A Survey of Reinforcement Learning with Temporal Logic Rewards

Hsuan-Cheng Liao
Technische Universität München
Email: brian.liao@tum.de

***Abstract*—This paper presents a survey of Reinforcement Learning frameworks with Temporal Logic rewards. Reinforcement Learning (RL) has emerged to be a powerful tool to solve sequential decision-making problems and been successfully applied to various fields, including classical control tasks, Atari games, and robot simulations. However, it has found its path rather challenging to domains involving high-level properties, which are not easily captured by typical reward mechanisms. An example is a robotic manipulator commanded to reach several targets in a specific order while avoiding other obstacles. Such problems, on the other hand, are shown possible to be handled by Temporal Logics (TLs), one branch of Formal Methods. Thus, this paper surveys the literature and presents a wide variety of algorithms that aims at solving such high-level RL problems with the aid of TLs. Specifically, the methods under investigation cover from classical model-based approaches to modern actor-critic techniques. Moreover, a few potential research directions are suggested in the end of the paper.**

I. Introduction

Over the past decade, Reinforcement Learning (RL) has attracted unprecedented research focus and achieved outstanding performance in a broad array of applications. However, the breakthroughs reported so far are mostly limited to areas where reward functions can be modeled heuristically in a straightforward fashion. For example, in Atari games, the agent purely aims at optimizing the accumulated game score, which is a clear numeric objective. There is still a lack of successful paradigms of RL approaches in the field of robotics or control applications in which learning goals of the agents are usually more complicated or even involve various tasks. As an example, Fig. 1 depicts a task where a robotic arm is supposed to reach three goals in succession, while avoiding all obstacles [1]. Such a task is rather complicated for typical real-valued reward functions, as it requires subtasks being fulfilled with certain high-level properties. The task is, however, possible to be modeled in formal specification languages such as Temporal Logics (TLs), which has been proven capable of incorporating high-level intentions, constraints, or domain knowledge. For instance, (1) gives the TL formula for the example described above.

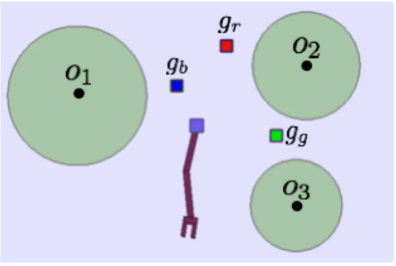

Fig. 1. An example of a high-level task requirement to reach g_r, g_b, g_g in a row and avoid obstacle $o's$ [1].

$$\phi = (reach\ g_{\mathrm{r}},\ then\ reach\ g_{\mathrm{g}},\ then\ reach\ g_{\mathrm{b}}) \wedge \Big(\bigwedge_{i=1,2,3} always\ avoid\ o_i\Big) \quad (1)$$

Considering the potential of TLs, we examine related work on formulating TLs as reward functions in RL frameworks. Both topics are widely studied, with TLs originated from model checking [2] and RL from optimal control and dynamic programming [3]. Nonetheless, it is observed that research applying the combination of the two to robotics, control, or planning problems has not drawn attention until lately in the past dacade. Therefore, we investigate papers from the early stages of this line of research, identify the challenges and improvements in the course, and propose promising directions for future research. As an overviewing paper, we defer readers to the original papers for detailed mathematical derivations and adopt an introductory manner to accommodate all types of related work in this paper.

The remaining of the paper is organized as follows: Section II elaborates a background on the two primary subjects, namely RL and TLs. The mass literature is reviewed in Section III, followed by a discussion including future work suggestions in Section IV. Finally, Section V gives the concluding remarks.

II. Preliminaries

In this section, we introduce the background knowledge for both Reinforcement Learning (RL) and Temporal Logics (TLs). First, Subsection II-A formalizes RL frameworks into Markov Decision Processes (MDPs). Then, Subsection II-B introduces the syntax and semantics of TLs. Lastly, Subsection II-C draws the technical problem statements with the provided definitions.

A. Markov Decision Processes and Reinforcement Learning

MDPs prescribe a general formalism of systems comprising sequences of state transitions and actions. Considering the use of TLs later on, we present here the definition of a Labeled MDP.

Definition II.1. (Labeled MDP) *A Labeled MDP is a tuple* $\mathcal{M} = \langle S, A, P, s_0, AP, L, R, \gamma \rangle$*, where* S *is a set of possible states,* A *is a set of primitive actions,* P *is the transition probability function defined as* $P : S \times A \times S \to [0, 1]$*,* $s_0 \in S$ *is the initial state,* AP *is a finite set of atomic propositions, and* $L : S \to 2^{AP}$ *is a labeling function which assigns each state a set of atomic prepositions that are valid in that state. In the context of RL, an additional reward function* $R : S \times A \times S \to \mathbb{R}$ *and a discount factor* γ *for the reward are usually emphasized and included in the tuple.*

Given the Markov property in an MDP, both the transition and reward function depend solely on the state and action in the previous step. However, we shall later see how various approaches circumvent this property, observing sequences of states and actions to evaluate TL specifications. Now, from the RL agents' perspective, there are several terms to be defined.

