Proceedings of the
1st Workshop on Artificial Intelligence and Formal Verification, Logics, Automata and Synthesis (OVERLAY),
Rende, Italy, November 19–20, 2019

Towards Verifiable and Safe Model-Free Reinforcement Learning*

Mohammadhosein Hasanbeig, Daniel Kroening, Alessandro Abate

University of Oxford, United Kingdom
[hosein.hasanbeig, daniel.kroening, alessandro.abate]@cs.ox.ac.uk

Abstract

Reinforcement Learning (RL) is a widely employed machine learning architecture that has been applied to a variety of decision-making problems, from resource management to robot locomotion, from recommendation systems to systems biology, and from traffic control to superhuman-level gaming. However, RL has experienced limited success beyond rigidly controlled or constrained applications, and successful employment of RL in safety-critical scenarios is yet to be achieved. A principal reason for this limitation is the lack of formal approaches to specify requirements as tasks and learning constraints, and to provide guarantees with respect to these requirements and constraints, during and after learning. This line of work addresses these issues by proposing a general framework that leverages the success of RL in learning high-performance controllers, while guaranteeing the satisfaction of given requirements and guiding the learning process within safe configurations.

1 Introduction

Machine learning is becoming ever more pervasive in providing workable solutions to industrial problems, and is starting to be leveraged to address tasks ever closer to the presence of humans or concerning data pertaining humans. Reinforcement Learning (RL), as an effective branch of machine learning, has similarly seen expanded use and increased success. RL provides a solution to decision-making problems either when no prior knowledge is available or when analytical solutions are hard to be found [55]. This practical approach has paved the way for RL to be employed in automatic control, game theory, economics, and biology inter alia [2,27,51,64].

Deep RL is arguably one of the recent breakthroughs in RL, whereby human-level game play has been achieved on a number of Atari games by incorporating the use of deep neural networks into RL [37]. The success of deep RL has resulted in the extensive use of RL beyond small-scale, classical contexts [19,29,52,58]. In particular, RL has recently been applied to safety-critical problems [3,10,14,16,22,23,30,40], including autonomous driving [47,48] and avionics [2,35]. This however inevitably entails the need for correct-by-design policy synthesis, in order to guarantee, among other quantitative requirements, the safety of

*This is work done with the following collaborators: Y. Kantaros, G. J. Pappas, and I. Lee from the School of Engineering and Applied Science, University of Pennsylvania, USA, and T. Melhem, N.Y. Jeppu, and L. Z. Yuan from the Computer Science Department, University of Oxford, UK. The last author is in part supported by the Alan Turing Institute, London, UK.

policies synthesised via RL. Furthermore, and with a different perspective on the notion, safety has to be present also "during learning", namely as the agent explores the environment.

Existing RL methods provide promising exploration guarantees, though they tend to rely either on an ergodicity assumption or a soft safety assumption. The essence of soft safety is that unsafe states, which might be absorbing sink states can be visited regardless of its possible catastrophic outcome. For most physical systems this assumption is not affordable as these systems may break before any meaningful exploration happens. Alternatively, the ergodicity assumption requires that any state is eventually reached from any other state when a proper policy is followed. This assumption allows RL to explore by simply favouring states that have rarely been visited, and are potentially unsafe in practice. Thus, unsurprisingly in safety-critical scenarios when the aforementioned assumptions do not hold most of the exploration methods are unrealistic.

Further to the discussed exploration problems, the expression of safety requirements in the RL literature has mostly relied on reward engineering [15]: it is often inconvenient to map complex goals to an appropriate reward structure in safety-critical problems [6]. In particular, [21] showed that the expected discounted reward might need careful tuning when dealing with safety requirements.

In this line of work we address the aforementioned issues in safe RL by leveraging concepts and techniques from Formal Methods. In particular, we employ Linear Temporal Logic (LTL) [43], as a formal high-level language in which a complex task can be easily described [9], and later automatically shape a reward function that is directly fed to an RL algorithm. The notion of safety in this sense is encompassed in the LTL specification and can be seen as RL generally not violating the property during and after learning. Thus, the proposed method benefits from reward engineering aspects that are standard in safe RL, and at the same time infuses notions from formal methods that allows guiding exploration and certifying learning outcomes in terms of the probability of staying safe. In addition to this algorithmic reward shaping, standard exploration schemes in RL need to relax the unrealistic assumptions on ergodicity and soft safety. In this work, we propose an adaptive safe padding mechanism [25] that does not rely on the aforementioned assumptions while automatically balances the trade-off between efficient exploration and ensuring safety during learning.

