SENIOR THESIS 2021-22

Mid-Semester Report

Samar Rahmouni
srahmoun@andrew.cmu.edu

Advisor: Prof. Giselle Reis
giselle@cmu.edu

1 Abstract

To redo, don't read. Reinforcement learning (RL), though a powerful and simple trial-and-error procedure that found a lot of success in games like Go, cannot be deployed in the real world because of the lack of security guarantees. For instance, though an autonomous car trained with reinforcement learning is bound to learn how to drive, the AI needs to crash to learn that crashing is not desirable.

In the context of the proposed thesis, we investigate both the security and the interpretability aspect of reinforcement learning in a cooperative adaptive cruise control inspired from [1], in the aim of finding how formal security frameworks can guide the representation, robustness and extrapolation of knowledge in reinforcement learning agents. We do so by experimenting and comparing both the optimization and security of three different RL implementations.

The first is a basic tabular Q-learning where we expect no security guarantees. The second is a hybrid architecture that incorporates the safe controller in [1] in a RL architecture. The safe controller (SC) computes a range of safe velocities given the current state of the environment in the car platooning scenario. Precisely, for multiple vehicles following each other, platooning aims to reduce the distance between them, hence taking less space on the road and allowing more vehicles to occupy highways, for instance. In the RL architecture, the SC then constricts the possible actions of the RL continuously at every given time step and the role of the RL is purely to find the optimal velocity in the safe range to minimize the distance between the cars. The third is what we will define as a logic-based inference RL. We investigate learning inferences rules in a deterministic environment where the result of an action given a state is not dependent on any probabilistic event. We approach the problem using inductive reasoning where the goal is to incorporate the learned rule knowledge into the decision making of a tabular Q-learning RL agent. In the given scenario, this will make use of the SC to develop a mapping from the state representation to a reward scheme, i.e. punish before a crash is bound to happen and adapt the Q-value and the epsilon-greedy approach.

2 Introduction

Implementing a robust adaptive controller that is effective in terms of precision, time, and quality of decision when facing dynamic and uncertain scenarios, has always been a central challenge in AI and robotics. As autonomous cars are deployed, IoT is popularized, and human-robot interactions become more complex, we are more and more confronted with the need for robotic agents that can effectively and continually adapt to their surroundings, not only in simulation, but also in practice, when deployed as a cyber-physical system. Since we are unable to provide a repertoire of all possible scenarios and actions, our agents need to be able to autonomously predict and adapt to new changes. RL is an approach that supports developing these capabilities, it is also the solution that AlphaGo, Deepmind AlphaStar, and OpenAI Five have adopted [2], respectively for Go, StarCraft II and Dota 2 and found success in.

However, as RL is a trial-and-error process, a car trained using RL is bound to crash to learn not to crash again. Safe reinforcement learning is then crucial to investigate in order to be able to deploy it in larger scales, but also out of simulation. The survey in [3] lays the foundations of verifications goals for Artificial Neural Networks (ANNs) to be ensured for safety-critical tasks. The need for this behavioral constraints stems from the inability to analyze and describe the behavior of NNs. The same argument translates to RL. In general, reinforcement learning does not ensure any of the verification conditions in [3]. The observable behavior of a RL agent is unpredictable (e.g. AlphaGo gamestyle) and will more often than not take hazardous actions since it is a trial-and-error process. Hence, it is important to investigate the use of verified safety controller and the interpretability of RL in order to achieve these verifications goals.

2.1 Reinforcement Learning as Trial-and-Error

Reinforcement Learning is a method of learning that maps situations to actions in order to maximize its rewards [4]. Compared to data-driven machine learning algorithms, reinforcement learning has the advantage of not requiring a prior dataset as the agent is not told what to do, but rather learns from the effect of its actions on the environment. Consider figure 1.

