Challenges for Relational Reinforcement Learning

Martijn Van Otterlo OTTERLO@CS.UTWENTE.NL
University of Twente, Department of Computer Science, P.O. Box 217, 7500 AE Enschede, The Netherlands

Kristian Kersting KERSTING@INFORMATIK.UNI-FREIBURG.DE
University of Freiburg, Institute for CS, ML Lab, Georges-Koehler-Alle 079, 9110 Freiburg, Germany

Abstract

We present a perspective and challenges for Relational Reinforcement Learning (RRL). We first survey existing work and distinguish a number of main directions. We then highlight some research problems that are intrinsically involved in abstracting over relational Markov Decision Processes. These are the challenges *of* RRL. In addition, we describe a number of issues that will be important for further research into RRL. These are the challenges *for* RRL and deal with newly arising issues because of relational abstraction.

1. Introduction

There has been much progress in reinforcement learning and Markov decisions processes recently. Several basic algorithms have been proposed and their behavior is relatively well understood today, cf. (Sutton & Barto, 1998). This has led to an increased interest into the effects of generalization and introduced new challenges. One of them concerns the use of reinforcement learning in relational domains in which one can generalize over *objects* and *relations* (Kaelbling et al., 2001). Even though a number of relational reinforcement learning (RRL) algorithms has been developed, the problem is still not well understood and a theory seems to be lacking. Progress has been made in model-based approaches as well, but many open problems and challenges remain.

In this position paper, we provide a perspective on the challenges of RRL, based on existing techniques in the literature and insights from previous work. After an initial overview of the field, we highlight challenges *of* RRL, i.e. the intrinsic problems that have to be tackled by RRL systems. After that we point to challenges *for* RRL, i.e., new challenges that arise because of relational abstraction, and which require a *theory* of RRL to be developed.

Appearing in *Proceedings of the ICML'04 workshop on Relational Reinforcement Learning*, Banff, Canada, 2004.

2. Modeling Relational MDPs

Relational Reinforcement Learning differs from traditional MDP-based research in that the underlying MDP is now factored into relational atoms. Solving these *Relational Markov Decision Processes* (RMDP) involves the use of logical languages to abstract over relational states and actions, and the definition of abstract value functions and policies. Some progress was made on an *intermediate* level using *deictic representations* (Finney et al., 2002) but in the current paper we focus on *purely* relational representations.

Conceptually, one can view the (***g**round*) RMDP as the *semantics* of the logical ***r**epresentation* level which we name the g-level and the r-level, respectively.

2.1. Relational Markov Decision Processes

The basic ingredients of an RMDP are a set of *predicates* P and a domain of *constants* C. Furthermore, one needs a set of *action* predicates A. The *Herbrand base* $\mathrm{HB}^{\mathtt{P}\cup\mathtt{C}}$ is the set of all ground atoms which can be constructed from P and C. The set of all states S' then consists of the powerset of $\mathrm{HB}^{\mathtt{P}\cup\mathtt{C}}$. Usually there is an implicit logical background theory Σ that excludes illegal states, and thus the *state space* S of the RMDP is $\{s \in S' \mid \Sigma \models s\}$. Similarly, an action space A can be defined based on A and C.

A relational MDP is now defined by the tuple $\langle S, A, T, R\rangle$ where $T : S \times A \times S \to [0,1]$ a *transition function* and $R : S \times A \times S \to [0,1]$ a *reward function*. Modeled in this way, we have essentially a discrete MDP, but the size of the state space grows exponentially faster than propositionally factored MDPs.

As an example, based on the predicates `on/2` and `cl/1`, the domain $\{\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{e},\texttt{floor}\}$ and the action set `move/2` we can define the blocks world containing 5 blocks with $|S| = 501$ legal states.

