Towards Verification and Validation of Reinforcement Learning in Safety-Critical Systems A Position Paper from the Aerospace Industry

Erik Nikko[1,2], **Zoran Sjanic**[1,2] and **Fredrik Heintz**[1]
[1]Linköping University
[2]Saab AB

Abstract

Reinforcement learning techniques have successfully been applied to solve challenging problems. Among the more famous examples are playing games such as Go and real-time computer games such as StarCraft II. In addition, reinforcement learning has successfully been deployed in cyber-physical systems such as robots playing a curling-based game. These are all important and significant achievements indicating that the techniques can be of value for the aerospace industry. However, to use these techniques in the aerospace industry, very high requirements on verification and validation must be met. In this position paper, we outline four key problems for verification and validation of reinforcement learning techniques. Solving these are an important step towards enabling reinforcement learning techniques to be used in safety critical domains such as the aerospace industry.

1 Introduction

Sequential decision-making is the area of making decisions that modify the environment both directly and in the long term, which in turns affect the decisions made later [Sutton and Barto, 2018]. Reinforcement learning (RL) is a type of technique that can solve these problems and that has seen tremendous progress in recent years as shown by impressive successes in playing Go [Silver *et al.*, 2016] and complex real-time computer games such as StarCraft II [Vinyals *et al.*, 2019]. In the cyber-physical setting, one finds examples such as robots playing a curling-based game [Won *et al.*, 2020].

While these results are important, as they provide technologies for solving increasingly more complex problems, there are still many issues left to solve. Among these are how to ensure that a set of properties (e.g. safety properties) hold. For example, unmanned aerial vehicles must avoid collisions to prevent hurting humans and damaging the environment. Another example is that a computer-controlled player in a game should behave fairly towards the other players and not single out a human player to make the game more enjoyable. Proving such properties is the area of formal verification [Mitsch and Platzer, 2016].

Proving properties is important to the aerospace industry where the requirements that the aircraft works correctly and safely are very high. To ensure that the system meets these requirements, all software is implemented and verified according to standards (for example, RTCA/DO-178C [RTCA, 2011]) as specified by the governing certification authority. To utilise any reinforcement learning technique in avionic software, it must be possible to validate that the software works as intended and verify that it works correctly to the same degree as specified in these standards. Unfortunately, existing methods are not always directly applicable due to the data-driven nature of RL techniques. Instead, new techniques for verification and validation are required before the RL systems can be used in aircraft.

One key issue when considering highly complex environments, such as autonomous aircraft in public airspace, is that all aspects of the environment cannot be considered at design time. The inability to do this is incompatible with the current regulations for aircraft. Hence, the problems presented in this paper concerns safety for autonomous aircraft (or similar settings) and future regulations that have the same high requirements on verification and validation as the current.

The idea of making a RL system fulfil a set of (safety) properties is not new. The field of safe reinforcement learning (safe RL) studies the safety (and risk) of a RL system [Garcıa and Fernández, 2015]. We aim to widen the question from safety to verification of general properties of RL techniques. Furthermore, we also stress the importance of showing that the system does the correct thing in the environment it is executed in.

In this paper, we outline four key problems related to verification and validation of RL systems that are worthy to investigate to build confidence to use them in safety critical domains. In addition, some recent work from safe RL is presented in relation to the questions. However, it should be stressed that these are indicative examples of how the problems are addressed in the field and not a formal survey of potential solutions.

It is worth mentioning that there are more problems than verification and validation that must be solved for RL to be used in safety critical systems. For example, defining and proving the correctness of the reward function and gathering representative data.

The structure of the paper is as follows. Section 2 defines

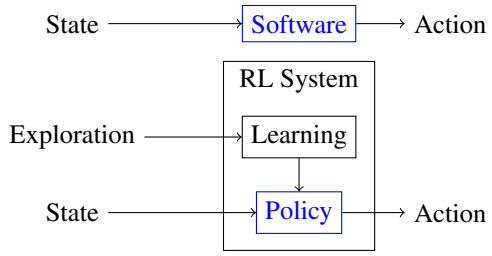

Figure 1: A simplified view of the targets for verification and validation for an agent implemented as traditional software and when implemented using RL techniques. The blue nodes are the parts that need to be verified and validated.

four problems related to verification and validation of RL systems. Thereafter, Section 3 briefly covers related techniques from the area of safe RL. Next is Section 4 with an example from the aerospace setting that aims to illustrate why it is important to solve the proposed problems. The last section (5) concludes the paper.

