Advising and Instructing Reinforcement Learning Agents with LTL and Automata

Toryn Q. Klassen*
Department of Computer Science
University of Toronto, Canada
toryn@cs.toronto.edu

Introduction

Reinforcement learning (RL) involves an agent learning through interaction with an environment how to behave so as to maximize the expected cumulative reward (Sutton and Barto 1998). Typically, the environment is modelled as a Markov Decision Process (MDP). A reinforcement learning agent learns a *policy*, a mapping from states to actions. In a recent series of papers (Toro Icarte et al. 2018a; 2018b; 2018c), my coauthors and I have investigated ways in which *linear temporal logic* (LTL) and related formalisms can be incorporated into reinforcement learning algorithms, either to specify tasks for the agent or to guide the learning process. The remaining two sections describe each of these two research directions in turn.

Task specification

LTL extends propositional logic with temporal operators that can be used to describe patterns of events over a linear timeline (Pnueli 1977). For example, LTL can express the proposition "eventually P", meaning that the proposition P becomes true at some point in the timeline. LTL is an interesting language for describing tasks because it can define *non-Markovian* rewards that depend on the entire history.

Various works have used LTL to specify goals for MDP-like formalisms (Bacchus, Boutilier, and Grove 1996; Thiébaux et al. 2006; Lacerda, Parker, and Hawes 2014; 2015; Camacho et al. 2017; 2018), including in the context of RL (Fu and Topcu 2014; Li, Vasile, and Belta 2017; Littman et al. 2017; Sahni et al. 2017; Hasanbeig, Abate, and Kroening 2018). Our contribution with respect to LTL and goals was creating the *LTL Progression for Off-Policy Learning* (LPOPL) algorithm (Toro Icarte et al. 2018b), which efficiently learns policies for a given set Φ of tasks, each described by a (co-safe) LTL formula.

The way LPOPL works is as follows. First, the set Φ^+ of all possible iterated *progressions* of the formulas in Φ is computed. The progression of an LTL formula ϕ is another LTL formula ϕ', which intuitively describes what the future must be like if ϕ is to be made true, given that the present is a certain way. So Φ^+ can be thought of as including formulas describing all the possible outstanding obligations (or subtasks) that the agent may have during completion of any task in Φ. Training involves completing a series of episodes, each attempting to complete some task from Φ (selected according to some curriculum). While each episode is running, separate policies are concurrently being learned (off-policy) for achieving each of the formulas in Φ^+. This off-policy learning can be done either with tabular Q-learning or Deep Q-Networks (DQN) (Mnih et al. 2015). In the tabular case, convergence to the optimal policy can be guaranteed.

In our experiments using LPOPL (where the off-policy learning was done with DQN), LPOPL outperformed the baselines we compared against. The baselines operated on modified MDPs whose states were expanded to keep track of the progression of the LTL goal formulas (this made the rewards Markovian, which allowed standard RL algorithms to be applied). One baseline was just to use DQN. We also considered hierarchical methods based on the extension to deep RL (Kulkarni et al. 2016) of the options framework (Sutton, Precup, and Singh 1999).

Progression is not the only way to keep track of the progress made in satisfying an LTL formula. It is well-known that there is a *Büchi automaton* corresponding to any LTL formula, and progress can be tracked by tracking the current states of this automaton. This suggests defining tasks by directly specifying some form of automaton.

This idea was realized with the *reward machine* (Toro Icarte et al. 2018c), a form of finite-state automaton that describes how an agent may be rewarded by multiple different reward functions over the course of an episode (depending on the pattern of events that occur). Like LTL, reward machines can be used with either tabular or deep RL. We have done experiments with both, and in each case using reward machines outperformed the corresponding baselines (which were respectively based on tabular or deep RL). Again, some of our baselines were based on the options framework.

We may note that the hierarchy in hierarchical RL methods like the options framework is defined independently from the reward function. Policies that make use of macro-actions (i.e., options) may be suboptimal. In contrast, the structure given by a reward machines (or by an LTL formula, when it used to define a task) is what defines the rewards. This allows for the task to be decomposed in an optimal way (though of course, if the policies for the subtasks are learned by an algorithm like DQN which does not provide convergence guarantees, then the learned policy for the overall task may not be optimal).

