

Bellman goes Relational (extended abstract)¹

Martijn Van Otterlo^{a,b} Kristian Kersting^b

Luc De Raedt^b

^a University of Twente, Dept. of Computer Science, HMI,
P.O. Box 217, 7500 AE Enschede, The Netherlands

^b University of Freiburg, Machine Learning Lab,
Georges-Koehler-Allee 079, 79110 Freiburg, Germany

Abstract

We introduce REBEL, a *relational Bellman update operator* that can be used for Markov Decision Processes in – possibly infinite – relational domains. Using REBEL we develop a *relational value iteration* algorithm.

1 Relational Markov Decision Processes

Many *reinforcement learning* (RL) and *dynamic programming* techniques have been developed for solving *Markov Decision Processes* (MDP). Until recently, the representation of MDPs was limited to states and actions that were discrete symbols or factored into propositions or attribute-value pairs. Recent approaches have shown that this representation can be upgraded towards *relational* (or *first-order*) representations – thereby enabling the use of first-order logical languages to represent states and actions in terms of *objects* and *relations* such as `inRoom(robot,kitchen)` and `driveTo(library)`. Although some approaches have appeared for solving *relational* MDPs (RMDP) (e.g. *relational RL* [2]), not much work has considered *exact, model-based* solution techniques. The only example is the work by Boutilier et al. (2001) which employs *situation calculus* for modeling an RMDP which is then solved by a *value iteration* algorithm. However, because of the complexity of the language, the algorithm was not fully automated yet, e.g. the *simplification* of expressions obtained is done manually. Here we show that by using a restricted language, the simplification is computationally feasible and we develop a fully automatic value iteration algorithm. Another aim of the paper is to give some insights into the framework of relational RL.

2 Relational Bellman Operator

The first step is the introduction of a logical formalism to specify MDPs over relational domains. A constraint logic programming language is used to define *four*

¹The full paper appeared in the Proceedings of the 21st International Conference on Machine Learning (*ICML'04*) [July 4–8, 2004, Banff, Canada].

ingredients: **(1) abstract states** are conjunctions of logical atoms aggregating sets of concrete states. **(2) abstract actions** are similar to first-order, probabilistic STRIPS operators. **(3) abstract rewards** form a reward model describing state-based rewards. **(4) integrity constraints** describe domain constraints.

Based on this formalism a relational upgrade of the *Bellman backup operator* can be defined. Starting from an initial state value function, each value backup step consists of the following operations: **(1)** For each action rule we employ *regression* to compute **weakest preconditions** (wp) of abstract states and abstract action rules. **(2)** For each of these wps derived by some action rule, we compute a **Q -value**, defining a Q -rule. **(3)** Each Q -rule defines a *partial* Q -value, because it is based on one outcome of the action. All the Q -rules for different outcomes of the same action are **combined** into a complete Q -rule. **(4)** Because $V(s) = \max_a Q(s, a)$ we **maximize** over the set of Q -rules to get an equivalent state-value function. Overall, these four steps refine the current value partition V^t into V^{t+1} .

Relational value iteration can now be implemented by repeated application of the described operations. Starting from the state-value function defining goal state rewards (V^0) we compute the series $V^1 \dots V^n$ until some stopping criterium.

3 Experiments

To show that REBEL is able to fully automatically implement relational value iteration, we report on three experiments. In the first, the goal is $\text{cl}(\mathbf{a})$ in a probabilistic blocks world. An optimal value function is calculated for almost 60 million ground states. This experiment also shows an interesting result: in a relational setting, *value iteration is not guaranteed to converge on the structural level*. An infinite number of abstract states can be required. However, often the corresponding (optimal) policy can be extracted after a couple of iterations if *background knowledge* is supplied. This is an important observation given the fact that usually background knowledge is seen as a feature, but it turns out to be a necessity in some cases. The second experiment computes the optimal value function for 10 blocks with goal $\text{on}(\mathbf{a}, \mathbf{b})$. It has been reported that this is a difficult problem for model-free learners (e.g. [2]) that do not get optimal policies on average even for 5 blocks. The third experiment is the *logistics* domain described in [1]. We show that it is possible to automatically compute the optimal value function, due to the more restricted language employed by REBEL. Furthermore, this experiment shows that for some domains REBEL can converge on both the structural and the value level unlike in the first two experiments.

References

- [1] C. Boutilier, R. Reiter, and B. Price. Symbolic dynamic programming for first-order MDP's. In *Proceedings of IJCAI'01*, 2001.
- [2] S. Džeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine Learning*, 43:7–52, 2001.