2 Background and Related Work

The interaction between an agent and its (possibly partly known) environment can be described by Markov Decision Processes (MDPs). MDPs are extensively used for modelling sequential decision making problems [44]. In this research, MDPs are discrete-time stochastic control processes defined as $\mathfrak{M} = (\mathcal{S}, \mathcal{A}, s_0, P, \mathcal{AP}, L)$ over a state space $\mathcal{S}$, an action space $\mathcal{A}$; a transition kernel $P(\cdot|s,a) \in \mathcal{P}(\mathcal{S})$ describes the dynamics, where $\mathcal{P}(\mathcal{S})$ is the set of probability distributions on subsets of $\mathcal{S}$; $\mathcal{AP}$ is a finite set of atomic propositions and a labelling function $L : \mathcal{S} \to 2^{\mathcal{AP}}$ assigns to each state $s \in \mathcal{S}$ a set of atomic propositions $L(s) \subseteq 2^{\mathcal{AP}}$; finally, the MDP is initialised in state s_0.

For the sake of generality, we assume that the MDP is initially unknown. Namely, the transition kernel P and also the labelling map L (known as grounding in RL) are not given to the agent. The agent is only able to observe its current state $s \in \mathcal{S}$, the label of the current state $L(s)$, and the action taken $a \in \mathcal{A}$.

A reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ is defined over the state-action space to denote the immediate bounded reward received by the agent from the environment after performing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$.

A policy is a rule according to which the agent chooses its action at a given state. More formally, a policy π is a mapping from the state space $\mathcal{S}$ to a distribution in $\mathcal{P}(\mathcal{A})$, where $\mathcal{P}(\mathcal{A})$ is the set of probability distributions on subsets of $\mathcal{A}$. A policy π is called stationary if $\pi(\cdot|s) \in \mathcal{P}(\mathcal{A})$ does not change over time and it is called a deterministic policy if $\pi(\cdot|s)$ is a degenerate distribution, namely if it deterministically selects a value in $\mathcal{A}$. For any policy π on an MDP $\mathfrak{M}$, the expected discounted return in state s is defined as [55]:

$$U^{\pi}_{\mathfrak{M}}(s) = \mathbb{E}^{\pi}[\sum_{n=0}^{\infty} \gamma^n \; R(s_n, a_n) | s_0 = s], \tag{1}$$

where $\mathbb{E}^{\pi}[\cdot]$ denotes the expected value under policy π, γ is the discount factor[1], and $s_0, a_0, s_1, a_1, \ldots$ is

[1] The discount factor γ is a hyper-parameter that in general can be tuned. In particular, there is standard work in RL on

the sequence of state-action pairs generated by the MDP under policy π. The expected discounted return is often referred to as the "utility function" and is the optimisation objective in RL. We might drop the subscript $\mathfrak{M}$ when clear from the context in the rest of this work. With (1) as the optimisation objective, an optimal policy π^* is a policy that maximises the expected return, as

$$\pi^*(s) = \operatorname*{argsup}_{\pi \in \varpi} U^{\pi}_{\mathfrak{M}}(s),$$

where ϖ is the set of stationary deterministic policies over the state space $\mathcal{S}$. Note that in any MDP $\mathfrak{M}$ with a bounded reward function and a finite action space optimal policies are proved to be stationary and deterministic [8, 44], hence the restriction to set ϖ.

Linear Temporal Logic (LTL) [43] denotes a formal language that can express a wide range of complex, time-dependent objectives in a succinct and human-interpretable form. In particular, LTL can be used to express requirements on a given MDP, and there exists a substantial body of research on extraction of LTL properties from requirements (safety, liveness and eventuality, successive satisfaction) expressed in natural languages [20, 39, 61]. Conversely, the expression of of such complex, possibly memory dependent objectives by hand crafting of rewards is often cumbersome, if at all possible [6, 28]. As detailed later, LTL allows to automatically and formally shape reward functions usable in RL with guarantees on the outcomes.