2 Verification and Validation

In this section, the problem of verification and validation of RL systems is introduced. This includes, the target of the verification and validation, the settings for which the target is verified and validated, and the formal definitions.

One important question when it comes to verification and validation for RL systems is what the target for the verification and validation is. In the case of traditional software, the target of the verification and validation is the software itself (e.g. verifying that a program produces the correct output given some input). However, RL system can conceptually be viewed as acting in two different parts. One part is a policy that determines what actions to take (output) given a state of the environment the system is in (input). The second part is the learning of the policy. Because the policy is the part responsible for making the decisions at any point in time, this is the part that is critical to verify and validate. Therefore, verification and validation of a RL system can be reduced to verify and validate the policies that the system produces. Figure 1 illustrates the difference between verification and validation of traditional software compared to RL systems in a simplified manner. While verifying and validating the policy is still a complex problem, it is more manageable than verifying and validating the whole RL system.

To further reduce the problem of verification and validation, one can limit the environments (an arbitrary environment is denoted Σ) for which the system is verified and validated. First, there is the target environment (denoted Σ_t) to which the agent is deployed to perform its task. However, this environment is not necessarily available when the agent learns a policy. Instead, learning might have to be done in one or more separate training environments (an arbitrary training environment is denoted Σ_*). The reason to why the target environment is not available could, for example, be economical, legal or ethical. When learning cannot be done in the target environment, the goal is that the policy learnt in the training environment generalises to the target environment. The target environment and the training environment(s) are the only environments that the agent will act in. Therefore, it is enough to verify and validate for these environments.

There is one fact that should be mentioned regarding the policies for exploration (agent tries different decisions to determine their usefulness and how the environment change) and exploitation (using the learnt knowledge to maximise the performance). Namely, these policies can be different where the former aims to explore as much as possible of the environment and the latter aims to perform as good as possible. Therefore, when doing verification and validation during exploration, it is enough to do so with regard to the policy controlling the agent at that time (the behaviour policy, denoted π_b). Similarly, when doing verification and validation during exploitation, it is enough to use the corresponding policy for that (the target policy, denoted π_t). It should be noted that the target policy may be the same as the behaviour policy.

With the above, there is enough information to describe the issue of verification of a RL system. Starting with the goal of verification: To guarantee that a set of properties (denoted ψ) hold when executing the system. To do this, external information (i.e. information about the environment that is not gained from exploration) regarding safety must be provided [Garcıa and Fernández, 2015]. The external information may differ in its form but for this paper it is considered to be a model of the environment (denoted M_v). Based on formal verification of cyber-physical system [Mitsch and Platzer, 2016], verification can now be defined for exploration and for exploitation as follows[1]:

Definition 1 (Verification during Exploration)**.** *Prove that the reinforcement learning system fulfils properties ψ according to model M_v when exploring environment Σ using policy π_b.*

Definition 2 (Verification during Exploitation)**.** *Prove that the reinforcement learning system fulfils properties ψ according to model M_v when exploiting policy π_t in environment Σ.*

By completing these verification tasks, it is shown that properties ψ hold during exploration and exploitation under the assumption that model M_v describes the environments within sufficient accuracy. Note that there will always be inaccuracies when acting in the physical world. For example, noisy sensors can create inaccuracies. Hence, any model that models the training or target environment can at most model it with some level of accuracy. The higher the accuracy, the stronger the claims based on the model are. Naturally, what is considered acceptable is domain dependent. In some domains, such as the aerospace domain, there are strict requirements on how certain it must be that nothing goes wrong. The accuracy of a model with regard to an environment (denoted Δ_e^m where m is the subscript of the model and e is the environment) is an important component when proving that the system meets the required safety.

