat{Q}_0(\mathbb{L}, \mathtt{a})$ arbitrarily for each $\mathbb{L} \in \mathbb{L}$
n=1
Repeat (for each episode)
  Initialize ground state $Z \in S_{\mathbf{M}(\mathbb{M})}$
  Repeat (for each step in episode)
    Choose action a in Z based on $\widehat{Q}_{n-1}$, cf. (4)
    Let $\mathtt{a}$ be the abstract action corresponding to a
    Take action a, observe r and successor state Z'
    Let $\mathbb{L} \in \mathbb{L}$ (resp. $\mathbb{L}' \in \mathbb{L}$) be the unique abstract state matching Z (resp. Z')
    $\alpha_n := (1 + \text{visits}_n(\mathbb{L}, \mathtt{a}))^{-1}$
    $\widehat{Q}(\mathbb{L}, \mathtt{a})_n := (1 - \alpha_n) \cdot \widehat{Q}_{n-1}(\mathbb{L}, \mathtt{a})$
      $+\alpha_n \cdot (r + \lambda \cdot \max_{\mathtt{a}'} \widehat{Q}_{n-1}(\mathbb{L}', \mathtt{a}'))$
    Set $Z := Z'$ and $n := n + 1$
  Until Z is terminal

Here, $\text{visits}_n(\mathbb{L}, \mathtt{a})$ is the total number of times the abstract state – abstract action pair has been visited up to and including the n-th iteration. $\widehat{Q}(\mathbb{L}, \mathtt{a})_n$ is the approximation of $Q(\mathbb{L}, \mathtt{a})$ after n iterations. To select an action a, we first probabilistically select an abstract action $\mathtt{a}$ in a state $\mathbb{L}$ so that the probability $P(\mathtt{a}|\mathbb{L})$ of selection $\mathtt{a}$ is proportional to $\widehat{Q}(\mathbb{L}, \mathtt{a})_n$, e.g.

$$P(\mathtt{a}|\mathbb{L}) = \frac{T^{\widehat{Q}_n(\mathbb{L}, \mathtt{a})}}{\sum_j T^{\widehat{Q}_n(\mathbb{L}, \mathtt{a}_j)}} \tag{4}$$

with $T > 0$. This is common in Q learning. Then, we select uniformly among all possible ground action given by $\mathtt{a}$ and Z to get a.

Let us now argue that LQ learning converges with respect to $\mathbf{L}$. Each selection of a ground state Z selects a unique state l_i in $\mathbf{L}$. Likewise, when we have observed Z', this uniquely specifies a state l_j. The rewards are stochastic, but they depend on Z and a only. Therefore, the convergence theorem for Q-learning for finite (nondeterministic) MDPs applies to $\mathbf{L}$, cf. [Watkins and Dayan, 1992; Jaakkola *et al.*, 1994]. Moreover, it might be the case that LQ learning can do even better. The equality $V_{\boldsymbol{\pi}}(\mathbb{L}_i) = V_\pi(l_i)$ seems to hold if for each legal trace of $\mathbf{L}$ we can find a legal trace within $\mathbf{M}(\mathbb{M})$. Due to the abstraction, LQ learning should generalize well even in unseen ground states.

6 Experiments

We implemented LQ learning using the Prolog system Sicstus-3.9.0. Our task was to learn an abstract policy for the *stack* LOMDP (see above). This task was motivated by the experiments in relational reinforcement learning (RRL) [Džeroski *et al.*, 2001] and by the fact that the blocks world is the prototypical toy domain requiring relational representations. One of the key differences with the experiments reported by [Džeroski *et al.*, 2001] is that we exclusively use the standard predicates `on`, `cl`, and `bl`. [Džeroski *et al.*, 2001] also needed to make use of several background knowledge predicates such as `above`, `height` of stacks as well as several directives to the inductive logic programming function approximator in order to be able to learn adequate policies. Another difference to our approach is that RRL induces the relevant abstract states automatically using a regression tree learner.