Definition II.2. (Policy) *A control policy* π *for the agent in a Labeled MDP* $\mathcal{M}$ *is a function* $\pi : S \to A$ *deterministically, or* $\pi : S \times A \to [0, 1]$ *stochastically.*

Definition II.3. (Trajectory) *Given a Labeled MDP* $\mathcal{M}$*, a state-action trajectory from time* 0 *to* T *is denoted as* $\tau = (s_0, a_0, s_1, a_1, ..., s_T)$ *under a certain policy* π*. In case of a stochastic policy* π*,* $p^\pi(\tau)$ *represents the trajectory distribution.*

For better generality, we do not restrict ourselves to finite trajectories at this point, meaning that possibly $T \to \infty$ in Definition II.3.

Definition II.4. (Accumulated reward) *Given a trajectory* τ *in a Labeled MDP* $\mathcal{M}$*, the accumulated reward is computed as* $G(\tau) = \sum_{t=0}^{T} \gamma^t r_t$ *where* r_t *is obtained from the Markov reward function* R*.*

Normally, the objective of an RL algorithm is to obtain an optimal policy that maximizes the cumulative reward over a trajectory, formulated as follows:

$$\pi^* = argmax_\pi \, \mathbb{E}_{\tau \backsim \pi}[G(\tau)]. \tag{2}$$

B. Temporal Logics

Temporal Logics (TLs) provide an interface to specify high-level system behaviors such as "safety" ($always \quad A$), "liveness" ($infinitely \quad often \quad A$), "stability" ($eventually \quad always \quad A$), and "priority" ($first \; A, \; then \; B, \; then \; C$) [4]. Throughout this paper, three variants of TLs are mentioned, namely Linear Temporal Logic (LTL), Signal Temporal Logic (STL), and Truncated Linear Temporal Logic (TLTL). The basic syntaxes are similar among these variants as all include (a) a set of atomic prepositions, (b) Boolean operators: negation ($\neg$), conjunction ($\wedge$), and (c) temporal operators: next ($\bigcirc$), and until (U). Other Boolean operators such as disjunction ($\vee$), implication ($\rightarrow$), equivalence ($\leftrightarrow$), and exclusive or ($\otimes$) can be solicited from the basic ones. Likewise, further temporal modalities including eventually ($\Diamond$) and always ($\Box$) can be derived from next ($\bigcirc$) and until (U). Due to the space constraint, we refer interested readers to Chapter 5.1 in [2] for formal derivations of the syntax of TL.

To evaluate a TL specification, two kinds of semantics are noticed. First, qualitative semantics determines if a system trajectory fulfills a specification, resulting in Boolean true or false values. Second, quantitative semantics evaluates how satisfyingly a system trajectory completes a specification, endowing numeric results. The former is provided by all three variants of TLs, whereas the latter is allowed by STL and TLTL only. One advantage of quantitative semantics is that it naturally serves as an indicator of how robust a state trajectory is against a specification, and thus being also called robustness degree. Definitions are detailed in the original papers [1] [5] [6] and other surveys [7]. We simply stress here the difference between TLTL and STL is that for the former, there is no need to explicitly declare boundaries on time modalities, such as "eventually before time t ($\Diamond_{[0,t]}$)" or "globally always over time t to $t+k$ ($\Box_{[t,t+k]}$)". This is later shown allowing the agent to focus on the high-level tasks, instead of the hard constraints regulating the time at which they have to be finished.

C. Problem Statements

With the background given, two formal problem statements extending the RL objective (2) into the domain of TLs are listed [5]. The first one, which has been seen in most of the work, maximizes the probability of satisfying an TL formula, while the second takes into consideration the quantitative semantics and maximizes the robustness degree.

Problem 1. Given a TL specification ϕ and a state trajectory τ under a labelled MDP $\mathcal{M} = \langle S, A, P, s_0, AP, L, R, \gamma \rangle$, find the optimal control policy such that

$$\pi_1^* = argmax_\pi \, p^\pi[\tau \models \phi], \tag{3}$$

where $p^\pi[\tau \models \phi]$ is the probability of trajectory τ satisfying specification ϕ under policy π.

Problem 2. Given a TL specification ϕ and a state trajectory τ under a labelled MDP $\mathcal{M} = \langle S, A, P, s_0, AP, L, R, \gamma \rangle$, find the optimal control policy such that

$$\pi_2^* = argmax_\pi \, \mathbb{E}_{\tau \backsim \pi}[\rho(\tau, \phi)], \tag{4}$$

where $\mathbb{E}_{\tau \backsim \pi}[\rho(\tau, \phi)]$ is the expected robustness degree of trajectory τ against specification ϕ under policy π.