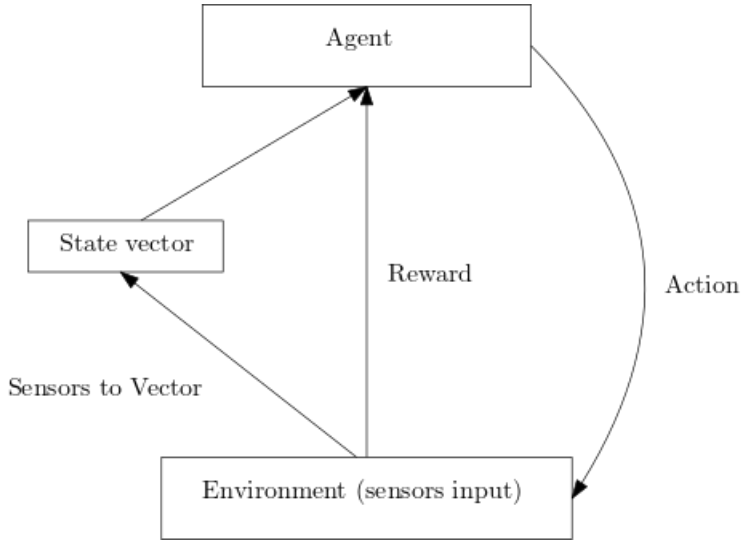

Figure 1: Reinforcement Learning Routine

The following RL routine shown in figure 1 describes the high-level of how an agent using reinforcement learning can be trained. Precisely, given the current external observation, which in practice are based on the sensors output are used to construct a state vector. This distinction between state and observation is crucial in a RL setting. The underlying assumption of any RL implementation is the **Markov Property**. In other words, the current state provides enough information to predict the future state and its associated reward. This needs not to be deterministic. It's the idea that a decision that was made n steps in the past should not affect the current step the agent is in. add some notes how this actually works a lot in practice. Thus, it's only through the state vector that the agent is trained. At every timestep, the agent takes in the state vector, and chooses an action. The environment then rewards it accordingly.

2.2 Engineering issues with Reinforcement Learning

In the routine shown in figure 1, we can identify two main engineering issues that are raised once RL is chosen for training. First, in the case of most scenarios, a reward function needs to be designed. There are reasonable criteria on how this should be designed. For instance, consider a reinforcement learning agent whose goal is to predict the weather. The closer the prediction, the more reward it should be given. Add some references about how best rewards are chosen.

Second, there is a need to choose from the observation what we assume is needed for the state to hold the Markov Property. This can be easy to do for certain scenarios. For instance, add an example here. However, it becomes more complex as tasks require more sensing. For instance, consider an autonomous vehicle. The observation could be the current fuel, velocity, position, proximity sensors and camera feedback. It is a problem similar to feature extraction in most Machine Learning (ML) settings. How it's usually being done in practice

The result of training is a policy, a function to map states to action, thus that maximizes the cumulative rewards. However, several problems come into place when considering the deployment of the trained agent. First, given the mapping from observation to state vector, certain uncertainties will not be expected, thus there can be no expectation on how the agent will behave. This results in an unsafe trained agent, thus constricting reinforcement learning to the realm of simulation.

Simulation in itself poses multiple issues. The first is again, the need to design and account for how the external real-world environment would behave. This can be hard to do accurately, and usually is the more exhausting part of the implementation process. A best case scenario is to be able to train any agent using RL in the real-world. This can be done if we are able to guarantee some safety properties. Those safety properties need not to be perfect, but should severly decrease the possibility of accidents and dangerous outcomes.

2.3 Safety in Reinforcement Learning

To be able to guarantee some safety properties, we look into a safe controller implementation that acts like an oracle, constricting the possible actions of the RL agent to those it deems "safe". The following work will focus on (1) the integration of this safe controller to the reinforcement learning training process, see section 3 and (2) the theoretical guarantees of the safe controller and its implementation, see section 4 and finally, (3) testing the hybrid architecture of RL+SC on a vehicle platooning problem done in simulation as to compare its safety and optimality with a basic RL implementation. Figure 2 is an overview of the proposed solution and the high level basis of the work.