The problem of control synthesis for LTL in finite-state MDPs has been considered in numerous works - we summarise a couple of broad options. In [60], a modified Dynamic Programming (DP) is employed, to maximise the worst-case probability of satisfying the LTL over all transition probabilities – however, [60] assumes to a-priori know the full MDP. [11] relaxes this assumption and considers an MDP to have unknown transition probabilities: a Probably Approximately Correct MDP (PAC MDP) is then generated via an RL-like algorithm, and value iteration is then applied to synthesise the control policy.

Focusing exclusively on the safety fragment of LTL, the concept of shielding is proposed in [3], where a shield is a reactive machine to ensure agent safety during learning given that an abstraction of the MDP is given. [12, 13] address safety-critical settings in the context of cyber-physical systems, where the agent has to select a correct model within a heterogeneous set of models in RL so that the safety constraint is not violated during and after learning.

By relaxing safety during learning requirement, [7] synthesises policies that maximise the probability of satisfying given unbounded reachability properties: the policy generation relies on an approximate DP over a model that is generated via model-based RL. Much in the same direction, [45] employs a learning-based approach to generate a policy that is able to certifiably satisfy a given LTL property. The algorithms in [3, 7, 45] are "model-based" and hinge on learning the transition probabilities of the unknown MDP, which requires a sufficiently large number of simulations has to be executed to make sure that the probability approximations are accurate enough.

Compared to the mentioned approaches, we propose the first "model-free" framework for LTL synthesis [22, 26], which means that we are able to synthesise optimal policies (1) without knowing MDP graph and its transition probabilities (as opposed to DP); and (2) without any pre-processing or construction of a model of the MDP. In standard RL literature, model-free methods are very successful, since they learn a "direct" mapping from states and actions to the associated expected reward. Model-free algorithms display applicability to a broader range of problems, whereas model-based approaches are not as general [18], even though model-based learning can come with convenient theoretical guarantees [31, 54].

In the domain of continuous-state and -action MDPs, to the best of our knowledge, no research has been done to enable model-free RL to generate policies under LTL properties. The framework proposed is the first that can handle MDPs with uncountably-infinite state space [24] and uncountably-infinite action space [63] in this context. Conversely, whenever a model of the MDP is known [1, 53, 56] there is a broad literature on this issue.

state-dependent discount factors [38, 42, 59, 62], which is shown to preserve convergence and optimality guarantees.

3 Logically-Constrained Reinforcement Learning (LCRL) [33]

Given an LTL specification, we propose the first framework that allows a model-free RL algorithm to synthesise a control policy for a finite-state MDP (and for the more general, continuous-state case), such that the generated traces satisfy the LTL property with maximal probability in finite-state MDP case. In this framework, the LTL property essentially acts as a high-level monitor for the agent planning, whereas the low-level synthesis is handled by a native RL scheme. In order to synchronise this high-level guide with the RL architecture, we convert the LTL property into an automaton, namely a finite-state machine [5]. In general, the LTL-to-automaton translation can generate non-deterministic models, over which policy synthesis for MDPs is not semantically meaningful. A standard solution to this issue is to use the "Safra construction" [46] to determinise the automaton, which as expected can greatly increase its size [41]. An alternative solution is to directly convert the given LTL formula into a Deterministic Rabin Automaton (DRA), which by definition rules out non-determinism. Nevertheless, it is known that such a conversion results, in the worst case, in automata that are doubly exponential in the size of the original LTL formula [4]. Nevertheless, the LTL-to-DRA conversion is the standard method employed for LTL synthesis in the RL literature, e.g. [7, 11, 45, 60]. Conversely, in this work we propose to express the given LTL property as a Limit Deterministic Büchi Automaton (LDBA) [49].