Looking upon these TL-oriented problems, two major challenges are spotted due to the Markov property from RL. First, for state transition, how should history-dependencies of TL events be captured over the single-step transition probability function? Second, for reward collection, how should a feedback be given immediately if the TL specification requires several steps of states over iterations? We shall refer these as **Challenge 1** and **Challenge 2**, and emphasize how various works resolve them in the upcoming sections.

III. Literature Review

The following main section of the paper surveys the related work, roughly aligned with the timeline of publications. Specifically, model-based approaches (III-A) which evolve from probabilistic model checking and control are revealed earlier, and model-free approaches (III-B) including Value Iteration, Policy Search, and Actor-Critic algorithms are attended more lately. In addition, a very recent line of work focusing on hierarchical RL and multitasking is illustrated in the last Subsection III-C.

A. Model-Based Approaches

1) Given system model: With an origin from system modeling and control, the early studies naturally assume given system dynamics as the MDP transition probability, and solves Problem 1 with dynamic programming algorithms. For example, the work from Wolff *et al.* [4] is among these. To tackle Challenge 1, they follow a two-step construction of a Product MDP, first converting the LTL formula to a Deterministic Rabin Automaton (DRA), and then integrating the DRA with the system MDP. The succeeding definitions are unified from [4] [8] [9], in which the notion of a reward function is not yet explicitly found.

Definition III.1. (DRA) *A DRA is a tuple* $\mathcal{A} = \langle Q, \sum, \delta, q_0, Acc \rangle$*, where* Q *is a finite set of states,* $\sum$ *is an input alphabet,* $\delta : Q \times \sum \rightarrow Q$ *is the transition function,* $q_0 \in Q$ *is the initial state, and* $Acc = (J_i, K_i)$ *with* $J_i, K_i \subseteq Q$ *for all* $i = 1, ..., N$ *is the set of accepting state pairs.*

Generally, a run of a DRA $\sigma = q_0, q_1, q_2, ...$ is accepted if there exists at least one pair $(J_i, K_i) \in Acc$ such that J_i is visited finitely often and K_i is visited infinitely often.

Definition III.2. (Product MDP) *With a Labeled MDP* $\mathcal{M} = \langle S, A, P, s_0, AP, L \rangle$ *and a DRA* $\mathcal{A} = \langle Q, \sum, \delta, q_0, Acc \rangle$*, the Product MDP is given by* $\mathcal{M}_p = \mathcal{M} \times \mathcal{A} = \langle S_p, A_p, P_p, s_{p_0}, Acc_p, L_p \rangle$*, with* $S_p = S \times Q$*,* A_p *inherited from A,* $P_p : S_p \times A_p \times S_p \rightarrow [0, 1]$ *being the transition probability function equal to* $P(s, a, s')$ *if* $q' = \delta(q, L(s'))$ *and* 0 *otherwise,* $s_{p_0} = (s_0, q)$ *where* $q = \delta(q_0, L(s_0))$*. Lastly,* Acc_p *and* L_p *are lifted similarly.*

Having incorporated TL history-dependencies into a Product MDP, it remains to deliver immediate rewards to the agent at each time step. In [4] [8] [9], Accepting Maximal Ending Components (AMECs) are proposed for such purpose.

Definition III.3. (AMEC) *Given a Product MDP* $\mathcal{M}_p$*, an End Component* $\mathcal{C}$ *is a sub-MDP* $\langle S_\mathcal{C}, A_\mathcal{C}, P_\mathcal{C} \rangle$ *such that* $S_\mathcal{C} \subseteq S_p$ *and the map* $A_\mathcal{C} : S_\mathcal{C} \rightarrow 2^{A_p} \subseteq A_p$ *are not empty, and that for any* $s_\mathcal{C} \in S_\mathcal{C}$ *and* $a_\mathcal{C} \in A_\mathcal{C}(s_\mathcal{C})$*, if* $P_\mathcal{C}(s_\mathcal{C}, a_\mathcal{C}, s'_\mathcal{C}) > 0$ *then* $s'_\mathcal{C} \in S_\mathcal{C}$ *and* $P_\mathcal{C}(s_\mathcal{C}, a_\mathcal{C}, s'_\mathcal{C}) = P(s, a, s')$*. An AMEC is the largest such End Component, whose* $S_\mathcal{C}$ *abides by* $J_{p_i} \notin S_\mathcal{C}$ *and* $K_{p_i} \in S_\mathcal{C}$ *for some* (J_{p_i}, K_{p_i}) *in* Acc_p*.*

