Literary Review: Reinforcement learning and Logic

Jonathan Azpur
York University
Lassonde School of Engineering
jonaac@yorku.ca

1. Introduction

In the past 3 to 4 years we have seen a rise in the number of researched focused on the use of logic (linear temporal logic, modal temporal logic, etc.) as a tool to improve and assist traditional Reinforcement Learning. This field of study captured my attention and I think it could be a good subject for my Master's thesis, so I decided it would be a good idea to dive deeper into this subject. In this project I will be doing a literary review on the current research that covers the conjoint topic of Reinforcement learning and Logic

This project has two main goals. First, it is meant to critically evaluate and provide an overview of the significant literature published on this topic. Second, this project will help me identify any potential gaps within the context of existing literature, and assist me in defining potential research topics that I will attempt to address on my future thesis.

I will go over all the research that I put together for this project, give an overview of their contents, and critically evaluate the research. I divided the body of work into three main topics; Safe Reinforcement Learning (Section 2), Improving RL performance (Section 3) and Generalization of RL policy learning (Section 4). I added some more research papers (Section 5) that I examined but did not go in-depth. Finally, I will identify any potential gaps within the context of existing literature and define potential research topics that I will attempt to address in my future thesis (Section 6).

2. Safety of Reinforcement Learning agents

The papers in this section are mainly focused on the use of Logic programming to assist Reinforcement Learning on making sure that the agent is safe during training, deployment, or both.

2.1. Li, Xiao et al. 2019 [14]

The first research paper I will be discussing is the on developed by Li, Xiao et al. (2019). The goal of this research paper is to try and find a solution to two of the main issues with Reinforcement learning, complex task specification and safe exploration. The authors propose a system that will use syntactically co-safe Truncated Linear Temporal Logic (scTLTL), which is a restricted version of truncated linear temporal logic, and its equivalent Finite State Automata (FSA). First, it will provide reward to the RL agents. Second, the system will perform goal selection for the control Lyapunov function (CLF) in order to perform guided exploration. Finally, it will define sets of safe states for the control barrier function (CBF) to keep the agent safe.

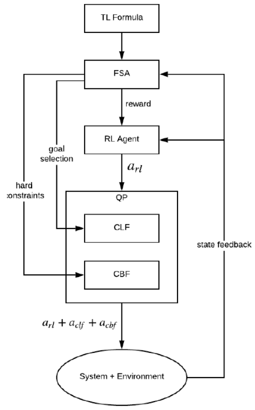

Figure 1. Diagram of the proposed TL guided safe reinforcement learning system

The system proposed has five main components and can be seen in Fig1. The scTLTL formulas, an FSA that represents the scTLTL formulas, the Reinforcement Learning agents, the Quadratic Program which in it of itself contains the CLF and the CBF and the environment.

The FSA will provide an augmented reward to the RL agent to promote the avoidance of potentially unsafe states given its current state. The reward is a follows:

$$\tilde{r} = min(\rho(s', D_q^{\phi}), c_r\rho(s', \neg\varphi_{q,q_{trap}})) \tag{1}$$

Where $\rho(s', D_q^{\phi})$ promotes the agent to progress from the current FSA state and $\rho(s', \neg\varphi_{q,q_{trap}})$ penalizes the agent

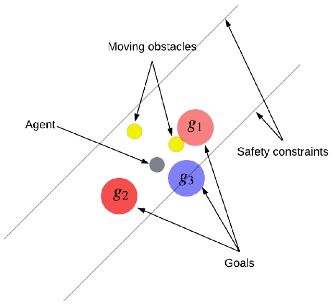

Figure 2. Simulation environment

for going towards the unsafe (or trap) state. Given this reward the agent will select the appropriate action a_{rl}.

The FSA will also be used for goal selection. The idea is to find the edge that is easiest for the agent to activate and then find the MDP state that maximizes the chance of completing the predicate guarding said edge. This can be formalized as:

$$s_g(q) = \underset{s \in S}{\operatorname{argmax}}\, \rho(s, \psi_{q,q'_{\rho_{max}}})$$
$$q'_{\rho_{max}} = \underset{q' \in \Omega^q, q' \neq q_{trap}}{\operatorname{argmax}} \quad \rho(s, \psi_{q,q'})$$

Then, the CLF can select an action a_{clf} for the agent to complete its goal. The FSA will also help define safety strategies by providing hard constraints to the CBF. The goal is to avoid the predicate $\varphi_{q,q_{trap}}$ that will take us to the unsafe state, or formally they want $\neg\varphi_{q,q_{trap}}$ always to be true. So, they restructure $\neg\varphi_{q,q_{trap}}$ to its conjunctive normal form $\neg\varphi_{q,q_{trap}} = \bigwedge_i (\bigvee_j \psi_{i,j})$ and extract the set of states that are unsafe from this predicate.

$$C_\phi(s,q) = h^0_\phi(s,q), \ldots, h^i_\phi(s,q).\text{where}$$
$$h^i_\phi(s,q) = \psi_{ij_{\rho_{min}}},\ j_{\rho_{min}} = \underset{j}{\operatorname{argmin}}\, \rho(s, \psi_{ij})$$

Then, the CBF can select an action a_{cbf} to promote keeping the agent safe.

This framework allows the system to execute a mostly traditional model-free Reinforcement learning algorithm, except for the fact that the state feedback from the environment depends on the sum of the actions a_{rl}, a_{clf} and a_{cbf} selected in each iteration, and the reward feedback from the environment will have the FSA reward $\tilde{r}$ added to it. The algorithm is shown to be able to learn an optimal policy

In order to evaluate this novel system Li, Xiao et al. deployed it in an environment consisting of an agents that moves in a 2D world that has to reach three location in a specific order while also avoiding a number of moving object, Fig 2.

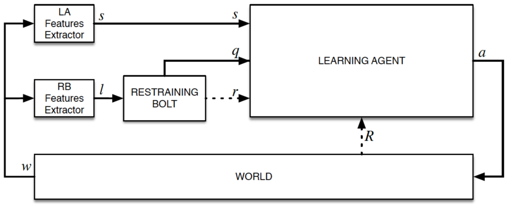

Figure 3. Learning Agent and Restraining Bolt

The results show that with the proposed framework the agent is capable of learning a policy quicker, can learn how to avoid unsafe scenarios and has a higher success rate.