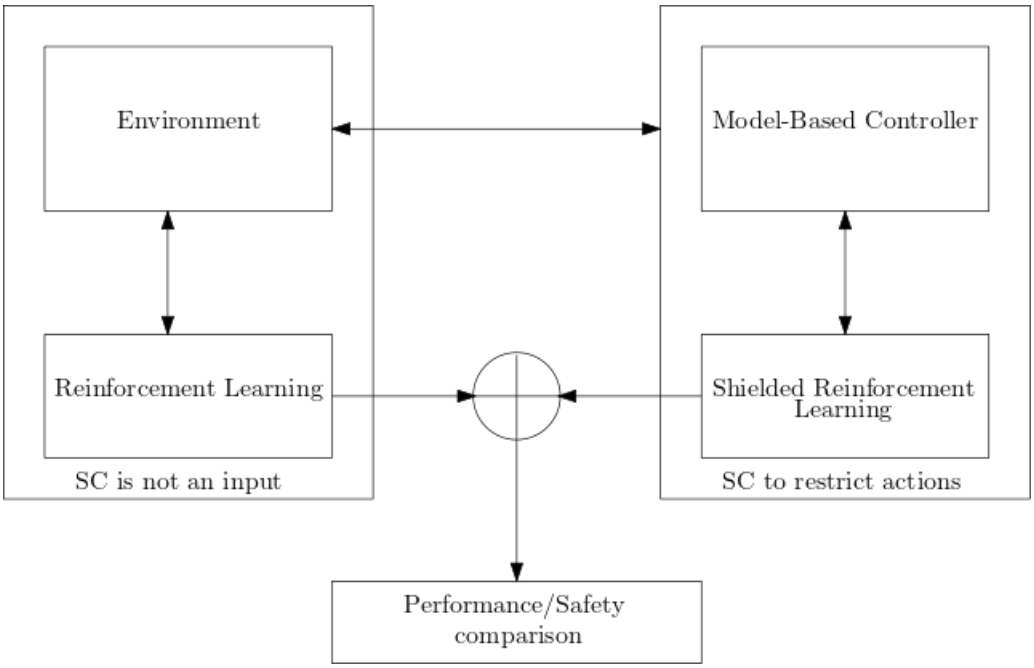

Figure 2: Overview of the proposed solution

Note that (3) is a work in progress. The current formalization of the scenario for the problem of vehicle platooning and its significance is in section 5.

3 RL+SC Architecture

Assume for the following that there exists a safe controller (SC) type of oracle that is able to take in the current environment as observed by the reinforcement learning agent and decide deterministically on a set of possible safe actions. More on the practical specifications of this safe controller can be found in section 4. We suppose that the SC can output false positives but never false negatives. In other words, if a safe controller deems a situation safe, then it has to be safe. We allow it to misjudge a situation and be conservative on safety.

