Künstl Intell (2015) 29:363–368
DOI 10.1007/s13218-015-0386-8

TECHNICAL CONTRIBUTION

Statistical Relational Artificial Intelligence: From Distributions through Actions to Optimization

Kristian Kersting[1] · Sriraam Natarajan[2]

Received: 26 February 2015 / Accepted: 25 June 2015 / Published online: 8 July 2015

Abstract Statistical Relational AI—the science and engineering of making intelligent machines acting in noisy worlds composed of objects and relations among the objects—is currently motivating a lot of new AI research and has tremendous theoretical and practical implications. Theoretically, combining logic and probability in a unified representation and building general-purpose reasoning tools for it has been the dream of AI, dating back to the late 1980s. Practically, successful statistical relational AI tools enable new applications in several large, complex real-world domains including those involving big data, natural text, social networks, the web, medicine and robotics, among others. Such domains are often characterized by rich relational structure and large amounts of uncertainty. Logic helps to faithfully model the former while probability helps to effectively manage the latter. Our intention here is to give a brief (and necessarily incomplete) overview and invitation to the emerging field of Statistical Relational AI from the perspective of acting optimally and learning to act.

1 Introduction: From Distributions to Actions

There are good arguments for an intelligent agent, who makes decisions about how to act in a complex world, to model its uncertainty; it cannot just act pretending that it knows what is true. However, an intelligent agent also needs to be able to reason about individuals—objects, entities, things—and about relations among the individuals. Both aspects have been studied mostly separately within AI, with propositional models for uncertainty often in terms of features and random variables, and rich logical languages for reasoning about relations that ignore the uncertainty inherent dealing with the real world. More precisely, AI has evolved from being skeptical of using probability to embracing probability, in particular in terms of probabilistic graphical models [28, 39], which exploit useful independencies and have revolutionized AI. The independencies given by graphs are natural, provide structure that can enable efficient reasoning and learning, and allow us to model domains of enormous complexity. Many AI problems arising in a wide variety of fields such as diagnosis, network communication, computer vision, and robotics can elegantly be encoded and solved using probabilistic graphical models. However, it must be mentioned that graphs are not enough, and we need logic in order to exploit the inherent structure of complex domains instead of explicitly enumerating everything. Consequently, statistical relational learning (SRL) [17] has emerged as a new subfield of AI. It investigates relational probabilistic models—collections of templates such as weighted clauses that are instantiated several times to construct a ground probabilistic graphical model—and has been proven successful in many AI tasks. Exploiting symmetries that may abound in the induced ground models, one can even speed up inference and learning [2, 23, 27, 53].

Here, we argue the same holds when moving beyond distributions towards acting optimally based on Markov decision processes (MDPs) and solving them using dynamic programming [5] and reinforcement learning

✉ Kristian Kersting
kristian.kersting@cs.tu-dortmund.de

Sriraam Natarajan
natarasr@indiana.edu

1 TU Dortmund University, Dortmund, Germany

2 Indiana University, Bloomington, USA

(RL) [50]. We review and illustrate the core ideas for lifting them to relational domains. Due to space restrictions, the overview is necessarily incomplete and brief—we apologize to anyone whose work we have omitted—and also does not introduce standard MDPs and RL; for that we refer to [5, 50].

We start off with relational variants of MDPs and solving them using dynamic programming. Then we move on to relational reinforcement learning. We conclude by providing a general perspective on relational AI via optimization.

2 Relational Markov Decision Processes

The initial work on acting in complex, uncertain worlds was on representations, in terms of the event calculus [40] and the situation calculus [3, 41]. This is challenging because in order to plan, an agent needs to be concerned about what information will be available for future decision. These approaches combined perception, action and utility to form first-order variants of fully-observable and partially-observable MDPs[1] [5, 21, 50]. Recent work has concentrated on how to plan with such representations either for the fully observable case [6, 7, 26, 35, 45, 55] or the partially observable case [36, 48, 56]. At the heart of all these approaches is symbolic dynamic programming (SDP), a generalization of the dynamic programming technique for solving propositional Markov decision processes [5]. SDP exploits the symbolic structure in the solution of relational and first-order logical Markov decision processes (FOL-MDP) through a lifted version of dynamic programming [47]. It constructs a minimal logical partition of the state space required to represent the value function induced by the MDP, which maps each state of the environment to a single number, the value, indicating the intrinsic desirability of that state.