With the set of AMECs found, an immediate reward takes a value of 1 at the transition into AMECs, and 0 in all other cases. During initialization, each state is assigned with a value of 0, 1, or $(0, 1)$ depending on whether it belongs to the accepting set, the blocking set, or the rest. In essence, the probability of entering the AMECs is equivalent to the probability of satisfying the corresponding LTL specification, so an optimal policy being extracted with the Product MDP and AMECs reaching probability can be suitably mapped back to one for the original system MDP and LTL specification [4]. By such, Challenge 2 is also overcome. Remarkably, [4] applies robust dynamic programming to consider uncertainties in the given system MDP. For instance, Fig. 2 illustrates learned results from the same LTL specification yet with two ways of uncertainty modeling:

$$\phi = home \wedge \Diamond\Box home \wedge \Box\neg unsafe \wedge \Box(R1 \wedge \Box(R2 \wedge \Box R3)). \quad (5)$$

In English, this means "sequentially visit R1, R2, R3 while always staying safe, and return to home once having visited all three regions."

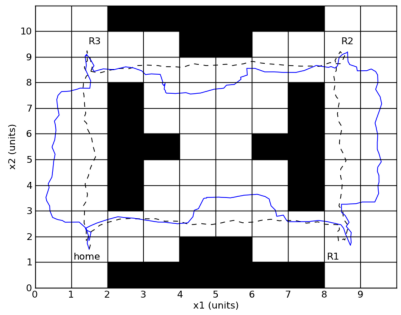

Fig. 2. Sample trajectories from two ways of uncertainty modelling, both successful in terms of task completion [4].

As a pioneering work, the paper [4] has set up a common framework to approach Problem 1, yet there is indeed room for improvement. First, the assumption of known system transitions does not hold for general control tasks. Second, the construction of AMECs requires that the graph model of the Product MDP is known, and that the complete state space can be explored [10].

Similar to [4], [8] establishes a Product MDP to encapsulate the system MDP and the LTL formula. However, they find that a policy maximizing the reaching probability of AMECs might not generate a uniquely optimal policy for the corresponding LTL formula. This is because a policy derived from the reaching probability concerns only the transient system behavior before entering the AMECs, whereas a policy to satisfy an LTL formula should regard both the transient and the long-term system behavior. Hence, inspired by the Average Cost Per Stage problem, they further formulate an optimization problem on the long-term system behavior while enforcing the satisfaction of the LTL formula with the AMECs. It is shown in their experiments that different criteria such as the expected time between the TLs events or the total distance travelled by a navigating robot optionally constitute the cost function in

the optimization problem. Although the improvement over [4] is recognizable, the applicability of the framework to real-world problems is still hindered by the strong assumption of known system dynamics. Consequently, we turn our focus to approaches without such assumption.

2) Learnt System Model: As the community has yet the tendency of perceiving Problem 1 in the fashion of probabilistic model checking, what immediately follow are the model-based RL approaches. These include [9] and [10].

[9] confronts Challenge 1 and 2 similarly to the previous work in [4] and [8], but maintains an estimated model for the system MDP. By complementing model approximation and policy optimization, it is claimed that the entire state space is explored and the converging policy is sufficiently close to the optimal one with high probability.

On the contrary, being the first work to exactly mention RL in the literature, [10] defines no AMECs but a reward function based on the Rabin accepting conditions Acc_p of the Product MDP $\mathcal{M}_p$ to resolve Challenge 2. Precisely, a positive reward is given if the updated product state s'_p falls in K_{p_i}, a negative reward if J_{p_i}, and 0 otherwise. An experiment of "liveness" and "safety" properties is displayed in Fig. 3, where the agent traverses between region A and B, and avoids all region C's.

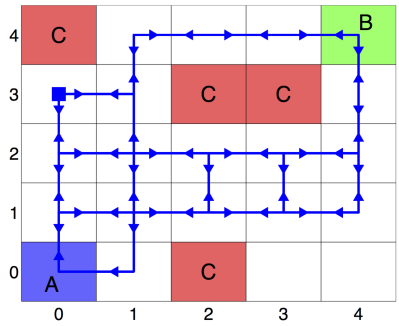

Fig. 3. An experiment demonstrating the accomplishment of "liveness" and "safety" properties using learnt system models [10].

Despite several successful paradigms recording "priority" (Fig. 2), "liveness" (Fig. 3), and "safety" (both Fig. 2 and Fig. 3), a major drawback of these aforementioned methods lies in the burden of constructing a Product MDP and forming AMECs or a reward function from the accepting conditions. In addition, these methods commonly answer only Problem 1 by considering sparse rewards such as the reaching probability or discretized real values. This is presumably the reason why the examples shown are rather simple in terms of system complexity and TL requirement. Furthermore, it might be true that maintaining a system model helps state exploration and increase sample efficiency, but a poorly learned model with biases is prone to bad performances too.