The discount factor was 0.9, and the temperature T to select an action was increased by 1.004 each epoch starting with 1.0. Therefore, the agent favors exploration during

[5] A nondeterministic MDP can be converted into a deterministic one. Maximizing the expected future reward depends only on the expected reward in each state, and not on the probability distribution over rewards. In our case, $R(l_i, a, l_j) := \sum_{\mathbb{H} \xleftarrow{p:r:\mathtt{a}} \mathbb{B} \in \mathbb{T}} \mu([\mathbb{B}]|[\mathbb{L}_i]) \cdot p \cdot \mu([\mathbb{L}_j]|\mathbf{S}(\mathbb{H})) \cdot r$ would do.

early states of learning, then gradually shifts towards a strategy of exploration. We randomly generated 10 blocks world states for 4 blocks, 20 for 6 blocks, 30 for 8 blocks, and 50 for 10 blocks using the procedure described by [Slaney and Thiébaux, 2001]. Note that for 10 blocks a propositional MDP would have to represent 58.941.091 states of which 3.628.800 states are goal states. Then, we ran LQ learning on these starting states in order $4, 6, 8$ and 10 blocks. The initial Q function was

$$Q\left(\left\{\begin{array}{c}\texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ \texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)},\texttt{bl(B)},\texttt{bl(D)}\end{array}\right\},\texttt{mv\_fl(A)}\right)=0.0$$
$$Q\left(\left\{\begin{array}{c}\texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ \texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)},\texttt{bl(B)},\texttt{bl(D)}\end{array}\right\},\texttt{mv(A,C)}\right)=0.0$$
$$Q\left(\left\{\begin{array}{c}\texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ \texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)},\texttt{bl(B)},\texttt{bl(D)}\end{array}\right\},\texttt{mv(A,E)}\right)=0.0$$
$$Q\left(\left\{\begin{array}{c}\texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ \texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)},\texttt{bl(B)},\texttt{bl(D)}\end{array}\right\},\texttt{mv(E,A)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(C,D)},\texttt{cl(A)},\texttt{cl(C)}\},\texttt{mv\_fl(A)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(C,D)},\texttt{cl(A)},\texttt{cl(C)}\},\texttt{mv(A,C)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(E,fl)},\texttt{cl(A)},\texttt{cl(E)}\},\texttt{mv\_fl(A)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(E,fl)},\texttt{cl(A)},\texttt{cl(E)}\},\texttt{mv(A,E)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{on(E,fl)},\texttt{cl(A)},\texttt{cl(E)}\},\texttt{mv(E,A)}\right)=0.0$$
$$Q\left(\{\texttt{on(A,B)},\texttt{cl(A)}\},\texttt{stop}\right)=0.0$$
$$Q\left(\{\texttt{cl(A)},\texttt{cl(B)}\},\texttt{mv(A,B)}\right)=0.0$$

where we omitted the `absorb` state in front. The whole experiment was repeated 5 times (including sampling the starting states). In all 5 runs, the learned policy (which is optimal at the given abstraction level) was:

$$\begin{array}{rcl}\texttt{mv\_fl(A)} & \leftarrow & \texttt{on(A,B)},\texttt{on(C,D)},\texttt{on(E,fl)},\\ & & \quad\texttt{cl(A)},\texttt{cl(C)},\texttt{cl(E)}.\\ \texttt{mv\_fl(A)} & \leftarrow & \texttt{on(A,B)},\texttt{on(C,D)},\texttt{cl(A)},\texttt{cl(C)}.\\ \texttt{mv(E,A)} & \leftarrow & \texttt{on(A,B)},\texttt{on(E,fl)},\texttt{cl(A)},\texttt{cl(E)}.\\ \texttt{mv(A,B)} & \leftarrow & \texttt{cl(A)},\texttt{cl(B)}.\end{array}$$