To give a flavour of FOL-MDPs, i.e., of how one encodes rewards, states, and actions relationally, consider the following logistics example in case form [6, 45]. We want to have a box (not a particular one, just a box) in Paris. We receive a reward of 10 if a box is in Paris and otherwise 0:

$$(\textbf{Reward})\ \textbf{If}\ \mathit{BoxIn}(b, \mathit{Paris})\ \textbf{then}\ 10\ \textbf{else}\ 0$$

where the placeholder resp. logical variable b for a box is existentially quantified saying that we receive the rewards whenever there is at least one box in Paris[2]. If this is the case, we have reached a terminal state and the task ends.

Now we encode the actions. The actions the agent can perform include load, unload, and drive. To encode them, one can use probabilistic STRIPS-like planning operators [15, 38]. For instance, the action to load a box b onto a truck t in city c can be defined as:

$$\begin{aligned}&(\textbf{Action})\,\mathit{unload}(b,t,c):\\&\textbf{Success Prob.:if}\,(\mathit{TruckIn}(t,c)\wedge \mathit{BoxOn}(b,t))\,\textbf{then}\,0.9\,\textbf{else}\,0\\&\textbf{Add Effects on Success:}\{\mathit{BoxIn}(b,c)\}\\&\textbf{Delete Effects on Success:}\{\mathit{BoxOn}(b,t)\}\end{aligned}$$

The interpretation is that (logical) atoms in the add effects describe what is added to the world as an effect of the action and atoms in the delete effects describe what is deleted from the world. Thus, if there is e.g. a box Box on a truck Truck in a city London, then with probability 0.9 we remove $\mathit{BoxOn}(\mathrm{Box}, \mathrm{Truck})$ from the current state and add $\mathit{BoxIn}(\mathrm{Box}, \mathrm{London})$ when performing action $\mathit{unload}(\mathrm{Box}, \mathrm{Truck}, \mathrm{London})$.

Finally, the relational state are implicitly encoded as any logical interpretation using the predicates and constants appearing in the rwards and action encodings.

While the exact logical representation differ among the different FOL-MDP formalisms, they all share more or less this kind of logical representation of rewards, probabilities, and values. Now, the agent's task is to find a policy for action selection in each relational state that maximizes its reward over the long term to reach a terminal state. To do so, it derives a logically parameterized plan that can be applied to any situation. For instance, it is possible to infer that in order to get box b to be in city c, the agent drives a truck to the city of b loads b onto the truck, drives the truck to city c, and finally unloads the box b_1 in c. This is achieved through the operations of first-order decision-theoretic regression and symbolic maximization [44, 45]. While the details of these operations are beyond the scope of the present short introduction, one should note that the operations are exactly the lifted versions of the traditional dynamic programming solution to Markov decision processes.

In our running example, let $\mathit{Val}(t)$ be the t stages-to-go value function. For $t \rightarrow \infty$, this results in the value function of a policy [5], which encodes the expected sum of discounted rewards accumulated while executing that policy when starting from state s. $\mathit{Val}(0)$ is the immediate

[1] Here, at each time step, a system is in some state s, and the agent may choose any action a that is available in state s. Then, depending on a, the system moves stochastically into state s', and the agent receives a reward $R_a(s, s')$.