2.2. De Giacomo, et al. 2019 [12]

Restraining bolts (RB) are a device that restricts an agents actions when connected to its systems. Restraining bolts are used to limit actions to a set of desired and safe behaviours. De Giacomo et al. propose combining RB, expressed as logical specification in LTL$_f$ / LDL$_f$, with RL to ensure the agent's safety. Incorporating RB to the overall system introduces a new problem, where now there is two distinct representations of the environment, one by the agent and one by the restraining bolts. So the goal is for the agent to learn a optimal policy while shaping its goals to suitably conform to the restraining bolt specifications even though they are not expressed in the same terms.

The system proposed has three main components and can be seen in Fig3. The RL agent, the restraining bolts that represents the LTL$_f$ / LDL$_f$ formulas, the feature extractors for both. The learning agents is modelled by a MDP $M_{ag} = \langle S, A, T_{r_{ag}}, R_{ag} \rangle$ and the restraining bolt is formally defined as $RB = \langle \mathcal{L}, \{(\varphi_i, r_i)\} \rangle$ where L is the set of possible fluent configurations and $\{(\varphi_i, r_i)\}$ is a set of restraining specification with φ_i being an LTL$_f$ / LDL$_f$ formula and r_i the reward associated with φ_i. The RL problem with LTL$_f$ / LDL$_f$ restraining specifications can therefore be defined as the pair $M^{rb}_{ag} = \langle M_{ag}, RB \rangle$

It is important to note that the use of LTL$_f$ / LDL$_f$ makes the whole system non-Markovian, so the solution to the RL with LTL$_f$ / LDL$_f$ restraining specifications is a policy $\rho : (Q_1 \times \cdots \times Q_m \times S)* \longrightarrow A$ that maximizes the expected reward. The solution can be obtained by reducing M^{rb}_{ag} to an NMDP $M^{ag}_n = \langle S \times \mathcal{L}, Tr^{rb}_{ag}, \{(\varphi_i, r_i)\} \cup \{(\varphi_s, R_ag\} \rangle$. Because the LTL$_f$ / LDL$_f$ formulas can be transformed into a deterministic finite state automata (DFA) it allows them to transform the NMDP into an equivalent MDP over an extended state space. Given this equivalence, De Giacomo et al. provide a theorem and its prove stating that the optimal policy from the original problem can be found by learning the optimal policy from the new, inferred MDP. This theorem provides them with a technique to learn the optimal policy for RL with LTL$_f$ / LDL$_f$ restraining specification.

Figure 4. Breakout, Sapiento and Cocktail Party

To evaluate the soundness of the proposed system it is tested on three distinct environments (Fig. 4). First, Breakout, a video game where the player is in control of a paddle that moves from left to right at the bottom of the screen and has to drive a ball to hit all the bricks at the top of the screen. Second, Sapiento, an educational game for children where a small mobile robot has to be programmed to visit specific cells in a 5x7 grid. Third, the Cocktail Party experiment, based on a service robot in a cocktail party who has to serve drinks and snacks to people.

The results show that in all three environments the agent is capable of learning the optimal policy over time. It is shown that in all cases the agent is capable of increasing its reward and its in-game score as a function of the number of iterations. The main result of this paper is that, in spite of the separation between the two representation of the environment, under general circumstances, the agent can learn to act so as to conform as much as possible to the LTL$_f$ / LDL$_f$ specifications.

2.3. Hasanbeig et al. 2020 [7]

Reinforcement Learning deliver good training outcomes and have been proven to be able to optimally solve a decision-making problem without any prior knowledge about the MDP model, but given its dependence on the agents experience (which are obtained by exploring their environment) and the ergodicity assumption (which claims that any state can be reached by any other state given the appropriate policy) RL lacks the ability to guarantee that an agent will not enter an unsafe state. Even though this may be acceptable in most cases, in scenarios such as safe-critical physical systems exposing the agent to unsafe states is not viable methodology. For example, if you are training a robot to fly a helicopter it is not viable to allow the robot to experience crashing the helicopter.

In order to address the lack thereof any sort of safety guarantee in RL, Hasanbeig et al. propose the use of Cautious RL, a safe exploration scheme for model-free RL. The algorithms are based on the idea of the agent having a limited knowledge of its own dynamics. The agent starts by performing exploratory cautious actions, and gradually, in line with the growing confidence about the environment obtained from observations, the range of acceptably safe actions grows, and the uncertain component of the dynamics becomes known. In addition Cautious RL will use LTL formulas in order to specify tasks for policy synthesis in RL, and it will automatically provide reward shaping and task decomposition for complex tasks. This will allow the algorithm to predict unsafe state-action pairs (safe padding) to limit the range of the agents exploration while policy learning for LTL task satisfaction. It is shown that the method guarantees asymptotic results.

In order to combine the use of Reinforcement Learning with LTL formulas the authors combine the MDP representing the RL problem $\mathcal{M} = (S, A, s_0, P, AP, L)$ with the limit-deterministic generalized Buchi automaton LGBA $\mathcal{U} = (Q, q_0, \Delta, \Sigma, F)$ corresponding the LTL formulas in a single entity called a Product MDP:

$$\mathcal{M} \otimes \mathcal{U} = (S \times Q, (s_0, q_0), A, P, Q, L^{\otimes}, F) \tag{2}$$

where $L^{\otimes} : S \times Q \to 2^Q$:

In addition, the Hasanbeig et al. introduce their state adaptive reward function:

$$R(s^{\otimes}, a) = \begin{cases} r_p & \text{if } q' \in \mathbb{A},\ s^{\otimes} = (s', q') \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbb{A}$ is the accepting frontier set. The reward function will provide a positive reward if the agent is getting closer to the goal states in the LGBA, and a neutral reward otherwise. Therefore, guiding the agent to visit the accepting sets infinitely often, and consequently, satisfy a given LTL property φ.