B. Model-Free Approaches

In the subsequent section, we shall see several pieces of work dismissing a given or learnt model, yet handling neatly Challenge 1 and 2.

1) Value Iteration: The first paper to our knowledge laying down a milestone is [5], in which Aksaray *et al.* propose to learn optimal policies using standard Q-learning with STL instead of LTL. As pointed out, STL not only allows for qualitative analysis but also robustness degree quantization with respect to system specifications. Due to the time-bounded characteristics of STL, the construction of a Product MDP becomes rather difficult. Instead, τ-MDP is used to resolve state history-dependencies, i.e. Challenge 1. Essentially, original states over τ time steps are concatenated and regarded as a single input to the τ-MDP, where τ depends on the time horizon of the given STL specification. Then, as a solution to Challenge 2, *log-sum-exp* approximation is performed to calculate numerically the robustness degree. This serves as the immediate reward at each iteration in standard Q-Learning for Problem 2. As for the satisfying probability in Problem 1, a further indicator function is prefixed, so that only 0 or 1 is given as immediate rewards.

Thanks to such manipulation with the semantics of STL, several experiments can be done to compare outcomes of Problem 1 and 2, concerning the specification:

$$\phi = \square_{[0,12]}(\lozenge_{[0,2]}(region\ A) \wedge \lozenge_{[0,2]}(region\ B)). \quad (6)$$

The above formula says, "Alternate between region A and B every two seconds, globally over the period from time 0 to 12." Fig. 4 shows the distinctive results from the original paper. It is obviously seen that the policy extracted with Problem 1 fails the "liveness" task, whereas the one with Problem 2 succeeds. The conclusion is then drawn that since robustness degree delivers partial credits to actions approaching the formula's satisfaction, the agent attains more guidance in the course of learning and hence the better performance.

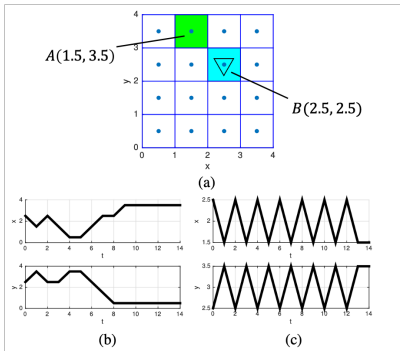

Fig. 4. (a) Simulation state space. (b) Trajectory maximizing probability of satisfaction. (c) Trajectory maximizing robustness degree [5].

Other prominent Value Iteration-related work include [11] [12] [13]. Noticeably, in [11] [12], the authors translate LTL specifications into Limit-Deterministic Büchi Automata (LDBA) instead of DRA. The difference of a LDBA and a DRA mainly sits in the accepting conditions. Compared to Definition III.1, a LDBA keeps only K_i for $i = 1, ..., N$ in the accepting conditions Acc and accepts a run $\sigma = q_0, q_1, q_2, ...$ if there is some K_i being visited infinitely. It is verified also in [14], that a DRA may fail to generate a uniquely optimal policy, and that

using a LDBA decreases the size of the Product MDP, speeds up the convergence rate, and permits easier reward assignment. With a LDBA-composed Product MDP managing Challenge 1, [11] collects rewards conventionally as [10] does (see III-A2), while [12] conceives a novel discounting reward function. By contrast, [13] adheres to translating LTL to DRA but establishes no Product MDP. They instead examine the LTL specification completeness by checking system MDP states s against DRA states q with the labelling function L, and then assign real-valued rewards of four classes accordingly. Furthermore, they transform the common Double Deep Q-Network (DDQN) into a finite-sum non-convex optimization problem, arguing for a more stable performance. These papers have demonstrated multiple degrees of success, but they are all confined in discrete state and action spaces, urging us to the next category of algorithms.

2) Policy Search: Regardless of the approving results in [5], two disadvantages of STL are pointed out [1] [15]. First, as the semantics indicates, STL specifications require boundaries on time modalities. This might be useful for monitoring system signals, but not for regulating robot behaviors. Looking at the example in Fig. 1, the instructor cares more about whether the goals are visited in the correct order and less about the exact time at which each goal is visited. Second, together with time boundaries, τ-MDPs scale badly with respect to system states and specification lengths.

Instead, Li *et al.* [1] [15] advocate Truncated Linear Temporal Logic (TLTL), which modifies LTL to favor only state trajectories of finite lengths. Such trajectories are conveniently taken from the state sequences of the entire episodes in a learning process. Now with an episodic trajectory, robustness degree is naturally evaluated against the given TLTL specification, as put in Problem 2 indeed. This induces the more principled policy search algorithms, and solves candidly both Challenge 1 and 2. For instance, [1] leverages Relative Entropy Policy Search, and [15] Gradient-Based Policy Search. Moreover, with a parametrized policy, such as a linear Gaussian function, continuous states and action spaces are enabled. The authors have tested their framework on a real-world robot, and shown unprecedentedly comparison studies against heuristic rewards. For example, in Fig. 5, the robot is programmed to reach a slot on the toaster and open the gripper when the slot is near so as to place the toast successfully.