[2] If there are more than one box in Paris, we still get only a reward of 10 due to the existential quantifier of b. That is, the formulas in the reward can be viewed as logical queries that are either true or false. If they are true, we get the reward.

reward as defined above. $Val(1)$ for the goal of getting box b to Paris can be computed using SDP to be:

$$\begin{aligned}&\textbf{If } BoxIn(b, Paris) \text{ then } Val(1) = 19\\&\textbf{else if} \exists t\, TruckIn(t, Paris) \wedge BoxOn(b, t) \textbf{ then } Val(1) = 9\\&\textbf{else } Val(1) = 0.\end{aligned}$$

The logical partition directly follows our intution. If $BoxIn(b, Paris)$ holds in a state then we get the highest reward since it is a terminal state. If not, we regress $\exists t\, TruckIn(t, Paris) \wedge BoxOn(b, t)$ by unifying the add effect of the *unload* action with the terminal state $BoxIn(b, Paris)$ and taking the precondition part of the success probability part of *unload*. Otherwise we unify with the empy 0 part of the reward partition resulting in $Val(1) = 0$ case. While the exact logical representation of the value function differ among the different FO-MDP formalisms, the essence of logical decision list representation is shared among all of them. Based on $Val(1)$, $Val(2)$ can be computed to be:

$$\begin{aligned}&\textbf{If } BoxIn(b, Paris) \textbf{ then } Val(2) = 27.1\\&\textbf{else if } \exists t\, TruckIn(t, Paris) \wedge BoxOn(b, t) \textbf{ then } Val(2) = 17.1\\&\textbf{else if } \exists c, t\, BoxOn(b, t) \wedge TruckIn(t, c) \textbf{ then } Val(2) = 8.1\\&\textbf{else } Val(2) = 0.\end{aligned}$$

After sufficient iterations, the t-stages-to-go value function converges. The key features to note are the state and action abstraction in the value and policy representation that are afforded by the first-order specification and solution of the problem. That is, this solution does not refer to unimportant domain individuals such as *Berlin*, *London*, and specific boxes and truck, but rather it provides a solution in terms of placeholders such as b, t, and c for *all possible domain individual instantiations* where possible. Classical dynamic programming techniques, which enumerate all states and actions, could never solve these problems for large domain instantiations, while domain-independent lifted solution can do so due to the power of logic. Moreover, the power of logic can also help in propositional problems if e.g. there are both continuous state and observation spaces [57]; while there may be an infinite number of possible observations, there are only a finite number of observation partitionings that are relevant for optimal decision-making when a finite, fixed set of reachable belief states are known.

Since the basic symbolic dynamic programming approach, a variety of exact algorithms have been introduced to solve MDPs with relational (RMDP) and first-order (FOL-MDP) structure. *First-order value iteration* (FOVIA) [19, 22] and the *relational Bellman algorithm* (ReBel) [26] are value iteration algorithms for solving RMDPs. In addition, *first-order decision diagrams* (FODDs) have been introduced to compactly represent case statements and to permit efficient application of symbolic dynamic programming operations to solve RMDPs via value iteration and policy iteration [55]. Furthermore, a class of linear-value approximation algorithms have been introduced to approximate the value function as a linear combination of weighted basis functions. *First-order approximate linear programming* (FOALP) [45] directly approximates the FOMDP value function using a linear program. Other heuristic solutions for instance induces rule-based policies from sampled experience in small-domain instantiations of RMDPs and generalizes these policies to larger domains [13]. In a similar vein, Gretton and Thiebaux [18] used the action regression operator in the situation calculus to provide the first-order hypothesis space for an inductive policy learning algorithm. Recently, Lang and Toussaint [30] and Joshi et al. [20] have shown that successful planning typically involves only a small subset of relevant individuals respectively states and how to make use of this fact to speed up symbolic dynamic programming significantly. Powell [42] described a framework for reasoning in such domains by counting over the number of individuals.

In general, FOL-MDPs solvers have been successfully applied in decision-theoretic planning domains typically used within the International Planning Competition [14, 45]. The FOALP system [45] was runner-up at the probabilistic track of the 5th International Planning Competition (IPC-6). Related techniques have been used to solve path planning problems within robotics and instances of real-time strategy games, Tetris, and Digger [8, 31, 33, 49]. Recently, there have also been approaches on extending probabilistic relational models for making decisions [7, 35].