To ensure safe exploration they create a safe padding by defining a set of safe states. Let the current state of an agent be s, the set of safe states is defined as:

$$O_{safe}(s) = \{x \in O(s), q \to q' \notin \Delta_{sinks}\} \tag{4}$$

The product MDP (2), the state-adaptive reward function (3) and the safe padding (4) will allow Hasanbeig et al. to develop a learning architecture, Cautious RL, capable of generating safe policies that satisfy a given LTL formula. Cautious RL is based on the Q-Learning algorithm, but it has a double learner architecture made off of an optimistic and pessimistic learner.

The optimistic learner will employ Q-learning to maximize the expected cumulative reward. The q-function will be updated by following the rule:

$$Q(s^{\otimes}, a) = Q(s^{\otimes}, a) + \mu[R(s^{\otimes}, a) + \gamma \max_{a' \in A^{\otimes}} Q(s^{\otimes\prime}, a') - Q(s^{\otimes}, a)] \tag{5}$$

The authors show that the optimistic learner will converge to an optimal action-value function Q^* and therefore one will be able to generate an optimal policy pi^* by selecting the action that returns the highest Q^*:

$$\pi^*(s^{\otimes}) = \arg \min_{a \in A^{\otimes}} Q^*(s^{\otimes}, a) \tag{6}$$

The pessimistic learner will use safe padding to generate the set of safe actions:

$$A_p(s^\otimes) = \{a \in A^\otimes : U(s^\otimes, a) < p_critical\} \tag{7}$$

The set will be sorted in from the action with the lowest to the highest, where is the maximum probability of violating a safety constraint and will select the top k actions based on:

$$a^* = \underset{a \in A_p[1:k]}{\operatorname{argmax}} Q(s^\otimes, a) - r_p U(s^\otimes, a) \tag{8}$$

where k is determined by a monotonically increasing function given the number of visitation at the agent's current state, and r_p is used to balance Q and U.

To test the Cautious RL algorithm Hasanbeig et al. test their algorithm in two environment. A slippery grid world, where the agent has to reach a highlighted area without falling on the slippery areas, and the video game Pacman. The results show that in both cases the safe padding provides a higher success rate.

Literary Review - Safety

The papers discussed provide different perspective on how to combine Reinforcement Learning with different types of Logic programming and formal logic languages to ensure the agent's safety during training, deployment, or both. The results of these papers are promising and demonstrate the potential of their methods, but in my opinion, there are some aspects of their research work that could be improved.

The system presented by Li, Xiao et al. [14] integrates a high number of hyperparameters to tune manually. The performance is heavily influenced by these parameters. To ensure the scalability and the potential generalization of this approach it would be interesting to try and update the methods for it to alleviate their burden. Also, when taking a look at the results we see that the algorithm using only the CBF has a higher success rate than the algorithm using CLF+CBF, which intuitively shouldn't be the case. There is no discussion as to why this could be and therefore misses an opportunity of further understanding the dynamics of both control functions and if there should be some sort of variable that manages the effect of the action value chosen by each control function. Finally, the proposed method seems to be computationally very expensive. It takes a lot of iteration for the algorithms to converge. Even though we see that the novel algorithm seems to perform better than a traditional RL algorithm it would be interesting to compare its performance with other logic based RL algorithms used for safety.

The evaluation methods used by De Giacomo, et al. [12] seems to be incomplete in my opinion. The only thing that can be said given the data is that their algorithm will learn a policy over time. There is no frame of reference on how good the algorithms actually is. For example, in order to see the potential usefulness of the algorithm it would be best to also compare it with a baseline RL algorithm and confirm that the approach is capable of outperforming the baseline in terms of the agents safety. Also, there is not mention on the computational cost of the proposed framework. Computational cost is an important feature to be aware of when working with RL because due to its necessity to perform random exploration to learn policies it has sample efficiency problems and we would ant to make sure that the proposed algorithm doesn't make it worse. Finally, there is no mention of the potential to scale the algorithm to more complex safety barriers and reward structures. This should be an important feature for considering the viability of a novel algorithm in future work.

In the work done by Hasanbeig et al. [7] they argue for the importance of a RL methods that provides some sort of safety guarantee during its training, like in safety-critical physical systems. When they perform their evaluation of the algorithm none of the scenarios necessarily need a safety guarantee and even though it proves that the proposed algorithm does keep the agent safe the method is not fully testing the robustness of their algorithm in a safety-critical physical system given the motivation of this specific scenario. Also, they introduce the notion of a double learner, where a optimistic learner looks to maximize the cumulative reward and a pessimistic learner looks to keep the agent safe, but there isn't much of a focus on the possible trade-off between the optimistic and pessimistic learner, which seems like an important aspect of the algorithms that wasn't covered.

3. Improving RL performance

The papers in this section are mainly focused on the use of Logic programming to assist Reinforcement Learning to improve its performance.

3.1. Toro Icarte et al. 2019a [9]

Finding optimal policy using model-based reinforcement learning can require a lot of exploration in the environment. Sometimes this can be of very high cost and in some cases not possible to perform, such a physical environments. To solve this issue, Toro Icarte et al. (2019a) propose the use of advice to guide the agent through a more efficient exploration. They define advice as recommendations regarding behaviour that may describe suboptimal ways of doing things, may not be universally applicable, or may even contain errors. The authors of this paper use Linear Temporal Logic (LTL) to provide the agent with advice and propose a customized version of the R-MAX RL algorithm that will have the ability to be guided by the advice.

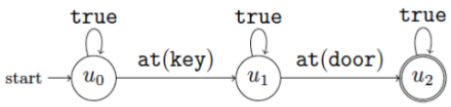

Figure 5. Example of an NFA

Linear temporal logic is the preferred formal logic language to represent advice because it allow to describe a set of action over time. In this case, the idea is to suggest the agent what actions to take over time. For example, take a look at the following LTL formula:

$$\Diamond(\text{at(key)} \wedge \bigcirc(\Diamond\text{at(door)})) \qquad (9)$$

can be understood as "the agent should eventually get to a state where it has the key and then eventually get to a state where it is at the door." which is basically advising the agent to make sure it has a key in order to open a door.