An LDBA is a special case of a finite-state machine called Generalized Büchi Automaton (GBA). A GBA $\mathfrak{A} = (\mathcal{Q}, q_0, \Delta, \Sigma, \mathcal{F})$ is a tuple, where $\mathcal{Q}$ represents a finite set of states, $q_0 \in \mathcal{Q}$ is the initial state, $\Delta : \mathcal{Q} \times \Sigma \to 2^{\mathcal{Q}}$ is a transition relation, $\Sigma = 2^{\mathcal{AP}}$ is a finite alphabet, and $\mathcal{F} = \{\mathcal{F}_1, ..., \mathcal{F}_f\}$ is the set of accepting conditions, where $\mathcal{F}_j \subset \mathcal{Q}, 1 \leq j \leq f$. A GBA $\mathfrak{A} = (\mathcal{Q}, q_0, \Delta, \Sigma, \mathcal{F})$ is an LDBA if $\mathcal{Q}$ can be partitioned into two disjoint sets $\mathcal{Q} = \mathcal{Q}_N \cup \mathcal{Q}_D$ such that [49]: (1) $\Delta(q, \alpha) \subset \mathcal{Q}_D$ and $|\Delta(q, \alpha)| = 1$ for every state $q \in \mathcal{Q}_D$ and for every $\alpha \in \Sigma$; (2) for every $\mathcal{F}_j \in \mathcal{F}$, $\mathcal{F}_j \subset \mathcal{Q}_D$; and (3) there are non-deterministic ε-transitions[2] from $\mathcal{Q}_N$ to $\mathcal{Q}_D$. See Figure 1a for an example. It can be shown that this construction results in an exponential-sized automaton for LTL\GU[3], whereas it results in nearly the same size as a DRA for the rest of LTL. Furthermore, a Büchi automaton is semantically easier than a Rabin automaton in terms of its acceptance conditions, which makes policy synthesis algorithms much simpler to implement [50, 57]. We should conclude emphasising that there exist a few LDBA construction algorithms for LTL, but not all of resulting LDBAs can be employed for model checking or synthesis over MDPs [32].

Once the LDBA is generated from the given LTL property, we employ it to monitor executions of the MDP in the learning process. This can be done leveraging a classical product construction, namely a synchronous product between the MDP and the resulting LDBA (Figure 1c). Technically, given an MDP $\mathfrak{M} = (\mathcal{S}, \mathcal{A}, s_0, P, \mathcal{AP}, L)$ and an LDBA $\mathfrak{A} = (\mathcal{Q}, q_0, \Delta, \Sigma, \mathcal{F})$ with $\Sigma = 2^{\mathcal{AP}}$, the product MDP is defined as $\mathfrak{M} \otimes \mathfrak{A} = \mathfrak{P} = (\mathcal{S}^{\otimes}, \mathcal{A}, s_0^{\otimes}, P^{\otimes}, \mathcal{AP}^{\otimes}, L^{\otimes}, \mathcal{F}^{\otimes})$, where $\mathcal{S}^{\otimes} = \mathcal{S} \times \mathcal{Q}$, $s_0^{\otimes} = (s_0, q_0)$, $\mathcal{AP}^{\otimes} = \mathcal{Q}$, $L^{\otimes} : \mathcal{S}^{\otimes} \to 2^{\mathcal{Q}}$ such that $L^{\otimes}(s, q) = q$ and $\mathcal{F}^{\otimes} \subseteq \mathcal{S}^{\otimes}$ is the set of accepting states $\mathcal{F}^{\otimes} = \{\mathcal{F}_1^{\otimes}, ..., \mathcal{F}_f^{\otimes}\}$, where $\mathcal{F}_j^{\otimes} = \mathcal{S} \times \mathcal{F}_j$. The transition kernel $P^{\otimes}(\cdot|s_i^{\otimes}, a) \in \mathcal{P}(\mathcal{S}^{\otimes})$ is such that given the current state (s_i, q_i) and action a, the new state is (s_j, q_j) is obtained by $s_j \sim P(\cdot|s_i, a)$ and $q_j \in \Delta(q_i, L(s_j))$. In view of the model-free feature of our approach, such as construction can unfold "on-the-fly", meaning that the algorithm tracks the state of an underlying structure without explicitly constructing the entire structure a-priori. As such, let us underline that, when running LCRL, there is no need to build the above product MDP and to store all its states and transitions in memory. The product MDP transitions can be executed so that the agent can track the automaton state by just looking at the trace that has been read so far. The agent only needs to store the current state of the automaton and observe the label at each step to check whether the automaton state has changed or not.

At this stage, in order to feed the constructed product MDP to an RL scheme, a reward function ought to be defined - however, recall we want to avoid ad-hoc reward engineering. On the contrary, we introduce a general reward function that depends on the accepting conditions of the Büchi automaton over the state-action pairs of the MDP [23, 26]. Such a reward function is adaptive to the current state of each learning episode. Using this reward, an RL scheme can generate a policy (or policies) returning the maximum expected reward: in the finite-state case we show that this is the policy that satisfies the given

[2] An ε-transition allows an automaton to change its state without reading any atomic proposition.