3 Relational Reinforcement and Imitation Learning

In many situations, however, the agent does not have knowledge about the underlying (relational) (PO)MDP given. In turn, the agent has to learn what to do through trial-and-error interactions with an uncertain, relational environment. This is called relational reinforcement learning [51, 54].

Consider household robots, which just need to be taken out of their shipping boxes, turned on, and then do some cleaning work. This "robot-out-of-the-box" has inspired research in robotics as well as in machine learning and artificial intelligence. Without a compact knowledge representation that supports abstraction by and unification of logical placeholders, and generalization of previous experiences to the current state and potential future states, however, it seems to be difficult—if not hopeless—to explore a new home in reasonable time. There are simply too many individuals a household robot may deal with such as doors, plates, boxes and water-taps. Using relational representations, the robot can learn about how doors work, how taps work, and how to wash dishes, even before it is delivered, and can then observe the particular individuals in the house, and continue learning. Indeed, there will still be uncertainty, but due to abstraction from individuals and by using general relations, the robot can already operate when delivered.

To deal with the obvious "curse of dimensionality" encountered—there are simply too many potential state-action pairs—a number of model-free [9–12, 24, 43] as well as model-based [31, 33, 38] *relational reinforcement learning* approaches have been developed. The core idea is to lift classical reinforcement learning approaches by using relational machine learning techniques that can abstract from individuals using logical queries. Relational learners such as relational regression trees, graph kernels, relational gradient boosting and inductive logic programming methods are used either to learn relational value functions, relational policies or even R(PO)MDPs.

For instance, triggered by the idea that finding many rough rules of thumb of how to change the way to act can be a lot easier than finding a single, highly accurate policy, one can represent a policy—mapping relational states to actions—as a weighted sum of relational regression trees grown in an stage-wise optimization in order to maximize the expected return [24]. Each regression tree can be viewed as defining several new feature combinations, one corresponding to each path in the tree from the root to a leaf. However, rather than using attribute-value or threshold tests in an inner node of the tree, a relational regression tree employs logical queries such as $BoxIn(b, c)$. This way, interactions among (sets of) states and actions are introduced only as needed, so that the potentially infinite search space is not explicitly considered.

Moreover, the inherent generalization of learned knowledge in the relational representation has profound implications on the exploration strategy: what in a propositional setting would be considered a novel situation and worth exploration may in the relational setting be an instance of a well-known context in which exploitation is promising [32, 33]. For instance, after having opened one or two water taps in a kitchen, the household robot can expect other water-taps in kitchens to behave similarly. Thus, the priority for exploring water-taps in kitchens in general is reduced and not just the encountered ones. Moreover, the information gathered is likely to carry over to water-taps in other places such as laundries, since we have learned something about water tapes in general. Without extensive feature engineering this would be difficult—if not impossible—in a classical, propositional setting. We would simply encounter a new and therefore unexplored water-tap again and again.

Finally, the same upgrading approach has been proven to be successful for relational imitation learning [34], where the agents does not even interact with the environment but has to learn what to do by observing a teacher in action. We do not present the details here, but refer to the paper.

4 From Actions to Optimization

Robots start to perform everyday manipulation tasks, such as cleaning up, setting a table or preparing simple meals. To do so they must become much more knowledgeable than they are today. Everyday tasks are specified vaguely and the robot must therefore infer what are the appropriate actions to take and which are the appropriate individuals involved in order to accomplish these tasks. This can only be done if the robot has access to general world knowledge [4, 52] grounded in sensor readings [1, 29, 37] turned into actions by relational methods.

Ultimately, however, Statistical Relational AI—the science and engineering of making intelligent machines acting in noisy worlds composed of objects and relations among the objects—has to move beyond distributions and actions towards optimization. The robot has to be able to solve several AI tasks jointly involving a multitude of interacting mathematical models like SAT, CSP, (PO)MDPs, probabilistic graphical models, (integer) linear and quadratic programs. Thus, we have to lift the declarative "Model + Solver" paradigm that was and is prevalent in AI [16] to the relational level: instead of outlining how a solution should be computed, we specify the problem using some high-level relational modeling language and solve it using (several interacting) general solvers. A recently proposed framework for relational linear programming [25] indicates that this dream of relational optimization is not insurmountable[3]. Moving along similar directions has the potential to make AI programming several times easier and more powerful than current approaches and is a step towards achieving the grand challenge of automated programming as sketched by Jim Gray in his Turing Award Lecture.