A comparable work [16] implements their Policy Search algorithm using multiple neural networks (NNs). With each NN tracking a subtask in their original specification-translated Task Monitor, fundamentally an automaton, a certain form of "memory" is said to be substantiated. In addition, a reward shaping process is utilized to accumulate denser rewards throughout the learning processes. Experiments are conducted to indicate a better convergence over [1], yet only in simpler simulations. It is questionable if the NN policies also converge well on real-world robots since these challenging scenarios usually exhibit severe non-convexity and demand much more training data. Nonetheless, an online toolbox (https://github.com/keyshor/spectrl_tool) is made available for

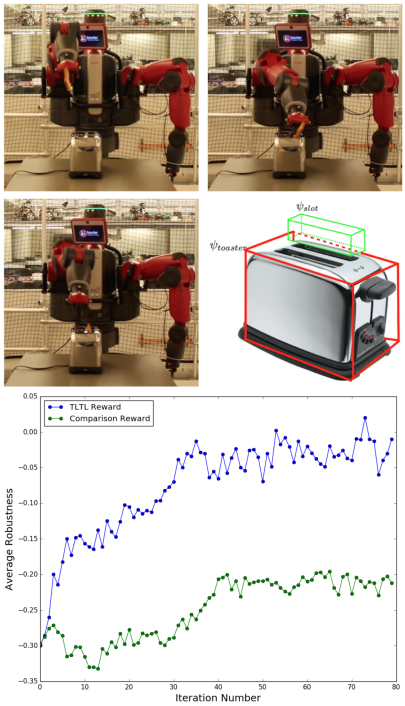

Fig. 5. (Up) Photos taken during the real-world experiment showing a successful trial. (Down) A comparison between TLTL-based and heuristic rewards [1].

stipulating customized TL requirements and training an RL agent.

To conclude Policy Search methods, in spite of the outstanding result in [1], it is set back by the fact that a certain degree of manually tuning has to be done to balance the weights between events in the TLTL specification. To elaborate using Fig. 5, the robustness values of avoiding obstacles and closing the grippers have to be normalized through empirical trials. Likewise, Policy Search methods are subject to convergences to local optima and high variances of estimated episodic rewards.

3) Actor-Critic: A very recent paper [17] has been noticed, which avoids the two aforementioned issues, namely manual normalization and high variance. Still considering the fundamental Challenge 1 and 2, a LDBA-based Product MDP is constructed in a similar fashion to [11] [12], but a seminal reward shaping technique is employed. In consequence, at each time step a numeric value which requires no manual weighting is delivered to the agent. Apart from handling Challenge 1 and 2 neatly, they apply the powerful Deep Deterministic Policy Gradient (DDPG) algorithm, which no longer exhibits large variance in results as [1] [15] [16] potentially do. Interested readers are deferred to the original paper for formal definitions. We show here an exceptional case study in Fig. 6, where the following LTL specification is endeavored:

$$\phi = \Diamond(a \wedge \Diamond d) \vee \Diamond(b \wedge (\neg c \,\mathrm{U}\, d)). \tag{7}$$

In plain texts, (7) triggers the robot to a or b first. If a is reached first, then d is designated without other restrictions.

If b is reached first, then d has to be visited while avoiding c.

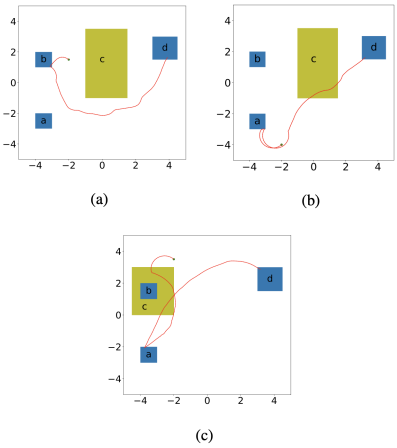

Fig. 6. (a) Initialized around b, the robot visits b first and then circumvents c towards d. (b) Initialized around a, the robot visits a and then d. (c) Initialized around b yet perceiving c is by no means avoidable after visiting b, it rather visits a and then d [17].

Equipped with many plausible features, including an actor-critic learning structure, continuous states and actions, and a dense reward function, [17] has indeed attained a promising performance. However, whether the reliance on constructing a Product MDP and reward shaping allows for real-world applications with more sophisticated dynamics remains a topic to be inspected deeper.