The learned policy is interesting for many reasons. First, it uniquely specifies a deterministic policy for ground states. Second, it is well known in the planning community [Slaney and Thiébaux, 2001]. It is called *unstack-stack* strategy because it amounts to putting all misplaced blocks on the table and then building the goal state by stacking all blocks from the floor onto one single stack. The total number of moves is at worst twice the optimal. Third,*unstack-stack* perfectly generalizes to all other blocks worlds, no matter how many blocks there are. Finally, it cannot be learned in a propositional setting because here the optimal policy would encode the optimal number of moves.

RRL has learned another policy ("move a block to the highest stack") than LQ learning. However, as argued above, this policy can only be described using additional background predicates, which are not needed in our approach. We believe that RRL would have difficulties in learning the unstack-stack policy using only the predicates `on`, `cl` and `bl`.

Rerunning the experiments with a simpler abstract Q function, omitting the first four abstract values, yields the *unstack-stack* policy, too, but the learning epochs were faster proceeded due to the higher abstraction.

7 Related Work

Within reinforcement learning (RL), there is currently a significant interest in using rich representation languages. [Finney *et al.*, 2002] investigated propositionalization methods in relational domains. They experimentally studied the intermediate language of *deictic representations* (DRs). DRs avoid enumerating the domain by using variables such as *the-block-on-the-floor*. Although DRs have led to impressive results [McCallum, 1995; Whitehead and Ballard, 1991], [Finney *et al.*, 2002]'s results show that DR may also degrade learning performance within relational domains. According to [Finney *et al.*, 2002], *Relational reinforcement learning* (RRL) [Džeroski *et al.*, 2001] is one way to effective learning in domains with objects. RRL is a combination of RL and inductive logic programming (ILP) [Muggleton and De Raedt, 1994]. The key idea is that the Q function is approximated using a relational regression tree learner. Although the experimental results are interesting, RRL has failed to explain – in theoretical terms – why RRL works. Some new insights on this have been obtained.

From a more general point of view, our approach is closely related to *decision theoretic regression* (DTR) [Boutilier *et al.*, 2000]. Here, state spaces are characterized by a number of random variables and the domain is specified using logical representations of actions that capture the regularities in the effects of actions. Because 'existing DTR algorithms are all designed to work with *propositional* representations of MDPs', [Boutilier *et al.*, 2001] proposed *first order DTR* which is a probabilistic extension of Reiter's *situation calculus*. The language is certainly more expressive than that of LOMDPs. However, it is also much more complex. Furthermore, [Boutilier *et al.*, 2001] assume that the model is given whereas in the present paper traditional model-free learning methods have been apply.

The idea of solving large MDP by a reduction to an equivalent, smaller MDP is also discussed e.g. in [Dearden and Boutilier, 1997; Givan *et al.*, 2003; Ravindran and Barto, 2002]. However there, only finite MDPs and no relational or first order representations have been investigated. Furthermore, there has been great interest in abstraction on other levels than state spaces. Abstraction over time [Sutton *et al.*, 1999] or primitive actions [Dietterich, 2000; Andre and Russell, 2001] are useful ways to abstract from specific subactions and time. This research is orthogonal and could be applied to LOMDPs in the future.

Finally, [Baum, 1999] reports on solving blocks worlds with up to 10 blocks using RL related techniques. However, the introduced language is domain-dependent and does not incorporate logic programming.

8 Conclusions

We have presented a representation framework that integrates Markov decision processes with logic programs. This frame-

work allows one to compactly and declaratively represent complex (relational) Markov decision processes. Using functors they might even be infinite. Furthermore, we have introduced abstract policies for LOMDPs and studied their properties. We have shown that their value functions cannot generally be learned using MDP techniques. However, the experiments with a simple upgrade of Q-learning have shown that even naive strategies to handle partially observability can sometimes work. The authors hope that this framework will be useful as a starting point for further theoretical developments in relational reinforcement learning.