The dependency on model M_v, leads to the next problem: Ensuring that the model used for verification models the en-

[1]The definitions for verification during exploration and exploitation could be merged into a single definition by using a generic policy π. However, work within safe RL does not always consider both exploration and exploitation. Hence, a separation between the two issues is done to make it easier to discuss these differences.

vironment the agent currently act in within the required accuracy when verification is being done. Note that this means that the validation of the verification model should be done for all environments where verification is done. This is formally defined as (based on an informal definition of run-time model validation [Mitsch and Platzer, 2016]):

Definition 3 (Validation of Verification Model)**.** *Prove that model M_v models environment Σ with an accuracy Δ^v_Σ.*

It should be acknowledged that validation of models is not a new question and for many types of models there exist good solutions. However, the environments such as public airspace or public roads that cyber-physical systems may act in contains uncontrolled factors and are of high dimensions and complexity. Partly because they contain many human actors and are therefore prone to human errors. To the authors' knowledge, validating models of these high complexity environments is not a solved issue. However, it should be noted that no thorough investigation of the subject has been conducted. Disregarding the complexity of the model and potential solutions, this problem must be solved to ensure the validity of the verification (Definitions 1 and 2).

There is one final problem that must be considered for verification and validation of RL systems: Validating that the system does the correct thing in the target environment. Under the assumption that the reward function is correctly defined (i.e. captures what is desired), this can be formulated as validating the performance (as measured by the total reward). An agent that has been trained to perform well in one or more training environments does not inherently guarantee to perform well in a target environment. This is because the policy learnt from the training environment may not generalise to the target environment. A generic informal definition of this problem could be *validate that the reinforcement learning system acts as expected in the target environment when acting according to the policy learnt during exploration*. Similar as in Definition 3, it is unreasonable to expect the exact same performance and therefore an error margin δ should be allowed. The following is a first attempt at formally defining the problem and should not be seen as the final definition of the problem. Instead, the goal is to promote a discussion regarding the problem, formal definitions and solutions.

Definition 4 (Validation of Performance)**.** *Prove that the total reward when exploiting policy π_t in target environment Σ_t when starting in state s is within an error margin of δ of the total reward when exploiting π_t in training environment Σ_* when starting in s.*

The four problems that are presented in this section are likely not possible to completely solve in the most general case. For example, verification during exploration (or exploitation) may be infeasible because there are too many situations that must be considered in the complex environments that are of interest. However, the number of situations can be reduced to a feasible number by only considering those that are relevant for a specific execution. Therefore, it is of interest to focus on the run-time version of these problems when it is not possible or feasible to solve for all possible situations. That is, prove that the verification problems are solved from the current state to some horizon into the future and that the validation problems are solved during the execution of the system. In fact, the run-time versions of the problems are of even more interest because the high dimensionality and complexity of cyber-physical systems and the environments they act in makes it unlikely that it is feasible to solve for all situations.

When working with the run-time versions of the problems, it becomes important to consider what to do when something is violated (during verification or validation). This is a very important part for the correctness of the system. However, it is also domain dependent, and it is therefore not presented as one of the generic problems.

3 Safe Reinforcement Learning

This section covers work that has been done within the field of safe RL that relates to the problem definitions in the previous section (or run-time version of them). It should be noted that this is neither an exhaustive list of all work that has been done nor a formal literature study of the field. For a survey of the field up until 2015 we refer to [Garcıa and Fernández, 2015].

One approach to ensure that a reinforcement learning agent fulfils a set of properties during exploration and exploitation (Definitions 1 and 2) is to restrict the actions that an agent may do in a state. This has been achieved by preventing the agent from selecting an unsafe action [Fulton and Platzer, 2018; Bouton *et al.*, 2019; Junges *et al.*, 2016; Jansen *et al.*, 2018] and by correcting selected unsafe actions to safe actions [Cheng *et al.*, 2019; Phan *et al.*, 2020; Alshiekh *et al.*, 2018]. Among the formalism used to achieve this, one finds different types of logics and Markov Decision Processes. Disregarding the formulation, they all have the common denominator that they have a predictive model of the environment that captures all the factors relevant to prove that the properties hold. It should be noted that the systems either requires a discrete action space or makes non-trivial assumptions about having a backup controller [Phan *et al.*, 2020] or a non-trivial function describing safety of states [Cheng *et al.*, 2019]. Furthermore, the systems have been evaluated on relatively low dimensional environments (e.g. inverted pendulum) and with potentially few uncontrolled but known actors (e.g. adaptive cruise control).