*The research presented in this paper is joint work with Rodrigo Toro Icarte, Richard Valenzano, and Sheila A. McIlraith.

Advice

Another use that LTL can be put to for RL is for providing *advice*. Advice is additional information given to the agent which (unlike task specifications) does not define what provides reward for the agent, but may help the agent in achieving good rewards. For example, if you know that the agent will get reward when it opens a door, you might advise the agent to get a key. Advice had previously been considered in the literature (Maclin and Shavlik 1996; Krening et al. 2016), but not in the form of LTL.

We incorporated LTL advice into a *model-based* RL algorithm (Toro Icarte et al. 2018a). In model-based RL, the agent learns the dynamics of the environment, and constructs a policy based on that. The high-level idea of our algorithm is that during exploration of the environment, the agent is always aiming to reach the closest unknown environment transition currently "recommended" by the advice formula. The current recommendation is determined by keeping track of how much of the advice has been followed so far (using automata) and seeing what the advice calls for next. For example, if the advice is to get the key and open the door, and the agent has already got the key, then the current recommendation would be to go to the door.

Experiments (in a simple grid-world environment) showed that some pieces of advice could reduce the amount of training needed to find a fairly good policy, and the algorithm also eventually recovered from unhelpful advice.

Acknowledgements I acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Bacchus, F.; Boutilier, C.; and Grove, A. J. 1996. Rewarding behaviors. In *Proceedings of the 13th National Conference on AI (AAAI)*, 1160–1167.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. A. 2017. Non-Markovian rewards expressed in LTL: guiding search via reward shaping. In *Proceedings of the 10th Symposium on Combinatorial Search (SOCS)*, 159–160.

Camacho, A.; Chen, O.; Sanner, S.; and McIlraith, S. A. 2018. Non-Markovian rewards expressed in LTL: Guiding search via reward shaping (extended version). In *First Workshop on Goal Specifications for Reinforcement Learning*.

Fu, J., and Topcu, U. 2014. Probably approximately correct MDP learning and control with temporal logic constraints. In *Robotics: Science and Systems X*.

Hasanbeig, M.; Abate, A.; and Kroening, D. 2018. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*.

Krening, S.; Harrison, B.; Feigh, K.; Isbell, C.; Riedl, M.; and Thomaz, A. 2016. Learning from explanations using sentiment and advice in RL. *IEEE Transactions on Cognitive and Developmental Systems* 9(1):44–55.

Kulkarni, T. D.; Narasimhan, K.; Saeedi, A.; and Tenenbaum, J. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016)*, 3675–3683.

Lacerda, B.; Parker, D.; and Hawes, N. 2014. Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1511–1516.

Lacerda, B.; Parker, D.; and Hawes, N. 2015. Optimal policy generation for partially satisfiable co-safe LTL specifications. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 1587–1593.

Li, X.; Vasile, C. I.; and Belta, C. 2017. Reinforcement learning with temporal logic rewards. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3834–3839.

Littman, M. L.; Topcu, U.; Fu, J.; Isbell, C.; Wen, M.; and MacGlashan, J. 2017. Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341*.

Maclin, R., and Shavlik, J. 1996. Creating advice-taking reinforcement learners. *Machine Learning* 22(1-3):251–281.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Pnueli, A. 1977. The temporal logic of programs. In *Proceedings of the 18th Symposium on Foundations of Computer Science*, 46–57.

Sahni, H.; Kumar, S.; Tejani, F.; and Isbell, C. 2017. Learning to compose skills. *arXiv preprint arXiv:1711.11289*. Presented at the NIPS 2017 Deep Reinforcement Learning Symposium.

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

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1-2):181–211.

Thiébaux, S.; Gretton, C.; Slaney, J. K.; Price, D.; and Kabanza, F. 2006. Decision-theoretic planning with non-Markovian rewards. *Journal of Artificial Intelligence Research* 25:17–74.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2018a. Advice-based exploration in model-based reinforcement learning. In *Proceedings of the 31st Canadian Conference on Artificial Intelligence*, 72–83.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2018b. Teaching multiple tasks to an RL agent using LTL. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 452–461.

Toro Icarte, R.; Klassen, T. Q.; Valenzano, R.; and McIlraith, S. A. 2018c. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2112–2121.