Any LTL formula can be converted into a Nondeterministic Finite State Automaton (NFA) such that a finite sequence of states will be accepted by the NFA if and only if they satisfies the LTL advice formula. The authors used a system previously developed by Baier and McIlraith [2] to transform the LTL formula into an NFA. Fig. 5 shows the NFA constructed from the advice shown in (12). The NFA will include dead-ends which signify that the agent has entered a state where the LTL formula can't be successfully completed any more. Even after falling into a dead-end, the advice may be useful to continue exploration. Therefore the advice will still be suggested once the agent leaves said state.

In order to follow any sort of advice, there has to be some way in which the agent can understand and apply it. To be able to do so, the authors propose that the agent have a background knowledge function $h_S : S \times A \times \text{lit}(\Sigma) \rightarrow \mathbb{N}$. This function will return an estimate of the number of actions the agent would have to take to reach a state $s\prime$ where the literal $l \in \text{lit}(\Sigma)$ is true. Intuitively it represents the agent's prior knowledge about how to make ground atomic formulae either true or false. Given the background knowledge function, Toro Icarte et al. (2019a) construct a new function $h : S \times A \times \mathcal{L}_\Sigma \rightarrow \mathbb{N}$ which is an extension of h_B that returns an estimate of the number of actions needed to satisfy an LTL formula $\varphi \in \mathcal{L}_\Sigma$.

In the described framework, to follow advice is equivalent to saying take the actions that will allow one to move through the NFA edge towards the accepting state. To do so, the authors want to identify which edges are "useful". By this they mean edges that will lead us to a path where one can reach an accepting state. So, Toro Icarte et al. (2019a) define a Advice Guidance Formula (AGF)

$$\hat{\varphi} \equiv \bigvee_{i=0}^{m} \left[\bigwedge_{(q,\beta,q\prime)\in useful(q^{(i)})} \text{to\_IBF}(\beta) \right] \qquad (10)$$

that will be used to give priority to actions that would take the agent closer to the accepting state, the AGF will be satisfied by any action that completes one of the formulas needed to transition over a useful edge.

The formula h can be used to rank how close each action is to making progress in satisfying the advice guidance formula $\hat{\varphi}$. In addition, it would be ideal to try and avoid dead-ends, and to this point the authors define a similar function called Advice Warning Formula

$$\hat{\varphi}_w \equiv \bigwedge_{i=0}^{m} \left[\bigvee_{q\in q^{(i)} \text{ and } (q,\beta,q\prime)\in\delta^{(i)}} \text{to\_IBF}(\beta) \right] \qquad (11)$$

indicate action that have the potential to reach dead-ends in the NFA. Now, $\hat{\varphi}_w$ will be used to define a set of action W that will potentially lead to dead-ends:

$$W(s) \equiv \{a \in A : h(s, a, \hat{\varphi}_w) \neq 0\} \qquad (12)$$

The idea is then for the agent to be guided by h and disfavour actions in W.

To solve the MDP Toro Icarte et al. (2019a) use the R-MAX RL algorithm. R-MAX is a family of RL algorithms in which its agent explores the environment by assuming that unknown transitions give maximal reward. To be able to incorporate the LTL advice they propose a variation of the R-MAX algorithm that plan towards the closest unknown transition that is not in W and has a minimum h. This methodology will make sure that the agent satisfies the advice and as proven by the authors on the paper will also converge to an optimal policy.

The algorithm is tested on two different environments. First a 25 × 50 grid world environment in with a door a key and nails around the room. The agent's goal is to grab the key, avoid the nails and open the door with the key. The second environment has the same elements, but with 4 concurrent 25 × 25 rooms in which the agent has to open the door for each room. In the first environment the algorithms that use positive advice converge to an optimal results quicker than the traditional R-MAX algorithm, and in the second environment the only algorithms that converge to optimal solution are the ones with that use positive advice.

3.2. Ali Payani and Faramarz Fekri 2020 [15]

Relational Reinforcement Learning is a variation of RL where the main idea is to describe the environment in terms of objects and relations. The RRL framework has four main advantages compared to the more traditional RL approach.

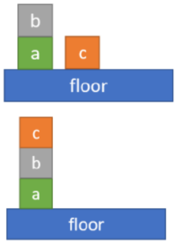

Figure 6. BoxWorld

First, the learnt policies are more interpretable. Second, the learnt policy can generalize better. Third, one can incorporate inductive biases into the learning. Finally, it allows for the incorporation of higher level concepts and prior background knowledge. On the other hand, RRL can't be used to work on complex environments. For example, RRL is not a viable solution when working with complex visual scenes and can't be used along deep learning structures. To solve this, the Payani et al. propose a novel deep RRL method based on differentiable Inductive Logic Programming (ILP) that can effectively learn from images and present the state of the environment as first order logic predicates while still being able take the expert background knowledge and incorporate it into the learning problem using appropriate predicates.

The proposed framework can be seen in Fig 8. First, the framework would process images through multiple convolutional layer in the CNN. The last layer of the convolutional network chain will then be treated as a feature vector and is usually augmented with some non-local information. This feature map is then fed into a relational learning unit which is tasked with extracting non-local features and obtaining an explicit representation of the state. Finally, the representation is fed to the dNL-ILP engine and it is tasked with selecting the desired actions.

The proposed system is tested in two different environments. First, the BoxWorld environment. It has been widely used as a benchmark in past RRL systems. The goal is to stack boxes on top of each other in a specific order, the dimension of the observation images is 64 × 64 × 3 and explicit relational information is not available for the agents. Second, the GridWorld environment. This environment is consisted of a 12 × 12 grid with keys and boxes randomly scattered. The agent must collect the key before accessing the box. When the agent has a key, provided that it walks over the lock box with the same color as its key, it can open the lock box, and then it must enter to the left box to acquire the new key which is inside. The agent cannot get the new key prior to successfully opening the lock box on the right side of the key box. The goal is for the agent to open the gem box.