Acknowledgments The authors thank the anonymous reviewers for their feedback. They are also grateful to all the people who

[3] Likewise, for solving FOL-MPDs, Sanner and Boutilier [46] used approximate linear programming with relational constraints represented via case statements. This approach scaled to problems of previously prohibitive size by avoiding grounding and is indeed close in spirit to relational linear programming.

contributed to the development of Statistical Relational AI, in particular to Scott Sanner and David Poole for previous and current joint efforts in writing introductions and overviews on statistical relational AI and symbolic dynamic programming, parts of which grew into the present paper. KK also likes to acknowledge the supported by by the German Science Foundation (DFG), KE 1686/2-1, within the SPP 1527 "Autonomous Learning", and SN the support of Army Research Office (ARO) grant number W911NF-13-1-0432 under the Young Investigator Program.

References

1. Abdo N, Kretzschmar H, Spinello L, Stachniss C (2013) Learning manipulation actions from a few demonstrations. In: 2013 IEEE international conference on robotics and automation (ICRA), pp 1268–1275
2. Ahmadi B, Kersting K, Mladenov M, Natarajan S (2013) Exploiting symmetries for scaling loopy belief propagation and relational training. Mach Learn 92(1):91–132
3. Bacchus F, Halpern JY, Levesque HJ (1999) Reasoning about noisy sensors and effectors in the situation calculus. Artificial Intelligence 111(1–2):171–208 (1999). URL:http://www.lpaig.uwaterloo.ca/fbacchus/on-line.html
4. Beetz M, Jain D, Mösenlechner L, Tenorth M (2010) Towards performing everyday manipulation activities. Robot Auton Syst 58(9):1085–1095
5. Bellman RE (1957) Dynamic programming. Princeton University Press, Princeton
6. Boutilier C, Reiter R, Price B (2001) Symbolic dynamic programming for first-order MDPs. In: Nebel B (ed) Seventeenth international joint conference on artificial intelligence (IJCAI-01). Morgan Kaufmann, Seattle, USA, pp 690–700
7. Van den Broeck G, Thon I, van Otterlo M, De Raedt L (2010) DTProbLog: A decision-theoretic probabilistic prolog. In: Proceedings of the AAAI conference on artificial intelligence (AAAI 2010)
8. Driessens K, Dzeroski S (2002) Integrating experimentation and guidance in relational reinforcement learning. In: Proceedings of ICML, pp 115–122
9. Driessens K, Dzeroski S (2005) Combining model-based and instance-based learning for first-order regression, pp 193–200
10. Driessens K, Ramon J (2003) Relational instance based regression for relational reinforcement learning, pp 123–130
11. Driessens K, Ramon J, Gärtner T (2006) Graph kernels and Gaussian processes for relational reinforcement learning. Mach Learn J 64(1–3):91–119
12. Džeroski S, De Raedt L, Driessens K (2001) Relational reinforcement learning. Mach Learn J 43(1/2):7–52
13. Fern A, Yoon S, Givan R (2003) Approximate policy iteration with a policy language bias. In: NIPS-2003. Vancouver
14. Fern A, Yoon S, Givan R (2006) Approximate policy iteration with a policy language bias: Solving relational markov decision processes. J Artif Intell Res (JAIR) 25:75–118
15. Fikes RE, Nilsson NJ (1971) STRIPS: a new approach to the application of theorem proving to problem solving. Artif Intell 2(3–4):189–208
16. Geffner H (2014) Artificial intelligence: from programs to solvers. AI Commun. 27(1):45–51
17. Getoor L (2007) In: Taskar B (eds.) An introduction to statistical relational learning. MIT Press, Cambridge
18. Gretton C, Thiebaux S (2004) Exploiting first-order regression in inductive policy selection. In: UAI-04, pp 217–225