An alternative to constraining the actions in a state is to enforce a property over a trajectory (i.e. a sequence of states and actions) [Wen and Topcu, 2018; Chow *et al.*, 2018; Achiam *et al.*, 2017]. However, it should be noted that these techniques only guarantees that the properties are not violated during exploration under certain assumptions. For example, it can be required that there exists an initial policy that is safe [Chow *et al.*, 2018; Achiam *et al.*, 2017] or that one can simulate the safety of a trajectory [Wen and Topcu, 2018]. If the assumption required for the technique does not hold, then the exploration will remain unverified. Hence, these address Definition 2 and conditionally address Definition 1. The techniques handle continuous action space but have only been evaluated in either controlled domains with complex agents (e.g. controlling human agent in static world) or in simple domains with uncontrolled factors (e.g. navigation in grid-world with noise on movement).

A final approach that has been proposed to ensure that a RL system is safe is to learn safety. This is an interesting approach that requires that the agent may act freely and unsafely during exploration (note that the exploration can be in a training environment from which learned safety may be transferred to the target environment) [Srinivasan *et al.*, 2020; Bharadhwaj *et al.*, 2020]. This means that it addresses the issue of verification during exploitation. However, it should be noted that extracting guarantees from a learnt safety function is a complex problem in itself. A variation of learning a safety function is to start with an action constraint approach and dynamically adapt the model of the environment that is used if it does not match the target environment [Pranger *et al.*, 2020]. This approach addresses the issue of verification during exploration and exploitation (Definitions 1 and 2), inherited from the action constraint approach. However, the validation of the verification model (Definition 3) may be trickier because the model is not static during execution.

When it comes to validation of the verification model (Definition 3), one of the action constraint approaches based on differential dynamic logic continuously validates the model when executing the system [Fulton and Platzer, 2018; Fulton and Platzer, 2019; Mitsch and Platzer, 2016]. In the work, there are also various options for using multiple models, dependent variables and bounds for error margins to allow for noise.

4 An Aerospace Example

To illustrate the importance of solving the problems in Section 2, consider the case where a commercial passenger aircraft is piloted by a RL agent. This agent is required to handle many tasks, including but not limited to take-off, flight between two destinations and landing. Furthermore, there are many distinctly different situations (different qualities on runways, human error affecting the environment in unforeseen ways, emergency landings, etc.) and environmental conditions (extreme wind, rain, hail, ice, etc.) that the tasks must be completed safely under. As a result, the target environment is a complex environment with high dimensionality and have many uncontrolled factors.

The first thing that should be noted is that it is not possible to let the agent train in the target environment. One of the most obvious reasons is that it would entail a high risk of harming humans and the environment. There are also economical (flying and potentially crashing an empty aircraft is not cheap) and legal aspects that makes this unacceptable. Therefore, the training must be done in training environments, that potentially gradually approaches the target environment (and thus relate to curriculum learning). It should be noted that at least some training environments will include situations where an inappropriate action is dangerous and/or costly (for example, when flying). This is because the agent must encounter some of these situations before being deployed to learn what to do in them (thus explicitly facing dangers to learn to avoid them). For example, taxing to the runway at a test facility still has the risk of driving off the road or crash into other objects. Therefore, it is important that verification during exploration is done to ensure safety (Definition 1). This in turn requires that the model used by the verification must be validated (Definition 3).

After the agent has been trained safely, the next step is to deploy it in the target environment (most likely in test case runs without passenger first and later in live scenarios). However, the target environment has not been explored completely during training due to its complexity (for example, huge amount of different weather conditions, internal failures, communication loss, other flying vehicles and wear on the aircraft). Disregarding, the agent must be able to handle the new situations safely and with good performance. Therefore, it is necessary to validate that it performs as intended in the target environment (Definition 4). Furthermore, it is required to verify in the target environment (Definition 2) to ensure safety in any new situation. For example, the behaviour of other vehicles (often human operated) when taxing may differ from the behaviour of the vehicles in the test environment (potentially remote controlled to reduce risk of harming humans).