[3] LTL\GU is a fragment of linear temporal logic with the restriction that no until operator occurs in the scope of an always operator

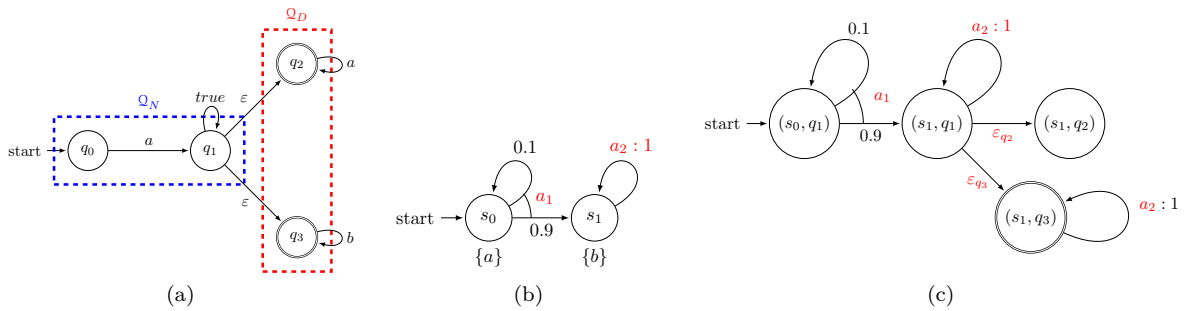

Figure 1: (a) LDBA for the example formula $a \wedge \bigcirc(\Diamond\Box a \vee \Diamond\Box b)$; (b) instance of an MDP; (c) Product MDP obtained from the LDBA and the MDP.

LTL property with maximal probability. As mentioned above, we also propose a mechanism to determine this probability while the agent is learning the MDP: consequently, we can certify the generated policy with respect to the LTL property at hand.

LCRL thus targets verified learning in its core, namely the learning-based synthesis of policies that abide by given requirements. Additionally, LCRL can be endowed with the capacity to "learn while being safe". More precisely, formulating the desired objective via LDBA in safety-critical problems allows the agent to predict unsafe outcomes before running into risky states. In particular, we present the concept of an adaptive safe padding in [25] that forces RL to synthesize optimal control policies while being safe. Enforcing RL to stay safe during learning might limit the exploration in some safety-critical cases. However, we show that the proposed architecture is able to automatically handle the trade-off between efficient progress in exploration and ensuring strict safety [25]. Recall that the expression of safety in the classical RL literature mostly relies on reward engineering [6, 15], while with the LDBA-synchronised reward function, we are able to automatically shape the reward function.

4 Case Studies

The performance of the proposed framework has been evaluated via a set of numerical examples and benchmarks, where we observe an improvement of one order of magnitude in the number of iterations required for policy synthesis, compared to existing approaches that mostly rely on model-based setups, e.g. [7, 45]. Much of this faster convergence and scalability is owed to our model-free architecture, and to the on-the-fly construction underpinning LCRL.

In the domain of finite-state MDPs, the experiments include robot planning in different layouts of slippery grid worlds with at least $|\mathcal{S}| = 1600$ [22]. Further, we have explored the well-known Atari 2600 game Pacman with state-space cardinality in excess of $|\mathcal{S}| = 80{,}000$, where conventional methods in RL have failed to synthesise an optimal and stable policy [26]. LCRL has outperformed DeepQN [37] in the complex environment of Atari 2600 Montezuma's Revenge (Figure 2a) with over $|\mathcal{S}| = 700{,}000$ states [23]. Finally, with safety as an additional hard constraint in the learning process, we have extended LCRL to synthesise optimal policies while being safe both in the slippery grid world and in Pacman [25].

With focus on uncountably-infinite state-space MDPs, we have tested the LCRL architecture in a mission planning task for an autonomous Mars-rover [24]. We have shown that, while keeping the sample complexity significantly lower, LCRL succeeds to generate policies that are more reliable with better expected reward, i.e. higher probability of satisfying the LTL property, as compared to other methods such as Voronoi quantizer [34] and fitted-value iteration [17]. [63] makes a step forward and extends MDPs with continuous states to encompass continuous actions, and incorporates deep learning architectures in LCRL to solve an actual mission of the NASA Opportunity rover (Figure 2b).