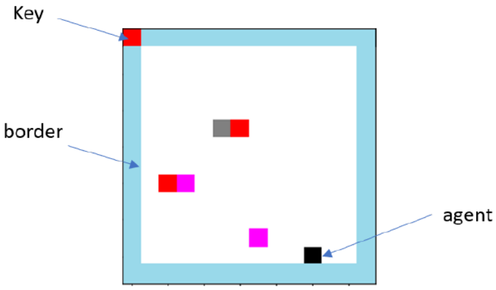

Figure 7. GridWorld

The algorithm is compared to A2C as a baseline algorithm. In the case of the BoxWorld environment the results shows that when tested with 4 boxes both models are able to learn a successful policy after around 7000 episodes. On the other hand, when tested with 5 boxes, the proposed approach converges after around 20K episodes whereas it takes more than 130K episodes for the A2C approach to converge, and even then it fluctuates and does not always succeed. In the case of the GridWorld environment the agent is tested in two variation of the environment, one with a dead-end brach and one without. In both cases the proposed approach can learn the solution in both settings very quickly. On the contrary, the standard deep A2C was not able to converge after 10^8 episodes.

3.3. Camacho et al. 2019a [5]

When it comes to Reinforcement Learning in some cases it may take the agent many interactions with the environment to learn from sparse rewards, and it can be challenging to specify reward functions that reflect complex rewardworthy behaviour. To address this issue Camacho et al. (2019a) propose the use of Reward Machines as a framework to represent RL reward functions in a normal form. They show that a reward specified in any number of formal languages (LTL, LDL, Golog, PLTL, Regular Expressions, etc.) can be translated into a Reward Machine and how the reward machine can be exploited by a tailored q-learning algorithm to improve the sample efficiency compared to traditional reinforcement learning algorithms.

A reinforcement learning agent does not have knowledge of the reward function. It is something that has to be provided by the user/programmer regardless of the environment. For the programmer, developing a reward function can be very challenging for two main reasons. On one hand, the states in the environment may not provide an intuitive representation for reward specifications (i.e. pixels). This may cause the states to not provide an adequate level of abstraction for the programmer to properly define the reward function. On the other hand, rewards may be a result of a complex temporally extended behaviour (i.e. opening

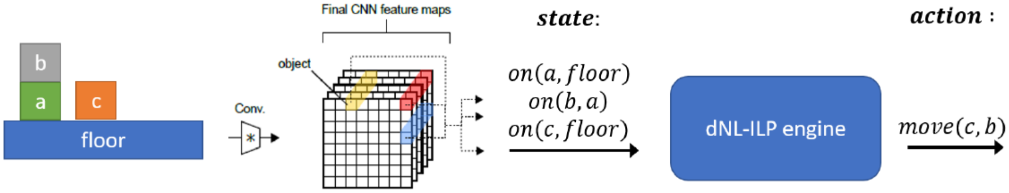

Figure 8. Design of the proposed RRL framework

a closet, grabbing your shoes and closing the door). These types of behaviours may not be able to be properly captured with an MDP reward function.

To address the first challenge, the authors suggest the use of some sort of vocabulary to perform reward specification. In this case, the vocabulary will be the propositional symbols from any formal language that can be transformed into a reward machine. Based on this idea Camacho et al. (2019a) define the reward specification as set:

$$R_s = \{(r_1 : \varphi_1), \ldots, (r_n : \varphi_n)\}$$

where $r_i \in R$ and φ_i is a formula over the chosen vocabulary.

To address the second challenge, the authors suggest the use of non-Markovian Reward Decision Process (NMRDP) which is basically an MDP for the exception of the reward which looks like:

$$R : (S \times A)^+ \times S \to R$$

As mentioned previously Camacho et al. (2019a) will be using Reward Machines [8]. Intuitively, they define RM as a framework used to indicate what reward function should currently be used to provide a reward signal given the sequence of state labels. The combination of formal languages and RM will help the agent with the problem of sample efficiency. To generate a RM one has to first construct a deterministic finite automata from a set of formal predicates and then construct the RM based on said DFA.

Bringing it all together, the authors provide proof that a reward machine and a reward specification can induce the same non-Markovian reward function and they also provide proof that such reward machine can be constructed and induce the same non-Markovian reward as the reward specification with formal language. The authors develop a new reward machine-tailored q-learning algorithm that significantly enhances the existing Q-Learning for RM (or QRM) algorithm through the exploitation of reward shaping.

To test the proposed framework Camacho et al. (2019a) test it on three different environments. First, the office work environment. In this grid world the agent can move in the four cardinal directions. At certain locations, the agent can find coffee, mail, an office, and decorations which

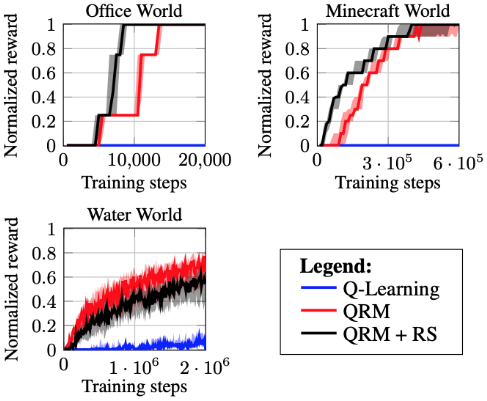

Figure 9. Normalized reward over iterations, Q-learning, QRM and QRM+RS

the agents interacts with depending on the task provided. The second environment is minecraft world, in this grid world domain, the agent can move in the four cardinal directions and can pick up different raw materials in order to build objects. The third environment is waterworld, where the agent moves around a continuous two-dimensional box, by changing its velocity in one of the four cardinal directions on every step. Different coloured balls are also moving around the environment and the tasks all correspond to touching different coloured balls at specific sequences.

In all three cases the QRM algorithm outperforms the traditional Q-learner (Fig. 9). The data illustrates the powerful advantage that QRM based approaches have over standard q-learning and it further shows that combining reward shaping with QRM (QRM+RS) leads to significant improvements in two of the three domains.

3.4. Toro Icarte et al. 2019b [10]

Even though the combination of neural networks for function approximation with Reinforcement Learning has become a staple in solving complex MDP problems, Deep RL methods struggle to find an optimal solution when the environment is partially observable. This issue is due to the agents in these environments requiring some form of memory to be able to learn optimal policies.