2.2. Many routes lead to Abstraction

The g-level consists of the ground RMDP. Associated with this level are optimal state (V^*) and state-action (Q^*) value functions and optimal policies π^*. However, because of the large sizes of the g-level some level of abstraction is needed, i.e. the r-level. In developing techniques for solving RMDPs one can (Van Otterlo, 2002) upgrade existing propositional techniques to the relational case or extend existing (agent) logics with utilities. Various logical languages can be employed such as *Horn logic*, *situation calculus* and *description logic*. For example, consider the following *abstract* policy for a blocks world. It optimally encodes the policy for reaching states where on(a, b) holds:

$\texttt{onTop}(\texttt{X},\texttt{b}) \rightarrow \texttt{move}(\texttt{X},\texttt{floor})$
$\texttt{onTop}(\texttt{Y},\texttt{a}) \rightarrow \texttt{move}(\texttt{Y},\texttt{floor})$
$\texttt{cl}(\texttt{a}),\texttt{cl}(\texttt{b}) \rightarrow \texttt{move}(\texttt{a},\texttt{b})$
$\texttt{on}(\texttt{a},\texttt{b}) \rightarrow \texttt{noop}$

Thus, the goal in research on RMDPs is finding an (optimal) policy $\hat{\pi}^* : S \rightarrow A$ on an *abstract level*. Because learning the policy in the ground RMDP is not an option, various routes can be taken in order to find $\hat{\pi}^*$. One can first construct *abstract value functions* and deduce a policy from that. One might also inductively learn the policy from optimal ground traces. Overall, one can distinguish between three main approaches.

The first one is the *inductive* route. The common factor of these approaches is that they use *ground samples* to learn abstract policies and value functions. The first paper introducing *Relational Reinforcement Learning* (RRL) (Džeroski et al., 1998) used a *relational regression tree* to learn an abstract Q-function from sampled traces. Later this approach was extended to the online *TG*-algorithm (Driessens et al., 2001), which devises an on-line structural regression tree learner. Related to this approach, Cole et al. (2003) employ the higher-order logic tree learner ALKEMY to learn Q-functions. The *relational instance based* (RIB) learner (Driessens & Ramon, 2003) and the *graph kernel-based* (Gärtner et al., 2003) approach developed in the RRL setting both inductively learn Q-functions from sampled traces. However, these approaches do not use general-purpose logical *languages*, but use distance- and kernel-based generalization using distance measures for structured representations. Recent work based on description logic used samples generated by a planning algorithm to inductively learn abstract policies directly and was later incorporated into an *approximated policy iteration* framework (Yoon et al., 2002; Fern et al., 2003). All these approaches are characterized by: (1) they implement different forms of *successive* abstraction, i.e. an abstract model is used to generate *biased* samples from the underlying RMDP and the abstract model is altered based on them, (2) they can – to some extent – be cast in the *learning from interpretations* approach known from *inductive logic programming* (ILP) (Muggleton & De Raedt, 1994) and (3) they aim at *approximative* abstractions.

The second route is *deductive*. The ground work for this direction was laid by Dietterich and Flann (1997) who showed that value backups can be generalized to multiple states in propositional MDPs by reversing the action operators. The first approach for RMDPs was *symbolic dynamic programming* (SDP) by Boutilier et al. (2001). In SDP a situation calculus abstraction of an RMDP is used and logical *regression* is employed within a *value iteration* algorithm on the r-level. In this way, value backups can be performed on abstract states, thereby updating sets of states of the underlying RMDP. The use of a very expressive logic did not permit an efficient computational algorithm yet. Recent work (Kersting et al., 2004) shows ReBel, a relational upgrade of the Bellman backup operator. ReBel uses a constraint logic language that is simpler than situation calculus and regression is employed to devise a *relational value iteration* algorithm. Overall, the deductive approaches can be characterized by: (1) they are *model-based*, (2) they aim at *exact* solutions, (3) logical reasoning methods such as regression and theorem proving are used to compute abstractions and (4) they are not sample-based, i.e. all computation is done at the r-level.

A very recent approach by Gretton and Thiébaux (2004) strikes a middle-ground between these approaches. It is related to SDP in the sense that a situation calculus specification of an RMDP is used. However, instead of fully implementing value iteration, the approach uses a number of logical regression steps after which the obtained abstraction level is used in an inductive way for further generalization.