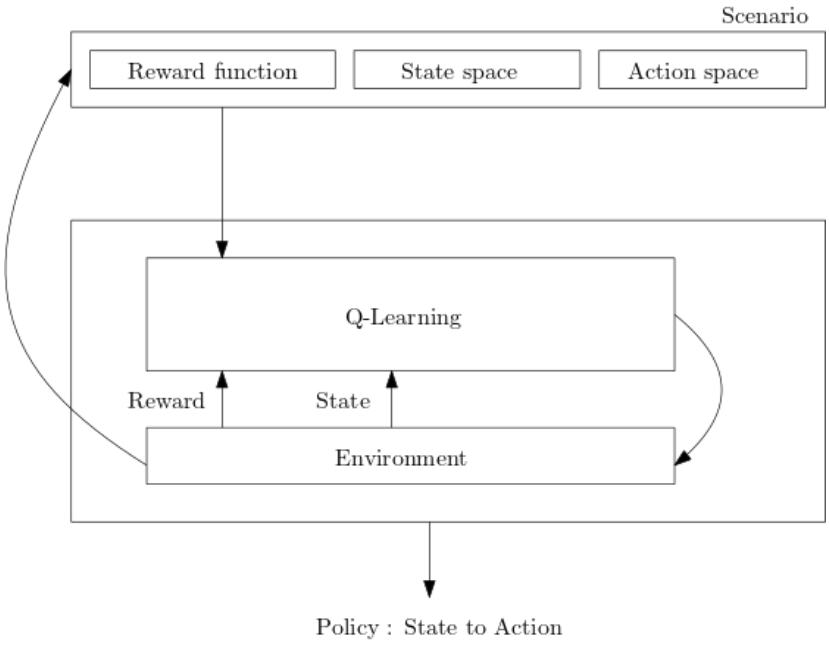

Figure 3: Basic reinforcement learning architecture

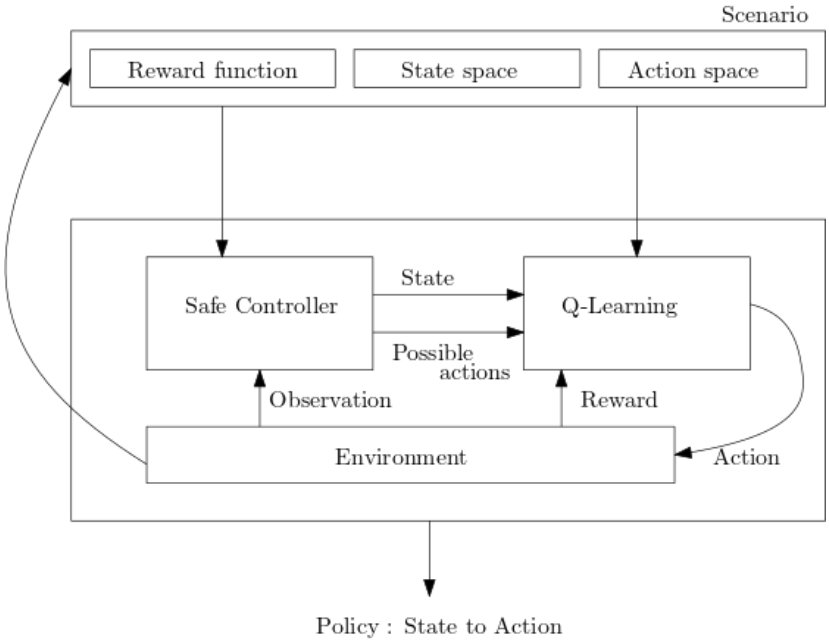

Figure 4: Safe controller integrated in reinforcement learning architecture

4 Safe Controller Specifications

In the previous section, we assume the existence of a safe controller that acts like an oracle. In our setting the safe controller takes in the current observation of the environment and returns the state representation and a set of possible actions to the reinforcement learning side. Some conditions need to hold on both the state representation and the set of possible actions to define a 'good' safe controller. In the following section, we lay down the foundations for such a module, starting by the guarantees we would like for it to have.

1. The returned state upholds the Markov Property.
2. The resulting state of any action of the possible set of actions should not include any known fatality conditions, except if necessary.

We make the following assumptions, (1) actions are deterministic, meaning if an agent chooses to go left, they will go left with certainty, (2) sensors can be faulty and they obey a gaussian distribution for their uncertainty, (3) the world is filled with uncertainties and the safe controller cannot account for all of them. Note that in the following, we will define uncertainty as an evolution of the world given the sensors that is unknown by the safe controller, i.e. no known step semantics for the uncertain.

Given our assumptions, we make those first observations.
Observation 1: Though our SC cannot account for all uncertainties, there exists a set of expected information that is informed given known step semantics, i.e. E_t. Precisely, considering actions are deterministic and the problem is a Markov Decision Problem (MDP), the SC can compute the resulting expected state.
Observation 2: The SC cannot ensure that the expected will be the next observed.
Observation 3: Since there is a finite number of sensors, every sensor output can be defined as a proposition associated with a specific value.

By taking our observation 1 and our assumption 2, we can construct the following deterministic approach to compute the state and the set of possible actions.