19. Hölldobler S, Skvortsova O (2004) A logic-based approach to dynamic programming. In: In AAAI-04 Workshop on Learning and planning in MDPs, pp 31–36. Menlo Park, CA
20. Joshi S, Kersting K, Khardon R (2010) Self-taught decision theoretic planning with first order decision diagrams. In: Proceedings of ICAPS-10
21. Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. Artif Intell J 101(1–2):99–134
22. Karabaev E, Skvortsova O (2005) A heuristic search algorithm for solving first-order MDPs. In: UAI-2005, pp 292–299
23. Kersting K (2012) Lifted probabilistic inference. In: Proceedings of the 20th European conference on artificial intelligence (ECAI), pp 33–38. **(Invited Talk at the Frontiers of AI Track)**
24. Kersting K, Driessens K (2008) Non-parametric policy gradients: A unified treatment of propositional and relational domains. In: McCallum SRA (ed) Proceedings of the 25th international conference on machine learning (ICML 2008). Helsinki, Finland
25. Kersting K, Mladenov M, Tokmakov P (2015) Relational linear programming. Artif Intell J. **(Accepted for publication)**
26. Kersting K, van Otterlo M, de Raedt L (2004) Bellman goes relational. In: ICML-04. ACM Press, Banff, Alberta, Canada. doi:10.1145/1015330.1015401
27. Kimmig A, Mihalkova L, Getoor L (2015) Lifted graphical models: a survey. Mach Learn 99(1):1–45
28. Koller D, Friedman N (2009) Probabilistic graphical models—principles and techniques. MIT Press, Cambridge
29. Kulick J, Toussaint M, Lang T, Lopes M (2013) Active learning for teaching a robot grounded relational symbols. In: Proceedings of the 23rd international joint conference on artificial intelligence (IJCAI)
30. Lang T, Toussaint M (2009) Relevance grounding for planning in relational domains. In Buntine WL, Grobelnik M, Mladenic G, Shawe-Taylo J (eds) Proceedings of the european conference of machine learning and knowledge discovery in databases (ECML/PKDD), Part I. Springer, pp 736–751
31. Lang T, Toussaint M (2010) Planning with noisy probabilistic relational rules. J Artif Intell Res 39:1–49
32. Lang T, Toussaint M, Kersting K (2010) Exploration in relational worlds. In: Balcazar J, Bonchi F, Gionis A, Sebag M (eds) European conference on machine learning and principles and practice of knowledge discovery in databases (ECML PKDD-10). Springer, Barcelona
33. Lang T, Toussaint M, Kersting K (2012) Exploration in relational domains for model-based reinforcement learning. J Mach Learn Res 13:3691–3734
34. Natarajan S, Joshi S, Tadepalli P, Kersting K, Shavlik J (2011) Imitation learning in relational domains: a functional-gradient boosting approach. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI), pp 1414–1420
35. Nath A, Richardson M (2012) Counting-mlns: Learning relational structure for decision making. In: Proceedings of the twenty-sixth AAAI conference on artificial intelligence
36. Ngo V, Toussaint M (2014) Model-based relational RL when object existence is partially observable. In: Proceedings of the 31th international conference on machine learning (ICML), pp 559–567
37. Orthey A, Toussaint M, Jetchev N (2013) Optimizing motion primitives to make symbolic models more predictive. In: 2013 IEEE international conference on robotics and automation (ICRA), pp 2868–2873
38. Pasula H, Zettlemoyer L, Pack Kaelbling L (2007)Learning symbolic models of stochastic domains. J Artif Intell Res 29:309–352
39. Pearl J (1991) Reasoning in intelligent systems: networks of plausible Inference, 2nd edn. Morgan Kaufmann, San Francisco, CA, USA
40. Poole D (1997) The independent choice logic for modelling multiple agents under uncertainty. Artif Intell J 94:7–56