Finally, there is the *modeling* route. Recent approaches show that the r-level can be modeled to define useful abstractions *a priori*. Kersting and De Raedt (2003) define the *Logical MDP* (LOMDP) which is an abstraction over RMDPs and uses Q-learning on this fixed, abstract model. Van Otterlo (2004) defines the *CARCASS* which is an RL-based abstraction for RMDPs. By learning a transition model, *prioritized sweeping* can be used over the logical abstraction to speed up learning. Morales (2003) introduces an a priori abstraction of RMDPs based on separate state and action spaces and uses Q-learning over the abstraction. Guestrin et al. (2003) uses *probabilistic relational models* and models *class-based* value functions assuming fixed relations. These approaches all *model* an abstraction over the underlying RMDP and use samples to learn parameters of these structures.

3. Challenges *of* RRL

When employing logical languages for abstraction over RMDPs a number of *structural* learning tasks naturally arises. To some extent, many of them can be cast in ILP terms. However, the intrinsic *probabilistic* nature of (R)MDPs, and also the notion of *concept drift* in RL approaches demand new techniques to deal with them. The growing field of *probabilistic logic learning* (De Raedt & Kersting, 2003) provides many new techniques for tackling these challenges. We will now mention some of the learning tasks for RRL and possible techniques.

1. Structural models. Relational abstractions can be used to model various substructures, and each of them can – in principle – be learned. Although Guestrin et al. (2003) used PRMs for modeling the domain, many more probabilistic relational models exist which have not been considered in the context of RRL. Relational upgrades of *Bayesian networks*, *hidden Markov models* and other probabilistic logic representation formalisms could be useful (cf. (De Raedt & Kersting, 2003)).

1a. Value functions. Abstract value functions can abstract *state* or *state-action* value functions. For example, $\mathtt{V(on(X,floor))}$ denotes that all states in which there is some block on the floor, get a value of 10. Some *relational regression* techniques have been used (e.g. in the inductive approaches). In general they can be upgraded versions of RL-based regression algorithms (e.g. (Driessens et al., 2001)) or adapted for use in RRL (e.g. (Driessens & Ramon, 2003)). Overall, it is necessary to develop relational regression algorithms that can handle *concept drift*.

1b. Action definitions. Action learning can be very useful to obtain an abstract model of the underlying MDP (Khardon, 1999; Pasula et al., 2004). For example, the action definition $\mathtt{on(X,Y),cl(X),cl(Z)} \xleftarrow{\mathtt{move(X,Y)}} \mathtt{cl(X),cl(Y),on(X,Z)}$ states that applying $\mathtt{move(X,Y)}$ can be performed if $\mathtt{X}$ and $\mathtt{Y}$ are both clear and that in the resulting state $\mathtt{X}$ is on $\mathtt{Y}$. In some domains, one can also consider learning *preconditions* only. Once an action model is learned, deductive techniques can be used to learn value functions. This combination with model-learning has not yet been explored. The associated reward function too may be learned.

1c. MDP Abstraction. Relational abstractions can also be defined over substructures of the underlying MDP itself, e.g. see the modeling approaches. For example, $\{\mathtt{cl(X),on(X,Z),cl(Y)},\langle\mathtt{move(X,Y),move(Y,X)}\rangle\}$ represents an abstract state with two abstract actions. The modeling approaches have defined such abstractions but they have not yet been learned.

1d. Policies. A policy assigns an action to each state. An abstract policy can consist of a set of rules, each abstracting a set of state-action pairs. Standard ILP can be used on optimal state-action pairs but in general the policy is induced from an abstract value function. In some cases this is a simple maximization process but the *learning from entailment* ILP setting (Muggleton & De Raedt, 1994) can be used as well. In general, most approaches use a strict (decision-list style) policy representation. An interesting approach would be to replace this by a set of (possibly overlapping) probabilistic rules.

2. Parameters. Tasks **1a**, **1b**, **1c** and **1d** deal with *structural* learning tasks. However, most structures contain parameters. The inductive approaches usually solve both problems together whereas the modeling approaches focus on the parameter estimation. By adding structure learning mechanisms to the latter one, iterative 2-phase learning algorithms can be developed separating structure and parameter learning, i.e. in an *approximated policy iteration* fashion. This would make it easier to analyze abstraction levels and use problem dependent learning algorithms for structure and parameters separately.