Toro Icarte et al. (2019b) proposes the use of Reward

Machines (RM) as a tool for providing memory in partially observable environments. RM have been originally conceived to provide a structured, automated representation of reward functions [5] [8]. The RM structure can be a tailored version of the q-learning algorithm called Q-Learning for Reward Machines (QRM) that is capable of learning separate policies for each state in the RM and it has been shown to outperform standard deep RL algorithms in several domains. In addition the authors propose a method that instead of defining an RM the system would learn the RM directly from experience in partially observable environments so it could be used as memory for an RL algorithm. It's only requirement is that the learning methods for the RM have to be provided with a finite set of detectors for properties that are used as the vocabulary for the RM. The goal for the learning of the RM is to allow it to represent the task as a discrete optimization problem and to find an efficient local search approach to solve it. The system will be learning the RM and the policy of the environment at the same time.

Toro Icarte et al. (2019b) tested their proposed approach (LRM) on three different POMDPs based on grid domains. First, the cookie domain. It has three rooms connected by a hallway. There is a button in the yellow room that, when pressed, causes a cookie to randomly appear in the red or blue room. The goal is for the agent to press the button and go reach the cookie. Second, the symbol domain has three symbols in the red and blue rooms. One of the symbols and possibly a right or left arrow are randomly placed in the yellow room. Intuitively, that symbol and arrow tell the agent where to go. Third is the 2-keys domain. The agent receives a reward of +1 when it reaches the coffee in the yellow room. To do so, it must open the two doors. Each door requires a different key to open it, and the agent can only carry one key at a time. Initially, the two keys are randomly located in either the blue room, the red room, or split between them.

The authors tested two versions of the suggested LRM method, LRM+DDQN which learns the policy using DDQN and LRM+DQRM which uses a modified version of the QRM. Based on the results of their experiments LRM approaches largely outperform the baselines algorithms, reaching close-to-optimal policies in the cookie and symbol domain, and the LRM+DQRM methods learns faster than LRM+DDQN methods but it is more volatile.

3.5. Illanes et al. 2020 [11]

Over the years Reinforcement Learning has become a great tool for solving complex continuous control problems in robotics. The advantage of using model-free RL is based on being able to learn policies that maximize an external reward signal through the interaction with the environment without the need of a preprogrammed model of the characteristics that control said environment. This advantage comes with perks. In order for the agent to learn the environment's dynamics and reward structure it can only be done through random exploration, rising a problem of sample efficiency. Also, RL is not meant to learn multiple tasks at the same time, an agent would have to learn a task first, then one would have to redefine a new reward structure for a second task and learn a new policy, this problem is known as transfer learning.

In order to solve for the sample efficiency limitations of Reinforcement Learning Illanes et al. propose leveraging high-level symbolic planning models and automated synthesis techniques, in combination with RL techniques. The authors base this solution on the observation that approximated understanding of the environment can be characterized as a symbolic planning model, while leaving possibly complex low level aspects of the environment unspecified. As a result the RL problem is taskable, given the user can program tasks as goal conditions in the symbolic domain. The RL agent can improve sample efficiency as the high-level plans can be used for transferring learning from previously learned policies, and the agent can learn complex low-level control policies as it relies on model-free RL to accommodate for all the information missing in the high-level model.

To develop this novel system, Illanes et al. base their work on the concept of learning through instructions [1], which shows that sample efficiency can be improved if a manually generated description of the task is given to the agent. Illanes et al. take it further. They propose to automatically generate useful instructions using a high-level abstraction of the environment and combine it with a hierarchical reinforcement learning (HRL) based algorithm to exploit said instructions.

To evaluate the proposed framework Illanes et al. compare their work with a standard HRL in two environments, OfficeWorld and Minecraft. After training and testing the system in these environments the results show that after the option policies are sufficiently trained the proposed system significantly outperforms the traditional Q-learning and HRL algorithms.

Literary Review - Performance

The papers discussed provide different perspective on how to combine Reinforcement Learning with different types of logic programming and formal logic languages to improve the agent's learning performance. The results of these papers are promising and demonstrate the potential of their methods, but in my opinion, there are some aspects of their research work that could be improved.

In the work presented by Toro Icarte et al. (2019a) [9] when working with the algorithm one has to provide a background knowledge function as a parameter. The accuracy of this function seems to have a big role on the success of this algorithm. This includes a level of complexity to the algorithm that is not discussed in the paper and can affect the scalability and generalization capabilities of this method. Also, the agent's obsession with the provided advice might result in slower convergence. There should exist some sort of trade-off between goal and advice. The authors do not dive into this topic at all and despite it being something important for the algorithm to be able to generalize or scale. Finally, the paper shows the potential usefulness for techniques that are based on advice (guidance) instead of constraints (pruning), but does not dive deeper into a comparison of both methods, specially with the high interest in safety of RL agents and its potential to generalize to these types of tasks.

Ali Payani and Faramarz Fekri [15] claim that their algorithm is capable of making the learned hypothesis interpretable, but the interpretation is actually limited to formal logic interpretation and therefore not interpretable to a non-expert. This claim should be better explained or further worked on for a better end-to-end interpretable solution. In recent years there have been other Deep RRL algorithms developed that are capable of processing images [17]. This Deep RRL algorithm would've been a better method to actually compare the viability of this algorithm given the authors claims.

Camacho et al. (2019a) [5] do not provide any discussion on the computational complexity of the QRM algorithm. The data provided is not clear on the efficiency of this algorithm and therefore one cannot determine if it may have application on more broader environments. The authors omit necessary detail in the evaluation section in regards to the tests. They claim to give the agents a set of tasks on each environment but do not provide any details in regards to what these tasks may be. This is important because we want to be able to properly determine how complex of a task can this algorithm solve, and if its comparable to other algorithms out there.

The work from Toro Icarte et al. (2019b) [10] and Illanes et al. 2020 [11] have the same limitation when it comes to scalability and their experiments. In terms of scalability, the results from the experiments do not provide any insight on its potential. How long does it take to solve the problems in these environments? Can this be scaled to more complex environments? These are questions that would be interesting to solve and to do so the authors should work on providing more detailed data. There is not a lot of transparency in regards to the environments used in the experiments. What is the size of the grid world? what is the size of the office world? How many steps does it take to traverse a room? etc. This type of information is crucial in order for us to appreciate and evaluate the actual performance difference of the methods provided.