41. Poole D (1998) Decision theory, the situation calculus and conditional plans. Electron Trans Artif Intell 2(1–2). URL:http://www.etaij.org
42. Powell WB (2010) Merging ai and or to solve high-dimensional stochastic optimization problems using approximate dynamic programming. INFORMS INFORMS J Comput 22(1):2–17
43. Ramon J, Driessens K, Croonenborghs T (2007) Transfer learning in reinforcement learning problems through partial policy recycling. In: Proceedings of ECML, pp 699–707
44. Sanner S, Boutilier C (2007) Approximate solution techniques for factored first-order MDPs. In: Proceedings of ICAPS
45. Sanner S, Boutilier C (2009) Practical solution techniques for first order mdps. AIJ 173:748–788
46. Sanner S, Boutilier C (2009) Practical solution techniques for first-order MDPs. Artif Intell J 173(5–6):748–788
47. Sanner S, Kersting K (2010) Symbolic dynamic programming. In: C. Sammut, G. Webb (eds.) Encyclopedia of machine learning, p 946–954. Springer
48. Sanner S, Kersting K (2010) Symbolic dynamic programming for first-order POMDPs. In: Proc. AAAI-2010
49. Sarjant S, Pfahringer B, Driessens K, Smith T (2011) Using the online cross-entropy method to learn relational policies for playing different games. In: 2011 IEEE conference on computational intelligence and games (CIG), pp 182–189
50. Sutton RS, Barto AG (1998) Reinforcement learning: an Introduction. MIT Press, Canbridge, MA
51. Tadepalli P, Givan R, Driessens K (2004) Relational reinforcement learning: an overview. In: Proc. ICML workshop on relational reinforcement learning
52. Tenorth M, Klank U, Pangercic D, Beetz M (2011) Web-enabled robots—robots that use the web as an information resource. IEEE robotics and automation magazine p 18
53. Van Haaren J, Van den Broeck G, Meert W, Davis J (2014) Tractable learning of liftable Markov logic networks. In: Proceedings of the ICML-14 workshop on learning tractable probabilistic models (LTPM)
54. van Otterlo M (2009) The logic of adaptive behavior—knowledge representation and algorithms for adaptive sequential decision making under uncertainty in first-order and relational domains. IOS Press
55. Wang C, Joshi S, Khardon R (2008) First order decision diagrams for relational mdps. JAIR 31:431–472
56. Wang C, Khardon R (2010) Relational partially observable MDPs. In: Proc. AAAI-2010
57. Zamani Z, Sanner S, Poupart P, Kersting K (2012) Symbolic dynamic programming for continuous state and observation pomdps. In: Proceedings of the 26th annual conference on neural information processing systems (NIPS), pp 1403–1411

Kristian Kersting is an associate professor for Data Mining at the TU Dortmund University, Germany. He received his Ph.D. from the University of Freiburg and was previously with the MIT, the Fraunhofer IAIS, and the University of Bonn. His main research interests are Statistical Relational AI and Data Mining. He has published over 130 technical papers and received an ATTRACT fellowship, the ECCAI Dissertation Award 2006, the ECMLPKDD 2006 Best Student Paper Award, the ACM GIS 2011 Best Poster Award, and the AAAI 2013 Outstanding PC Member Award. He is a co-founder of the international workshop series on Statistical Relational AI, co-chaired ECMLPKDD 2013, and is an associate editor of JAIR, AIJ, DAMI, and MLJ.

Sriraam Natarajan is an assistant professor at Indiana University, USA. He was previously an assistant professor at Wake Forest School of Medicine, a post-doctoral research associate at University of Wisconsin-Madison and had graduated with his Ph.D. from Oregon State University. His main research interests lie in the fields of (Statistical Relational) AI and Machine Learning and their application to healthcare problems. He has published over 50 technical papers and is the recipient of an ARO Young Investigator Award. He gave a tutorial on lifted inference at IJCAI 2011, co-chaired SRL 2012 as well as CoLISD 2011–12, and is a co-founder of the international workshop series on Statistical Relational AI.