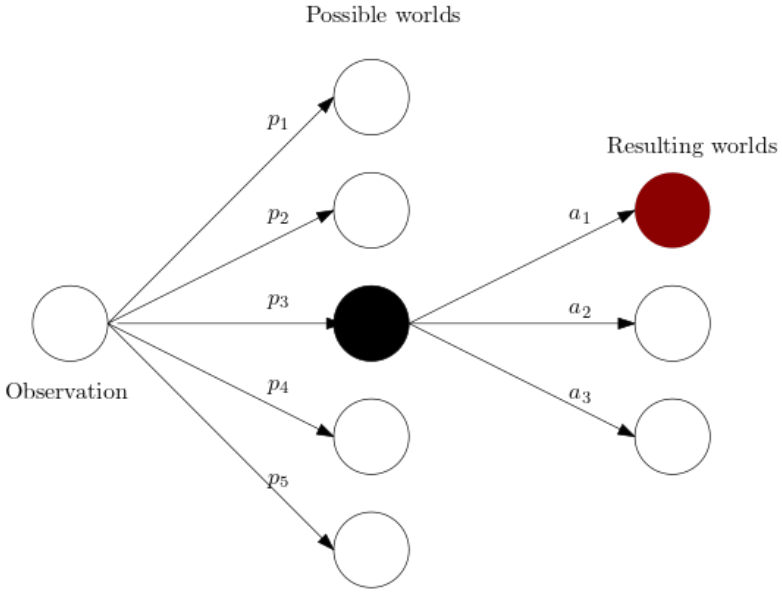

Figure 5: SC to compute the state and possible actions

In the first layer, if we know the distribution of our sensors, we are able to compute the possible worlds and assume the one with the highest probability. In the case of figure 5, it is p_5. By assuming our state and the possible actions, we can utilize domain specific knowledge into computing the expected possible

worlds. Given some fatality conditions, it is possible to assess whether a given world is desirable or should be blocked.
There's a lot more than can go on here. A couple of my notes.

- Consider history, basically : $[S_t, a, S_{t+1}/E_t]$. We are technically able to compute the level of uncertainty of a result, since we're only dealing with numerical computation. Basically, suppose I never had $sound(e)$ in my observation, I don't know what it is, for all I know, it could be the worse possible outcome and I have no clue how to react to it. I should be able then to choose my safest action, the one I know to always result in non fatal conditions. Suppose I survive such an uncertainty, then now I know what the next state was actually compared to what I expected, if I 'normalize' the next state while accounting for uncertainty of sensors and compare it to my expected, I now have a numerical value that I can embed in my step semantics to adapt for this uncertainty. Inductive reasoning basically.
- Obviously, the more domain specific knowledge, the less uncertainties to deal with.
- Could also just give the uncertainty decision to the RL, basically randomly try it. It's a sacrifice we're willing to make for learning. We're still ensuring some aspect of safety so already doing better.
- For a better argument on how the deployment in real life can be fine is a distinction between undesirable vs fatal. I don't really want to swerve into the road but I also won't exist if I crash into a car.

4.1 From SC to verification conditions

todo.

5 Vehicle Platooning : Case Study

To put our approach in practice, we consider the case of vehicle platooning. The setting is as follows: multiple autonomous vehicles are tasked with following each other while ensuring that they minimize the gap between them. As described in [5], forming a vehicle platoon helps reduce fuel consumption severly. The closer the proximity and faster the speed, the more fuel is saved. Even by assuming no faulty communication between the vehicles, the uncertainties of the road make it a challenge to deploy autonomous vehicles trained using reinforcement learning with that goal in mind. For instance, consider an unexpected construction on the usual route that the vehicles take. As the leader of the platoon will be the first to notice, we would like to communicate as early as possible to the followers that a decelaration needs to happen. In the context of reinforcement learning, there are two cases, (1) the agents have been trained on that occurence, thus they all decrease their velocities, or (2) the agents have never encountered that problem in training, thus are bound to fail.