3. Hierarchy. In a hierarchical task, abstraction is not only performed on the structure of the current state, but also on the structure of the task. In this case, one can abstract *action sequences* (e.g. subpolicies) but also necessary subgoals needed to achieve the goal. Logical abstractions naturally allow parameterized subpolicies, something that happens on a more ad-hoc fashion in propositional approaches. Hierarchical approaches are also related to the connection between *learning and reasoning*. Using learned task hierarchies in *plan libraries* with corresponding reasoning mechanisms in logic-based agent architectures can be seen as bridging the gap between learning and reasoning (Van Otterlo et al., 2003). Hierarchical methods and the combination of learning and reasoning are open problems in the field of RRL.

4. Challenges *for* RRL

Next to the challenges *of* RLL there are the challenges *for* RRL. These arise in the new setting of (logical) abstraction over RMDPs.

4.1. A Theory of RRL

The development of theoretical insights has not kept pace with empirical work. There exists a number of experimental approaches, each targeting a specific learning task or domain. However, development of theories on how and why some methods work, are almost absent for RRL. Some work has started in the modeling approaches to identify useful concepts that can be transferred to the inductive approaches, such as *abstraction levels.* From a logical point of view, the deductive approaches are more amenable to formal analysis, but because the inductive approaches are based on the same semantics – the g-level – both may benefit.

4.1.1. Exploring Abstraction Levels

Depending on the complexity of the logic used, various *abstraction levels* can be defined, even over infinite domains, e.g. (Kersting et al., 2004). If the abstraction level is too general, we might introduce *partial observability* (Van Otterlo, 2004). Various logical connectives and quantifiers give more representational powers to abstraction levels – and are especially useful for the deductive approaches– but this in turn influences learning possibilities for the inductive approaches. The properties of the abstraction levels influence the (im)possibility of learning optimal policies.

Abstraction levels can be homogeneous or inhomogeneous (Givan et al., 2003; Kim & Dean, 2003). Current deductive approaches aim at exact solutions and keep their abstractions value-homogenous. However, abstractions obtained in inductive approaches might be more compact and more eligible for generalization to other domains. The connection to related work on *adaptive resolution* in propositional systems was only made by (Driessens et al., 2001) by upgrading a propositional tree learner based on state-splitting. Related work in model-free and model-based state-splitting can be exploited to develop similar techniques for relational representations. Splitting (and merging) state descriptions in relational languages, is more complex than in the propositional case though. Splitting in the joint state-action space is not considered much in the propositional case where usually the state space is partitioned and the action set is kept separate. For RMDPs it the joint space is more important because of variables connecting states and actions.

Comprehensibility of learned abstractions is one of the promises of RRL. Although relational rules can be more comprehensible than e.g. the weights of a neural network, there are some trade-offs: consider the following abstraction in the blocks world:

(1) $\mathtt{on(a,A), cl(a), on(b,B), on(X,b), cl(X), X \neq a}$
(2) $\mathtt{on(a,A), on(X,a), cl(X), on(b,B), cl(b), X \neq b}$
(3) $\mathtt{on(a,A), on(b,a), cl(b)}$
(4) $\mathtt{on(a,A), on(X,a), cl(X), on(b,B), cl(b)}$

The abstraction levels $\{1,2,3\}$ and $\{1,4\}$ are logically equivalent. From a *minimum description length* (MDL) point of view, the latter is preferred. However, most humans will overlook that in (4) variable X can unify with b. But if we have to visually inspect 100 rules or 10, less rules are preferred. So, although logical representations may yield *comprehensible* abstractions, there are important trade-offs to be considered.

The abstractions currently used are of the *aggregation*, or *averaging* (Gordon, 1995) type. So far, *exact* versions of this are used. However, more robust (and possibly more compact) abstractions could be explored using *soft aggregation*, where the abstractions are overlapping and aggregation is probabilistic in nature.

Yet another dimension is *action abstraction.* Because action abstractions are usually (syntactically) dependent on state or precondition descriptions, state and action abstractions are more tightly connected than in propositional representations. The consequences of this for modeling abstraction levels has not been addressed explicitly.