4. Generalization of RL policy learning

The papers in this section are mainly focused on the use of Logic programming to assist Reinforcement Learning to improve the possibility of generalization for the RL algorithm.

4.1. Zhengyao Jiang and Shan Luo 2019 [12]

Even though Deep Reinforcement Learning (DRL) algorithms have been able to achieve incredible results in complex learning tasks, the learnt policies it produces are usually very hard to interpret and are not able to be generalized to different from the one it was meant to solve. Interpretability is a crucial feature of reinforcement learning and in general of all machine learning algorithms for system verification and improvement. The ability for an algorithm to generalize is important for reinforcement learning, usually in real world scenarios, it is not common that the training and test environments are exactly the same.

To address these limitations, Jiang et al. propose a new algorithm called Neural Logic Reinforcement Learning (NLRL) with the goal of representing the policies in reinforcement learning by first-order logic. NLRL is a policy gradient methods that uses differentiable inductive logic programming which has been proven to provide significant advantages for interpretability and generalization in supervised tasks.

Differentiable Inductive Logic Programming (or DILP) is a reimplementation of ILP into an end-to-end differentiable architecture. With the differentiable deduction, the system can be trained with gradient-based methods. The authors introduce a new DILP architecture called Differentiable Recurrent Logic Machine (DRLM). Compared to DILP, in DRLM there is more flexibility in the number of clauses that are used to define a predicate. It also needs less memory to construct a model and it enables for a longer learning logic chaining of different intentional predicates. These benefits make DRLM capable of working with larger problems.

Jiang et al. present a formulation of MDPs with logic interpretation. An MDP with logic interpretation is a triple (M, p_S, p_A), such that:

$$M : (S, A, T, R)$$

is a finite horizon MDP,

$$p_S : S \to 2^G$$

is the state encoder that maps each state to a set of atoms including both information of the current state and background knowledge, and

$$p_A : [0,1]^{|D|} \to [0,1]^{|A|}$$

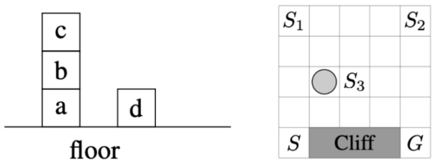

Figure 10. Block Manipulation and Cliff-Walking

is the action decoder that maps the valuation of a set of atoms D to the probability of actions.

Following this, the authors provide a brief overview on how to solve the presented MDP (with logic interpretation) with the combination of policy gradient and DILP. To evaluate the NLRL algorithm Jian et al. run the algorithm in two environments (Fig.). First, Block Manipulation, which consists of stacking blocks into a specific form. Second, Cliff-Walking, which consists of a 5 × 5 grid world where the agent has to reach a goal state without falling on the cliff.

In the Block Manipulation experiment the agent is tested on three tasks; STACK, UNSTACK and ON. To test the generalizability of the induced policy in the STACK and UNSTACK learning tasks they construct the test environment by modifying its initial state by swapping the top 2 blocks or dividing the blocks into 2 columns. The agent is also tested in the environments with more blocks stacking in one column. For the ON task they swap either the top two or middle two blocks, and also increase the total number of blocks. Similarly, to test the generalizability of the induced policy in the Cliff-Walking experiment the authors changed the initial position of the agent and the goal positions the agent has to reach. Also, they changed the size to the grid world to 6 × 6 and 7 × 7 and even tested a stochastic variant called Windy Cliff-Walking where the agent has a 10% chance to move downwards regardless of the chosen action.

The results show that the NLRL agent succeeds to find near-optimal policies on all the tasks.

Literary Review - Generalization

The paper discussed provides different perspective on how to combine Reinforcement Learning with different types of Logic programming and formal logic languages to improve the ability to generalize the RL solution. The results of these papers are promising and demonstrate the potential of their methods, but in my opinion, there are some aspects of their research work that could be improved.

The system introduced by Zhengyao Jiang and Shan Luo [12] integrates a high number of hyperparameters to tune manually. The performance is heavily influenced by these parameters. To ensure the scalability and the potential generalization of this approach it would be interesting to try and update the methods for it to alleviate their burden. The authors claim that their algorithm is capable of making the learned hypothesis interpretable, but it is actually limited to formal logic interpretation and therefore not interpretable to a non-expert. This claim should be better explained or further worked to provide better end-to-end interpretability. Finally, the novel system can be interpreted as being computationally expensive and it is not addressed as an issue even though it could affect its generalization capability when dealing with more complex scenarios.

5. Various topics

In this section I will be covering the papers that couldn't be classified in one of the three main topics mentioned above. As mentioned in the Introduction these papers were not examined in-depth.

Reinforcement Learning requires extensive exploration of the environment in order to find the optimal policy, and this is especially true when in comes to complex tasks. Transferring knowledge from a source task to a target task can be an effective way to expedite RL [13]. Zhe Xu and Ufuk Topcu [16] propose the transfer of knowledge between temporal tasks, which are tasks whose timing of the events matter. They develop a form of characterizing the similarity between temporal tasks through the use of logical transferability and develop a method to transfer learning between similar temporal tasks. So, the system performs RL on an extended state environment that includes the locations and clock valuations of the timed automata of the source task, establishing mappings between the components of said automata from the two tasks, and transfer the extended q-function based on these mappings. Then, it performs RL on the extended state for the target task. The evaluation of the proposed methods in two domains show its capability to improve the sampling efficiency of the RL problem.

Banihashemi et al. [3] develop a framework for agent abstraction based on situation calculus and the CoGolog programming language. They develop a refinement mapping specification on how each action of the high-level specification of the agent is implemented by CoGolog programming the low-level specification of the agent and how the fluents are translated. They introduce the concept of sound abstraction between the high-level and low-level action theories as suitable bi-simulation between their respective models, as well as complete abstraction where all actions that can happen in the high-level can also happen in the low level. This novel framework can be used to efficiently solve complex reasoning problems by finding a solution in the abstract model, and then use this abstract solution as a template to guide the search for a solution in the actual model.