5.1 Preliminary Results

todo

5.2 Discussion

todo

6 Related Work

One of the many ways safety has been approached is by formalization and symbolic reasoning. In the case of artificial intelligence, recent work proposes neurosymbolic integration. Neurosymbolic integration has been an ongoing work in the last years towards a combination of deep learning (DL) and symbolic reasoning. The work has been a response to criticism of DL, precisely, the lack of formal semantics and intuitive explanation and the lack of expert knowledge towards guiding machine learning models. Key questions the field targets are identifying the necessary and sufficient building blocks of AI [6], namely, how can we provide the semantics of knowledge, and work towards meta-learning? Meta-learning in reinforcement learning is the problem of learning-to-learn, which is about efficiently adapting a learned policy to conditions and tasks that were not encountered in the past. In RL, meta-learning involves adapting the learning parameters, balancing exploration and exploitation to direct the agent interaction [7, 8]. Meta-learning is a central problem in AI, since an agent that can solve more and more problems it has not seen before, approaches the ideal of a general-purpose AI.

Approaches that were concerned with safe reinforcement learning, have approached the problem in previous work [9] in two ways. First was changing the optimization criteria [10], precisely by incorporating risk into the performance of the policy. Namely, either considering the worst case scenario and constraining on it, reducing the variance to be more sensitive or applying constraints i.e. only update the policy if the action is in a safe set. This does not however guarantee safety, but tends to minimize the probability of risk, hence not allowing RL systems to be deployed in the physical world. Second was to consider the exploration process, either by incorporating external knowledge i.e. learning from demonstration [11] or adopting a risk directed exploration [12]. These approaches can be considered rigid, as requiring more data and expert knowledge that needs to be proven safe and sound, or in the latter, decreasing the efficiency and requiring more time for the learning process. In particular, in these previous two approaches, work that makes use of formal security frameworks is yet to be investigated.

Some of the more recent work that does investigate a hybrid architecture (i.e. RL and symbolic reasoning) makes use of set-theoretic techniques and constraint satisfaction problems to optimize from the constraints [13], or proposes a reactive system called a shield [14] to either constraint the actions given by the environment or adapt them once the RL module chooses one. In both cases, the added safety controller is described by its specifications, rather than being an independent existing controller.

Partially observable Markov decision process belief system to improve from observation to state vector.

Explain how this idea is novel.

7 Conclusions

todo.

References

[1] Yuri Gil Dantas, Vivek Nigam, and Carolyn Talcott. A formal security assessment framework for cooperative adaptive cruise control. In *2020 IEEE Vehicular Networking Conference (VNC)*, pages 1–8, 2020.

[2] Yuxi Li. Reinforcement Learning Applications. Technical Report arXiv:1908.06973, August 2019.

[3] Zeshan Kurd and Tim Kelly. Establishing safety criteria for artificial neural networks. volume 2773, pages 163–169, 09 2003.

[4] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[5] Erik Larsson, Gustav Sennton, and Jeffrey Larson. The vehicle platooning problem: Computational complexity and heuristics. *Transportation research. Part C, Emerging technologies*, 60(C):258–277, 2015.

[6] Artur d'Avila Garcez and Luis C. Lamb. Neurosymbolic ai: The 3rd wave, 2020.

[7] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-Reinforcement Learning of Structured Exploration Strategies. In *Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, page 10, 2018.

[8] Nicolas Schweighofer and Kenji Doya. Meta-learning in Reinforcement Learning. *Neural Networks*, 16(1):5–9, January 2003.

[9] Javier García and F. Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

[10] R Rockafellar and Stan Uryasev. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 01 2000.

[11] Nils T. Siebel and G. Sommer. Evolutionary reinforcement learning of artificial neural networks. *Int. J. Hybrid Intell. Syst.*, 4:171–183, 2007.

[12] Edith Law. Risk-directed exploration in reinforcement learning. 01 2005.

[13] Yutong Li, N. Li, H. E. Tseng, A. Girard, Dimitar Filev, and I. Kolmanovsky. Safe reinforcement learning using robust action governor. In *L4DC*, 2021.

[14] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. 08 2017.