4.1.2. Proofs of Convergence

Convergence analysis is an important issue. So far, this issue has not been approached for RRL. Relational abstraction is essentially a form of *function approximation*, for which – especially in model-free learning – convergence guarantees are scarce. However, given the fact that general relational abstraction can be viewed as *averaging* (Gordon, 1995) useful bounds might be computed for fixed abstraction levels. However, given the fact that most inductive approaches are of the *adaptive resolution* kind, i.e. the abstraction level changing during the learning process, it is largely open how to approach convergence in these contexts. In general we have to distinguish between *structural* convergence, i.e. obtaining a stable abstraction level and *value* convergence within an abstraction level. Not converging on the structural level does not have to exclude computing an optimal policy (Kersting et al., 2004). For the deductive approaches one could use *stochastic bisimulation* approaches (Givan et al., 2003) to asses the difference of the r-level to the g-level. The use of *action abstraction* in general breaks many existing convergence proofs and is perhaps the most important open problem.

4.1.3. Complexity Trade-Offs

The trade-off between more powerful abstractions and the increased cost of manipulating relational structures should be investigated. For some classes of problems *propositionalization* might be more efficient in terms of computation, memory or sample complexity. It is interesting to study how the relation between r-level and g-level changes if the latter grows, possibly to infinite size. Furthermore, using more expressive logics such as situation calculus enables more powerful abstraction (lower space complexity) but it comes with an increased cost of manipulating and learning them. Comparing propositional and first-order languages in terms of MDP-specific criteria, such as number of episodes, number of value backups and the relative sizes of value functions and policies are interesting and mostly absent. For the inductive approaches the number of needed samples is important, whereas in the deductive approaches the cost of manipulating expressions is an important issue (e.g. theorem proving). Combinations of deduction and induction such as (Gretton & Thiébaux, 2004) are interesting in this respect. Furthermore, being able to *reason* about experience might help solve *partially observable* problems. These kind of problems have not been approached in RRL yet.

4.2. Values vs. Policies

The overall goal RRL is to learn an abstract policy $\hat{\pi}^*$. All current methods are *value-based* [1]. However, whereas the relational structure of the value function can be large (or infinite), a corresponding optimal policy can be very compact, by generalizing over objects *and* values. As an example, consider (part of) the value function for the goal $\texttt{on(a,b)}$ in the blocks world as shown in figure 1. The structure of the value function is relatively complex, and might need an infinite number of abstract states. However, even in infinite worlds, an optimal policy can be stated as the very short decision list from section 2.2. Additionally, it can be learned from a small part of the value function. Recent work has shown that for induction of finite policies *background knowledge* can actually be a *necessity*, and not a mere *feature* (Kersting et al., 2004). The use of background knowledge enables new ways of abstraction and it has not been analyzed explicitly yet.

[1]even the work by e.g. (Yoon et al., 2002) that uses planning heuristics for sampling

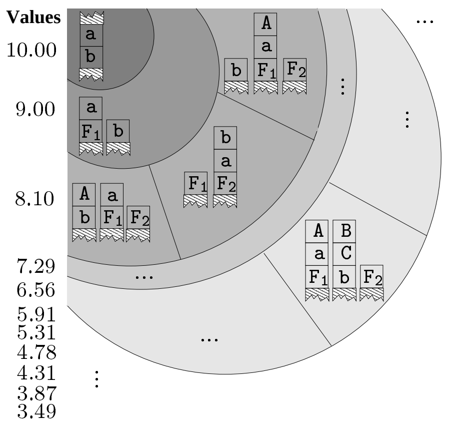

Figure 1. Parts of the abstract value function for the goal $\texttt{on(a,b)}$ after 10 iterations (Kersting et al., 2004). F_i can be a block or the floor.

Optimal value functions may have infinite range and can be heavily *shattered*. This creates difficulties especially for model-free learners such as RRL. But also deductive approaches are hindered by the explosion of the value partition. For that reason, it is also interesting to consider *direct policy* learning. There are at least three interesting directions to explore: (1) Upgrading *policy gradient* techniques to the relational case seems a promising direction. Related are *generalized expectation maximization* (GEM) approaches. In both cases however, the problem of how to *parameterize* abstract policies to derive necessary gradients should be solved. (2) The use of *evolutionary algorithms* (EA) has proved very efficient in standard RL, e.g. evolution of neural networks is a powerful way of obtaining neural networks representing control policies. Also related is *genetic programming.* (3) As was done for learning first-order action models (Pasula et al., 2004) and policies (Yoon et al., 2002) *supervised* learning abstractions can be done using ILP methods. For this direction, suitable contexts for obtaining optimal samples or traces should be explored. It is interesting to analyze the policy vs. value trade-off in the relational setting and to also see whether approximative and probabilistic approaches are the answer to shattered value functions.