C. Hierarchical Approaches

Thus far, we have seen papers mostly handling tasks with one TL specification. In this last subsection, a group of algorithms able to handle multiple TL formulas at once is examined. Icarte *et al.* publish lately three papers along this research line [18] [19] [20]. They take advantage of Non-Markovian Reward Decision Processes (NMRDPs) and propose the so-called Reward Machine (RM), essentially a Mealy Machine, as a realization of the NMRDP. Unlike MDPs, NMRDPs forgives the Markov property and formulates the reward function with sequences of states, posing no longer Challenge 2. In addition, as a Product MDP assists the system MDP, an RM helps memorize the TL requirements for the NMRDP and outputs an appropriate reward function at each time step, thereby tackling Challenge 1. We denote here the two-step construction of an RM from multiple specifications, and keep the formal definitions within the original papers.

Step 1: (Translation to Deterministic Finite Automata (DFA)) Translate each specification ϕ_i into a DFA $\mathcal{A}^{(i)}$, which is similar to a DRA or a LDBA yet only accepting a run σ if the end state q_n achieves $K \subseteq Q$ (compare III.1). As requiring the run σ to be finite, only safe and co-safe specifications are considered here (see Section 3.1.4 in [18] for explanation).

Step 2: (Integration to RM) Compound all DFA $\mathcal{A}^{(i)} = \langle Q^{(i)}, \sum, \delta^{(i)}, q_0^{(i)}, K^{(i)} \rangle$ into an RM $\mathcal{M}_R = \langle Q_R, \sum, \delta_R, q_{R_0}, \zeta_R \rangle$ with a state space $Q_R = Q^{(1)} \times ... \times Q^{(N)}$, an initial state $q_{R_0} = (q_0^{(1)}, ..., q_0^{(N)})$, the transition function $\delta_R(q_R, \alpha) = (\delta^{(1)}(q^{(1)}, \alpha), ..., \delta(q^{(N)}, \alpha))$, and the reward assignment function $\zeta_R(q_R, \alpha) = \sum_{i=1}^{N} \zeta^{(i)}(q^{(i)}, \alpha)$, where $\zeta^{(i)}(q^{(i)}, \alpha)(s, a, s') = r_i \cdot \mathbb{1}(\delta^{(i)}(q^{(i)}, \alpha) \in K^{(i)})$.

As seen, an RM has the capability to accommodate a number of TL formulas and construct an overall product state space across them. Utilizing the off-policy nature of Q-learning, [20] builds an efficient experience buffer with learning tuples not only across time steps, but also across TL tasks. Fig. 7 shows their experiment domain and marks a superior learning outcome of Q-learning on RM. Please see [20] for a complete list of 10 task specifications, which consists of various operations such as fetching natural resources or building a bridge in the Minecraft world.

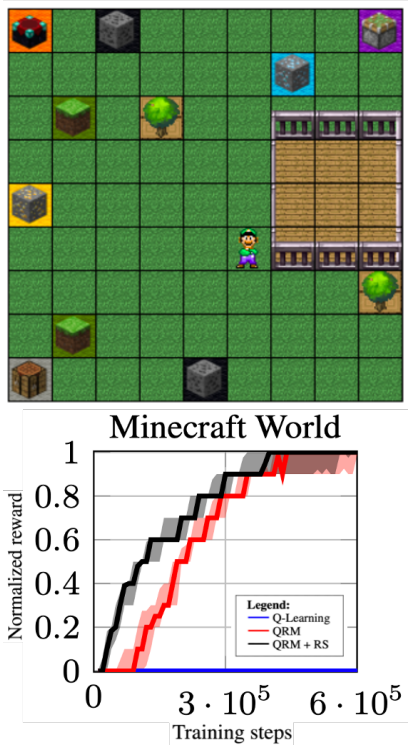

Fig. 7. (Up) Minecraft world domain. (Down) Learning curves from standard Q-learning, Q-learning on RM, and Q-learning on RM with reward shaping [20].

It is noteworthy that with the aid of RM, skill decomposition is learnt and multiple tasks are achieved, in comparison to failures from standard Q-learning. Additionally, the effect of reward shaping is again emphasized. An online toolbox is provided at https://bitbucket.org/acamacho/fl2rm, offering the functionality of translating TL formulas into DFAs, and integrating them into a final RM for training. However, similar to cases with Product MDPs, an RM might be cumbersome to build. This limits the applications to simpler TL requirements, and the Q-learning algorithm limits them to discrete state and action spaces.

IV. Discussion

Having overviewed a comprehensive literature, we first present Table I as a summary of the strengths and weaknesses from exemplary approaches, and then we suggest prospective directions for future research.