(a) (b)

Figure 2: (a) The agent successfully unlocks the door in Montezuma's Revenge first stage, (b) Generated path by LCRL around the Victoria crater replicating the actual NASA Opportunity rover traverse map.

5 Conclusions and Future Directions

LCRL has shown promising performance in all the benchmarks we have considered, and quite importantly it is fully compatible with native RL algorithms, as well as with new ones that are at the core of recent developments in the community, e.g. [36, 37]. Namely, LCRL is a general framework that can accept any off-the-shelf model-free RL scheme. Hence, we believe that the proposed approach can open up to further cross-fertilisation in the area.

For future work we are currently looking into a multi-agent setup, in which a set of agents attempts (in coordinated or adversarial fashion) to satisfy a temporal property (or set thereof). Further, we would like to extend this approach to partially observable MDPs to limit the knowledge of the agent even more.

References

[1] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[2] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. *NIPS*, 19:1, 2007.

[3] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding. *arXiv preprint arXiv:1708.08611*, 2017.

[4] R. Alur and S. La Torre. Deterministic generators and games for LTL fragments. *TOCL*, 5(1):1–25, 2004.

[5] C. Baier, J.-P. Katoen, and K. G. Larsen. *Principles of Model Checking*. MIT press, 2008.

[6] S. Bharadwaj, S. Le Roux, G. Pérez, and U. Topcu. Reduction techniques for model checking and learning in MDPs. In *Proceedings of the 26st International Joint Conference on Artificial Intelligence, Melbourne*, pages 4273–4279, 2017.

[7] T. Brázdil, K. Chatterjee, M. Chmelík, V. Forejt, J. Křetínskỳ, M. Kwiatkowska, D. Parker, and M. Ujma. Verification of Markov decision processes using learning algorithms. In *ATVA*, pages 98–114. Springer, 2014.

[8] R. Cavazos-Cadena, E. A. Feinberg, and R. Montes-De-Oca. A note on the existence of optimal policies in total reward dynamic programs with compact action sets. *Mathematics of Operations Research*, 25(4):657–666, 2000.

[9] E. M. Clarke Jr, O. Grumberg, D. Kroening, D. Peled, and H. Veith. *Model checking*. MIT press, 2018.

[10] S. P. Coraluppi and S. I. Marcus. Risk-sensitive and minimax control of discrete-time, finite-state Markov decision processes. *Automatica*, 35(2):301–309, 1999.

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

[12] N. Fulton. *Verifiably Safe Autonomy for Cyber-Physical Systems*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 2018.

[13] N. Fulton and A. Platzer. Verifiably safe off-model reinforcement learning. In *TACAS*, pages 413–430, 2019.

[14] J. Garcia and F. Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

[15] J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *JMLR*, 16(1):1437–1480, 2015.

[16] P. Geibel and F. Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

[17] G. J. Gordon. Stable function approximation in dynamic programming. In *Machine Learning*, pages 261–268. Elsevier, 1995.

[18] M. Grześ. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 565–573. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[19] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[20] E. Gunter. From natural language to linear temporal logic: Aspects of specifying embedded systems in LTL. In *Proceedings of the Monterey Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation*, 2003.

[21] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *TACAS*, pages 395–412. Springer, 2019.

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

[23] M. Hasanbeig, A. Abate, and D. Kroening. Certified reinforcement learning with logic guidance. *arXiv preprint arXiv:1902.00778*, 2019.

[24] M. Hasanbeig, A. Abate, and D. Kroening. Logically-constrained neural fitted Q-iteration. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2012–2014. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[25] M. Hasanbeig, A. Abate, and D. Kroening. Cautious reinforcement learning with logical constraints. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

[26] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. *IEEE Conference on Decision and Control*, 2019.

[27] M. Hasanbeig and L. Pavel. On synchronous binary log-linear learning and second order Q-learning. In *The 20th World Congress of the International Federation of Automatic Control (IFAC)*. IFAC, 2017.

[28] M. Hasanbeig, N. Yogananda Jeppu, A. Abate, T. Melham, and D. Kroening. Deepsynth: Program synthesis for automatic task segmentation in deep reinforcement learning. *arXiv preprint arXiv:1911.10244*, 2019.