Acknowledgements

The authors are deeply grateful Bob Givan for pointing out problems with an earlier version of this paper and for providing the example LOMDP at the end of Section 4.

References

[Anderson *et al.*, 2002] C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov Models and their Application to Adaptive Web Navigation. In D. Hand, D. Keim, O. R. Zaïne, and R. Goebel, editors, *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, pages 143–152, Edmonton, Canada, 2002. ACM Press.

[Andre and Russell, 2001] D. Andre and S. Russell. Programmable reinforcement learning agents. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 1019–1025. MIT Press, 2001.

[Baum, 1999] E. B. Baum. Towards a Model of Intelligence as an Economy of Agents. *Machine Learning*, 35(2):155–185, 1999.

[Boutilier *et al.*, 2000] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121, 2000.

[Boutilier *et al.*, 2001] C. Boutilier, R. Reiter, and B. Price. Symbolic Dynamic Programming for First-order MDPs. In B. Nebel, editor, *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 690–700, Seattle, Washington, USA, 2001. Morgan Kaufmann.

[Dearden and Boutilier, 1997] R. Dearden and C. Boutilier. Abstraction and approximate decision theoretic planning. *Artificial Intelligence*, 89(1):219–283, 1997.

[Dietterich, 2000] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[Džeroski *et al.*, 2001] S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine Learning*, 43(1/2):7–52, 2001.

[Finney *et al.*, 2002] S. Finney, N. H. Gardiol, L. P. Kaelbling, and T. Oates. The thing that we tried didn't work very well: Deictic representation in reinforcement learning. In *Proceedings of the Eighteenth International Conference on Uncertainty in Artificial Intelligence (UAI-02)*, 2002.

[Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence (IJCAI-99)*, pages 1300–1309, Stockholm, Sweden, 1999. Morgan Kaufmann.

[Givan *et al.*, 2003] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 2003. (in press).

[Hanks and McDermott, 1994] S. Hanks and D. V. McDermott. Modelling a dynamic and uncertain world I: Symbolic and probabilisitic reasoning about change. *Artificial Intelligence*, 66(1):1–55, 1994.

[Jaakkola *et al.*, 1994] T. Jaakkola, M. I. Jordan, and S. P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6), 1994.

[Kaelbling *et al.*, 1996] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research (JAIR)*, pages 237–285, 1996.

[Kersting and De Raedt, 2001] K. Kersting and L. De Raedt. Towards Combining Inductive Logic Programming with Bayesian Networks. In Céline Rouveirol and Michèle Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157 of *LNAI*, pages 118–131. Springer, 2001.

[Kersting *et al.*, 2003] K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden markov models. In R. B. Altman, A. K. Dunker, L. Hunter, T. A. Jung, and T. E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing*, pages 192 – 203, Kauai, Hawaii, USA, 2003. World Scientific.

[McCallum, 1995] A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden States*. PhD thesis, Department of Computer Science, University of Rochester, 1995.

[Muggleton and De Raedt, 1994] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19(20):629–679, 1994.

[Muggleton, 1996] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[Puterman, 1994] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[Ravindran and Barto, 2002] B. Ravindran and A. G. Barto. Model Minimization in Hierarchical Reinforcement

Learning. In *Proceedings of the Fifth International Symposium on Abstraction, Reformulation and Approximation (SARA-02)*, volume 2371 of *LNCS*, pages 196–211, Kananaskis, Alberta, Canada, 2002. Springer.

[Slaney and Thiébaux, 2001] J. Slaney and S. Thiébaux. Blocks World revisited. *Artificial Intelligence*, 125:119–153, 2001.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[Sutton *et al.*, 1999] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[Watkins and Dayan, 1992] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279 – 292, 1992.

[Whitehead and Ballard, 1991] S. D. Whitehead and D. H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45 – 83, 1991.