The final part in this example is the backup-procedures that applies when a violation is found in the verification or validation. In this example, the emergency procedures could range from aborting take-off to different levels of emergency landing. However, it is likely that all procedures are required to be as predictable and standardised as possible (same as the proposed emergency procedures for remotely piloted aircraft from ICAO [ICAO, 2017]).

5 Concluding Words

In this paper, four problems for verifying and validating RL systems for safety critical domains have been defined. Furthermore, some work within safe RL that either tangents with or aims to solve some of the presented problems have been summarised. Finally, an example from the aerospace industry has been presented to illustrate the importance to solve the four problems for safety critical domains.

The work that has been done in safe RL is an important step towards enabling using RL systems in safety critical domains. However, there is still much to be done before a RL system can be claimed to be verified and validated to the extent that it can be used in the safety critical parts of an aircraft. To our best knowledge, the work that has been done regarding verification either considers environments that are not of the desired complexity, does not handle continuous action space or does not consider verification during exploration. Furthermore, validation of the models used for the verification is considered by only one of the presented techniques used for verification, indicating that there is still much to be done. However, it should be stressed that other fields than safe RL may provide partial solutions to several of the presented problems. Future work is to refine the problem formulations, perform more extensive surveys of related work in other disciplines and to develop techniques for solving the presented problems.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous System and Software Program (WASP) founded by the Knut and Alice Wallenberg Foundation.

References

[Achiam *et al.*, 2017] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 22–31, 2017.

[Alshiekh *et al.*, 2018] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI Conference on Artificial Intelligence*, pages 2669–2678, 2018.

[Bharadhwaj *et al.*, 2020] Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

[Bouton *et al.*, 2019] Maxime Bouton, Jesper Karlsson, Alireza Nakhaei, Kikuo Fujimura, Mykel J Kochenderfer, and Jana Tumova. Reinforcement learning with probabilistic guarantees for autonomous driving. *arXiv preprint arXiv:1904.07189*, 2019.

[Cheng *et al.*, 2019] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[Chow *et al.*, 2018] Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8092–8101, 2018.

[Fulton and Platzer, 2018] Nathan Fulton and André Platzer. Safe reinforcement learning via formal methods. In *AAAI Conference on Artificial Intelligence*, 2018.

[Fulton and Platzer, 2019] Nathan Fulton and André Platzer. Verifiably safe off-model reinforcement learning. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 413–430. Springer, 2019.

[Garcıa and Fernández, 2015] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[ICAO, 2017] ICAO. Remotely piloted aircarft systems (rps) concept of operations for international ifr operations. Technical report, ICAO, 2017.

[Jansen *et al.*, 2018] Nils Jansen, Bettina Könighofer, Sebastian Junges, and Roderick Bloem. Shielded decision-making in mdps. *arXiv preprint arXiv:1807.06096*, 2018.

[Junges *et al.*, 2016] Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. Safety-constrained reinforcement learning for mdps. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 130–146. Springer, 2016.

[Mitsch and Platzer, 2016] Stefan Mitsch and André Platzer. Modelplex: Verified runtime validation of verified cyber-physical system models. *Formal Methods in System Design*, 49(1-2):33–74, 2016.

[Phan *et al.*, 2020] Dung T Phan, Radu Grosu, Nils Jansen, Nicola Paoletti, Scott A Smolka, and Scott D Stoller. Neural simplex architecture. In *NASA Formal Methods Symposium*, pages 97–114. Springer, 2020.

[Pranger *et al.*, 2020] Stefan Pranger, Bettina Könighofer, Martin Tappler, Martin Deixelberger, Nils Jansen, and Roderick Bloem. Adaptive shielding under uncertainty. *arXiv preprint arXiv:2010.03842*, 2020.

[RTCA, 2011] RTCA. *RTCA/DO-178C. Software considerations in airborne systems and equipment certification*. RTCA, Incorporated, 2011.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Srinivasan *et al.*, 2020] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

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

[Vinyals *et al.*, 2019] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[Wen and Topcu, 2018] Min Wen and Ufuk Topcu. Constrained cross-entropy method for safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 7450–7460, 2018.

[Won *et al.*, 2020] Dong-Ok Won, Klaus-Robert Müller, and Seong-Whan Lee. An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions. *Science Robotics*, 5(46), 2020.