4.3. Exploring Generalization

Because relational formalisms abstract over objects, it is tempting to learn policies in small domains and apply them to larger domains. However, in probabilistic domains, due to the domain *size* this might not work. Guestrin et al. (2003) give insight into a restricted case but in general, it is an open problem to determine when generalization can be applied.

Consider for example a blocks world with probabilistic actions. Assume that one of the outcomes of `move(X, Y)` is that block `X` is moved onto somewhere else (i.e. not on `Y`). An abstract n-steps-to-go value function generalizes no matter how many blocks there are. However, if we consider a blocks world where this outcome is to move onto *some other block*, it depends on how many blocks there are and how many stacks, i.e. the probability distribution over the next states depends on this (and so may be the reward). In such a context, the value function is dependent on the number of blocks. In some contexts, even the policy may depend on the domain size. In other words, *upgrading* learned value functions or even policies to larger domains might not always be valid. Insight is needed into when *upgrading* from small problem instances (RMDPs) is applicable. As shown by Kersting et al. (2004), sometimes *background knowledge* might solve the policy induction problem, but also here insight is needed how and when to apply this. Overall, changing the semantics, i.e. the `g`-level, by adding domain elements might be harmful to the `r`-level and analysis is needed, especially for the infinite case.

4.4. Proofs of Soundness & Completeness

For the `r`-level, many different logical formalisms can be (and have been) employed, such as Horn logic and description logics. It is important to provide *mappings* between these frameworks for comparison purposes. Also important are *representation theorems*: when inducing or defining an abstraction level, one has to consider the relation between `g`-level and `r`-level. For the deductive approaches, *soundness* of the reasoning process and simplification of expressions is important, e.g. ReBel uses domain constraints to enforce this. The modeling approaches too have to ensure that the models they define corresponds to a correct underlying `g`-level. For the inductive approaches, *completeness* is more an issue. Induced structures should be rich enough to model the underlying `g`-level.

4.5. More Real-World Applications

A final important issue is convincing (real-life) *applications*. The blocks world is used in most studies, but it should be viewed upon as – analogously to genetics – a *Drosophila*. Other applications are logistics domains and computer games such as Digger, Tetris and FreeCraft. In order to more fully show the benefits of relational representations for MDPs, one should solve larger, problems for which existing, propositional algorithms are insufficient.

Taking inspiration from the field of *probabilistic logic learning*, webmining and bio-informatics applications are challenging domains due to their size and their intrinsically relational structure. RRL upgrades of RL *Webspiders* (Rennie & McCallum, 1999) would be very interesting. Other domains in which learning and reasoning could be combined are interesting, for example in *multi-agent* contexts. Here, many interesting problems can be explored, such as communication and cooperation. Agents in these contexts are often modeled in terms of first-order languages, whereas usually behavior learning is performed using propositional languages or ad-hoc methods are applied to connect learning to the structural representations of knowledge.

5. Conclusion

Relational reinforcement learning aims at tackling one of the central open questions in machine learning and artificial intelligence, namely reasoning and acting in richly structured domains. At the heart of all challenges arising in relational reinforcement learning lies the *trade-off between exploration and exploitation of abstraction levels*. To accumulate a lot of reward, the learning system must *exploit* the best experienced abstraction level. However, to discover better action selections for the future, it must *explore* new abstraction levels. For these matters we need a *theory of RRL*.

Acknowledgements

The authors thank Luc De Raedt for encouragement for writing this paper and interesting discussions on the topic. This research was supported by the European Union IST programme, contract no. FP6-508861, *APrIL II*. Martijn Van Otterlo was supported by a Marie Curie fellowship at DAISY, HPMT-CT-2001-00251.