When dealing with complex problems Reinforcement learning can struggle to produce a policy that is humanly interpretable. To address this issues Camacho et al. (2019b) [6] introduces the problem of learning a Linear Temporal

Logic formula that can capture set of positive and negative example traces. This approach to learning LTL uses symbolic state representation and searches through a space of labelled formulas to generate an alternating automaton that models observed behaviour from which the LTL can read off. Testing their system in different environments showcases its ability to learn human-interpretable behaviour models.

Brafman et al. [4] investigate non-Markovian Fully Observable Planning Domains (NMFOPD) and introduce a variation of the NMFOPD called TFOND where conditions on the history and specified using linear temporal logic on finite traces LTL$_f$ / LDL$_f$. They develop the necessary algorithms for planning in TFOND environments for goal expressed in LTL$_f$ / LDL$_f$ and determine complexity bound with respect to the domain and the goal. In addition, Brafman et al. show that TFOND are able to capture NMFONDs where the dependency on the history is "finite state", and can also capture Partially Observable Nondeterministic Planning Domains.

Toro Icarte et al. (2018) [8] introduce the concept Reward Machines (which is used in previously discussed papers). They define Reward Machines as a type of finite state machine that supports the possibility of specifying reward functions to the learner and allow for task decomposition. When paired with the introduced Q-Learning for Reward Machines algorithm (QRM, which can be incorporated with deep neural networks), it can appropriately decomposes the reward machine and uses off-policy q-learning to simultaneously learn sub-policies for the different components. The authors show that QRM will converge to an optimal policy. Toro Icarte et al. (2018) test their proposed framework in two discrete domains and show that find better policies more quickly than Hierarchical Reinforcement Learning algorithms in domains with a continuous state spaces.

6. Future Work

I have critically reviewed each paper but there are some major critiques and gaps that can be generalized to all the research literature I have read and that helped me define potential future work. Most of the research presented lacks any type of testing in complex environments. Most of the experiments are performed in 2D environments. None of them even try and include more complex components such as computer vision into their testing, and therefore the novel systems introduced can't be determined to be robust enough to hold in real world complex problems. All the algorithms work with the assumption of the agents having a perfect perception system. Human and computer vision are imperfect at detecting objects sometimes making the problem of safety guarantee much more complex. The introduction of this type of noisy data is neglected by all systems. This makes it impossible to determine if the novel systems are robust enough to properly perform in environments that may be more similar to real world scenarios. Given these critiques, the other gaps identified in current research, and the existing literature I was able to define two potential sources for the development of a new research project.

First, would be to choose one or more of the main topics (safety, performance or generalization) and develop a method that could be compared in more complex benchmarks. The current research has been eye-opening in the sense of its potential to improve the RL algorithm, but as mentioned above the tests run are limited to very basic bench marks. It would be interesting to compile and define a list of bench marks and sort them by complexity and see how far a algorithm that combines RL and Formal Logic can work in said milestones. Personally I would be interested in having a focus in safety of the RL agent.

A more ambitious and complex research project would consist of the use of RL and Formal Logic in complex environments, such as a 3D world that includes noisy data and computer vision. I would train an object detection system (with an existing algorithm) to extract the positions of safety related objects, use the pre-trained object detection system in RL training and apply LTL formula to improve the original RL and assist with safety of the agent. It would be interesting to determine if RL + Logic has the potential to be adapted to real world complex applications.

Other minor research topics that have come to mind based on the gaps in the current literature; a comparative analysis on the use of Linear Temporal Logic (LTL) and the use of LTL$_f$ / LDL$_f$ (LTL / LDL on Finite Traces) to determine which language is better fit for which problems, similarly, a comparative analysis on the use of Inductive Logic programming (ILP) and the use LTL to determine which language is better fit for which problems, or a deeper dive into the modularity of Reward Machines and Q-Learning for Reward machines.

References

[1] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pages 166–175, 2017.

[2] Jorge A Baier and Sheila A McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 788. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[3] Bita Banihashemi, Giuseppe De Giacomo, and Yves Lespérance. Abstraction in situation calculus action theories. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[4] Ronen I Brafman and Giuseppe De Giacomo. Planning for ltlf/ldlf goals in non-markovian fully observable nondeterministic domains. In *IJCAI*, pages 1602–1608, 2019.

[5] Alberto Camacho, Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pages 6065–6073, 2019a.

[6] Alberto Camacho and Sheila A McIlraith. Learning interpretable models expressed in linear temporal logic. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 621–630, 2019b.

[7] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Cautious reinforcement learning with logical constraints. *arXiv preprint arXiv:2002.12156*, 2020.

[8] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116, 2018.

[9] Rodrigo Toro Icarte, Toryn Q Klassen, Richard Anthony Valenzano, and Sheila A McIlraith. Advice-based exploration in model-based reinforcement learning. In *Canadian Conference on Artificial Intelligence*, pages 72–83. Springer, 2019a.

[10] Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. Learning reward machines for partially observable reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 15523–15534, 2019b.

[11] León Illanes, Xi Yan, Rodrigo Toro Icarte, and Sheila A McIlraith. Symbolic plans as high-level instructions for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 540–550, 2020.

[12] Zhengyao Jiang and Shan Luo. Neural logic reinforcement learning. *arXiv preprint arXiv:1904.10729*, 2019.

[13] Girish Joshi and Girish Chowdhary. Cross-domain transfer in reinforcement learning using target apprentice. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7525–7532. IEEE, 2018.

[14] Xiao Li and Calin Belta. Temporal logic guided safe reinforcement learning using control barrier functions. *arXiv preprint arXiv:1903.09885*, 2019.

[15] Ali Payani and Faramarz Fekri. Incorporating relational background knowledge into reinforcement learning via differentiable inductive logic programming. *arXiv preprint arXiv:2003.10386*, 2020.

[16] Zhe Xu and Ufuk Topcu. Transfer of temporal logic formulas in reinforcement learning. *arXiv preprint arXiv:1909.04256*, 2019.

[17] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.