[29] M. Hausknecht and P. Stone. Deep recurrent Q-learning for partially observable MDPs. In *2015 AAAI Fall Symposium Series*, 2015.

[30] N. Jansen, B. Könighofer, S. Junges, and R. Bloem. Shielded decision-making in MDPs. *arXiv preprint arXiv:1807.06096*, 2018.

[31] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

[32] D. Kini and M. Viswanathan. Optimal translation of LTL to limit deterministic automata. In *TACAS*, pages 113–129. Springer, 2017.

[33] LCRL_Website. https://grockious.github.io/lcrl.

[34] I. S. Lee and H. Y. Lau. Adaptive state space partitioning for reinforcement learning. *Engineering applications of artificial intelligence*, 17(6):577–588, 2004.

[35] J. Lope and J. Martin. Learning autonomous helicopter flight with evolutionary reinforcement learning. In *International Conference on Computer Aided Systems Theory*, pages 75–82. Springer, 2009.

[36] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937, 2016.

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

[38] R. G. Newell and W. A. Pizer. Discounting the distant future: how much do uncertain rates increase valuations? *Journal of Environmental Economics and Management*, 46(1):52–71, 2003.

[39] A. P. Nikora and G. Balcom. Automated identification of LTL patterns in natural language requirements. In *ISSRE*, pages 185–194. IEEE, 2009.

[40] M. Pecka and T. Svoboda. Safe exploration techniques for reinforcement learning–an overview. In *International Workshop on Modelling and Simulation for Autonomous Systems*, pages 357–375. Springer, 2014.

[41] N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *LICS*, pages 255–264. IEEE, 2006.

[42] S. Pitis. Rethinking the discount factor in reinforcement learning: A decision theoretic approach. *arXiv preprint arXiv:1902.02893*, 2019.

[43] A. Pnueli. The temporal logic of programs. In *Foundations of Computer Science*, pages 46–57. IEEE, 1977.

[44] M. L. Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[45] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In *CDC*, pages 1091–1096. IEEE, 2014.

[46] S. Safra. On the complexity of omega-automata. In *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pages 319–327. IEEE, 1988.

[47] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

[48] S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[49] S. Sickert, J. Esparza, S. Jaax, and J. Křetínskỳ. Limit-deterministic Büchi automata for linear temporal logic. In *CAV*, pages 312–332. Springer, 2016.

[50] S. Sickert and J. Křetínskỳ. MoChiBA: Probabilistic LTL model checking using limit-deterministic Büchi automata. In *ATVA*, pages 130–137. Springer, 2016.

[51] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[52] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[53] S. E. Z. Soudjani, C. Gevaerts, and A. Abate. FAUST^2: Formal Abstractions of Uncountable-STate STochastic Processes. In *TACAS*, pages 272–286. Springer, 2015.

[54] A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[55] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[56] I. Tkachev, A. Mereacre, J.-P. Katoen, and A. Abate. Quantitative automata-based controller synthesis for non-autonomous stochastic hybrid systems. In *HSCC*, pages 293–302. ACM, 2013.

[57] I. Tkachev, A. Mereacre, J.-P. Katoen, and A. Abate. Quantitative model-checking of controlled discrete-time Markov processes. *Information and Computation*, 253:1–35, 2017.

[58] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, et al. Alphastar: Mastering the real-time strategy game StarCraft II. *DeepMind Blog*, 2019.

[59] Q. Wei and X. Guo. Markov decision processes with state-dependent discount factors and unbounded rewards/costs. *Operations Research Letters*, 39(5):369–374, 2011.

[60] E. M. Wolff, U. Topcu, and R. M. Murray. Robust control of uncertain Markov decision processes with temporal logic specifications. In *CDC*, pages 3372–3379. IEEE, 2012.

[61] R. Yan, C.-H. Cheng, and Y. Chai. Formal consistency checking over specifications in natural languages. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pages 1677–1682. EDA Consortium, 2015.

[62] N. Yoshida, E. Uchibe, and K. Doya. Reinforcement learning with state-dependent discount factor. In *ICDL*, pages 1–6. IEEE, 2013.

[63] L. Z. Yuan, M. Hasanbeig, A. Abate, and D. Kroening. Modular deep reinforcement learning with temporal logic specifications. *arXiv preprint arXiv:1909.11591*, 2019.

[64] Z. Zhou, X. Li, and R. N. Zare. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12):1337–1344, 2017.