References

Boutilier, C., Reiter, R., & Price, B. (2001). Symbolic dynamic programming for first-order MDP's. *Proc. of IJCAI'01.*

Cole, J., Lloyd, K., & Ng, K. (2003). Symbolic learning for adaptive agents. *Proc. of the Annual Partner Conference, Smart Internet Technology Cooperative Research Centre.* http://csl.anu.edu.au/jwl/crc_paper.pdf.

De Raedt, L., & Kersting, K. (2003). Probabilistic logic learning. *ACM-SIGKDD Explorations, special issue on Multi-Relational Data Mining, 5.*

Dietterich, T. G., & Flann, N. S. (1997). Explanation-based learning and reinforcement learning: a unified view. *Machine Learning, 28*, 169–210.

Driessens, K., & Ramon, J. (2003). Relational instance based regression for relational reinforcement learning. *Proc. of ICML-2003.*

Driessens, K., Ramon, J., & Blockeel, H. (2001). Speeding up relational reinforcement learning through the use of an incremental first order decision tree algorithm. *Proc. of ECML'01.*

Džeroski, S., De Raedt, L., & Blockeel, H. (1998). Relational reinforcement learning. *Proc. ICML'98* (pp. 136–143). Morgan Kaufmann.

Fern, A., Yoon, S., & Givan, R. (2003). Approximate policy iteration with a policy language bias. *Proc. of NIPS'03.*

Finney, S., Gardiol, N., Kaelbling, L., & T.Oates (2002). The thing that we tried didn't work very well: Deictic representations in reinforcement learning. *Proc. of UAI'02.*

Gärtner, T., Driessens, K., & Ramon, J. (2003). Graph kernels and gaussian processes for relational reinforcement learning. *Proc. of ILP'03.*

Givan, R., Dean, T., & Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, *147*, 163–223.

Gordon, G. (1995). Stable function approximation in dynamic programming. *Proc. of ICML'95.*

Gretton, C., & Thiébaux, S. (2004). Exploiting first-order regression in inductive policy selection. *Proc. of UIA'04.*

Guestrin, C., Koller, D., Gearhart, C., & Kanodia, N. (2003). Generalizing plans to new environments in relational MDPs. *Proc. of IJCAI'03.*

Kaelbling, L., Oates, T., Hernandez, N., & Finney, S. (2001). Learning in worlds with objects. *The AAAI Spring Symposium.*

Kersting, K., & De Raedt, L. (2003). Logical Markov decision programs. *Proc. of the IJCAI'03 Workshop on Learning Statistical Models of Relational Data.*

Kersting, K., Van Otterlo, M., & L.De Raedt (2004). Bellman goes relational. *Proc. of ICML'04.*

Khardon, R. (1999). Learning to take actions. *Machine Learning*, *35*, 57–90.

Kim, K.-E., & Dean, T. (2003). Solving factored mdps using non-homogeneous partitions. *Artificial Intelligence*, *147*, 225–251.

Morales, E. (2003). Scaling up reinforcement learning with a relational representation. *Proceedings of the Workshop on Adaptability in Multi-Agent Systems at AORC'03, Sydney.*

Muggleton, S., & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, *19 & 20*, 629–680.

Pasula, H., Zettlemoyer, L., & Kaelbling, L. (2004). Learning probabilistic relational planning rules. *Proc. of ICAPS'04.*

Rennie, J., & McCallum, A. (1999). Using reinforcement learning to spider the web efficiently. *Proc. of ICML'99.*

Sutton, R., & Barto, A. (1998). *Reinforcement learning: an introduction.* Cambridge: The MIT Press.

Van Otterlo, M. (2002). Relational representations in reinforcement learning: Review and open problems. *Proc. of the ICML'02 Workshop on Development of Representations.*

Van Otterlo, M. (2004). Reinforcement learning for relational MDPs. *Machine Learning Conference of Belgium and the Netherlands (BeNeLearn'04).*

Van Otterlo, M., Wiering, M., Dastani, M., & Meyer, J.-J. (2003). A characterization of sapient agents. *Proc. of the International Conference on the Integration of Knowledge Intensive Multi-Agent Systems KIMAS'03.*

Yoon, S., Fern, A., & Givan, R. (2002). Inductive policy selection for first-order MDPs. *Proc. of UAI'02.*