TABLE I
STRENGTHS AND WEAKNESSES OF EXEMPLARY APPROACHES

Method	Strengths	Weaknesses
Model-Based [4], [10]	· Pioneering paradims	· AMEC preproccessing · Sparse rewards
Value Iteration [5]	· Qualitative vs. Quantitative	· τ-MDP construction · Discrete states and actions
Posicy Search [1]	· Continuous states and actions · Robustness degree · Real-world robot	· Manual normalization · Credit assignment problem
Actor-Critic [17]	· Continuous states and actions · Reward shaping	· Product MDP construction
Hierarchical [20]	· Multitasking and decomposition · Reward shaping	· RM construction · Discrete states and actions

From Table I, a few characteristics are clearly desirable and thus advocated. These includes permitting continuous states and actions, delivering dense reward, and avoiding heavy construction of automata. All together, these characteristics point towards a potential framework in which an actor-critic mechanism serves as the RL backbone and a fitting model is integrated for trajectory sampling as well as reward calculating. In particular, at each iteration, a system model is approximated and a state trajectory is predicted with the model. The trajectory is in turn evaluated against the TL specification as a critic feedback, and then an actor updates accordingly in an off-policy manner. In spired by [20], we believe off-policy learning is beneficial for subtask decomposition or even multitasking with TL requirements. By such, both Challenge 1 and 2 are eliminated without the need of Product MDPs. Additionally, both Problem 1 and 2 are feasible as computing qualitative semantics with reward shaping and quantitative semantics directly are both manageable. However, as seen in the model-based frameworks, such method demands a strong model function and possibly heavy computation to attain a good learning and control performance, especially in a highly dynamical environment such as motion planning at urban intersections. Undoubtedly, the suggested framework requires further inspection and is merely an attempt to stimulate further discussion in the research community.

V. Conclusion

In this paper, we have presented principles of Reinforcement Learning (RL) and Temporal Logics (TLs) including LTL, STL, and TLTL. We have also investigated a broad variety of methods that formulate TL specifications as reward functions in RL frameworks. In specific, two technical problems are addressed, respectively making use of qualitative and quantitative semantics of TLs. Additionally, two notable challenges are observed and emphasized across the paper to identify several good solutions. These include model-based, Value Iteration, Policy Search, Actor-Critic, and hierarchical approaches. Lastly, a comparison of the presented solutions and promising future directions are discussed. In conclusion, we have seen TLs showing crucial advantages over typical heuristic rewards for high-level properties such as "liveness" and "priority." Hence, as an encouraging remark, we prompt the society for further work in this direction.

Acknowledgment

This work is done in a seminar course, Cyber-Physical Systems, at the Department of Informatics, Technical University of Munich. The author thanks his advisor M.Sc. Xiao Wang and the course coordinator Prof. Dr.-Ing. Matthias Althoff gratefully. Their guidance throughout the course is very helpful and greatly appreciated.

References

[1] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. 2, 3, 6, 8

[2] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008. 2, 3

[3] L. Busoniu, *Reinforcement learning and dynamic programming using function approximators*. CRC Press, 2010. 2

[4] E. M. Wolff, U. Topcu, and R. M. Murray, "Robust control of uncertain markov decision processes with temporal logic specifications," *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, 2012. 3, 4, 5, 8

[5] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016. 3, 5, 6, 8

[6] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems*, K. Chatterjee and T. A. Henzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–106. 3

[7] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan, *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*. Cham: Springer International Publishing, 2018, pp. 135–175. 3

[8] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of markov decision processes with linear temporal logic constraints," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014. 4, 5

[9] J. Fu and U. Topcu, "Probably approximately correct mdp learning and control with temporal logic constraints," *Robotics: Science and Systems X*, 2014. 4, 5

[10] 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," *53rd IEEE Conference on Decision and Control*, 2014. 4, 5, 6, 8

[11] M. Hasanbeig, A. Abate, and D. Kroening, "Logically-constrained reinforcement learning," *arXiv: Learning*, 2018. 5, 6

[12] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," 2019. 5, 6

[13] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019. 5, 6

[14] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Omega-regular objectives in model-free reinforcement learning," *Tools and Algorithms for the Construction and Analysis of Systems Lecture Notes in Computer Science*, p. 395–412, 2019. 5

[15] X. Li, Y. Ma, and C. Belta, "A policy search method for temporal logic specified reinforcement learning tasks," *2018 Annual American Control Conference (ACC)*, 2018. 6

[16] K. Jothimurugan, R. Alur, and O. Bastani, "A composable specification language for reinforcement learning tasks," in *Advances in Neural Information Processing Systems 32*, 2019. 6

[17] C. Wang, Y. Li, S. L. Smith, and J. Liu, "Continuous motion planning with temporal logic specifications using deep neural networks," 2020. 6, 7, 8

[18] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Teaching multiple tasks to an rl agent using ltl," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems AAMAS)*, 2018. 7

[19] ——, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018, pp. 2112–2121. 7

[20] A. Camacho, R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Ltl and beyond: Formal languages for reward function specification in reinforcement learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 6065–6073. 7, 8