2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion

Combining Machine-Learning with Invariants Assurance Techniques for Autonomous Systems

Extended Abstract
Piergiuseppe Mallozzi
Advised by Patrizio Pelliccione
Chalmers University of Technology | University of Gothenburg, Gothenburg, Sweden
mallozzi@chalmers.se

***Abstract*—Autonomous Systems are systems situated in some environment and are able of taking decision autonomously. The environment is not precisely known at design-time and it might be full of unforeseeable events that the autonomous system has to deal with at run-time. This brings two main problems to be addressed. One is that the uncertainty of the environment makes it difficult to model all the behaviours that the autonomous system might have at the design-time. A second problem is that, especially for safety-critical systems, maintaining the safety requirements is fundamental despite the systems adaptations. We address such problems by shifting some of the assurance tasks at run-time.**

We propose a method for delegating part of the decision making to agent-based algorithms using machine learning techniques. We then monitor at run-time that the decisions do not violate the autonomous system's safety-critical requirements and by doing so we also send feedback to the decision-making process so that it can *learn*. We plan to implement this approach using reinforcement learning for decision making and predictive monitoring for checking at run-time the preservation and/or violation of invariant properties of the system. We also plan to validate it using ROS as software middleware and miniaturized vehicles and real vehicles as hardware.

***Keywords*-autonomous systems; invariants enforcement; machine-learning;**

I. Introduction

Autonomous Systems must be able to adapt to any environment. We will be seeing them in many application domains including safety-critical ones such as automotive. An autonomous vehicle can, in fact, adapt to different situations using information coming both from the vehicle itself (its internal state) and from the outside world. It can communicate to other vehicles or to the environment and make decisions according to the available data.

Furthermore, due to their openness, autonomous systems are continuously exposed to the unpredictability of the environment. This can lead to uncertainties in the system behaviours; it is, in fact, hard or impossible to explicitly model the behaviour of the system for each situation.

We are employing machine learning to build software agents that are flexible enough to deal with things that have potentially never seen before. By contrast, software that has been handcrafted for a specific purpose as soon as it faces a situation that the programmers did not expect might most probably crash.

Current approaches express the system's adaptability in terms of different configurations that are predefined at design-time and activated at run-time. What we propose in our method is to not pre-program all the system's behaviours and configurations but instead, we let the system to continuously evolve by learning the behaviours from the environment and from its mistakes. In this method we propose machine learning for generating the actions to be taken by the autonomous system and in this way, we plan to achieve the self-adaptability. There are many subfields in machine learning; we use reinforcement learning because it provides a system with the ability to generate actions based on data.

However, because these actions are not generated by logical deterministic algorithms we can not assure that they are always compliant with the system's requirements. In safety-critical systems, this can lead to hazardous situations. This is why for each generated action we monitor it against the system's safety-critical requirements, that we refer as *invariats*, and we execute only actions that do not violate them.

We plan to implement, test and validate our solution. The first step would be through a simulation of the decision-making algorithms and the monitoring process in the automotive domain [1]. Then we plan to integrate everything in ROS [2] and use miniaturized vehicles. Finally, we plan to use real vehicles using REVERE lab [3], seamlessly integrating machine-learning algorithms with the vehicles legacy code.

II. Proposed Method

We propose a method for delegating part of the decision-making process of an autonomous system to machine-learned algorithms while still preserving system's invariants. The main components of our approach are shown in Figure 1 and briefly described in below:

- *Perception and Execution*
  Autonomous systems require measurements and interaction with the environment. The perception of the environment is a semantic representation of the raw data coming from the sensors. The execution component applies the actions to the external environment. Both components communicate with the world model so that the latter can keep track of the actions generated and their effect on the environment.

978-1-5386-1589-8/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE-C.2017.40

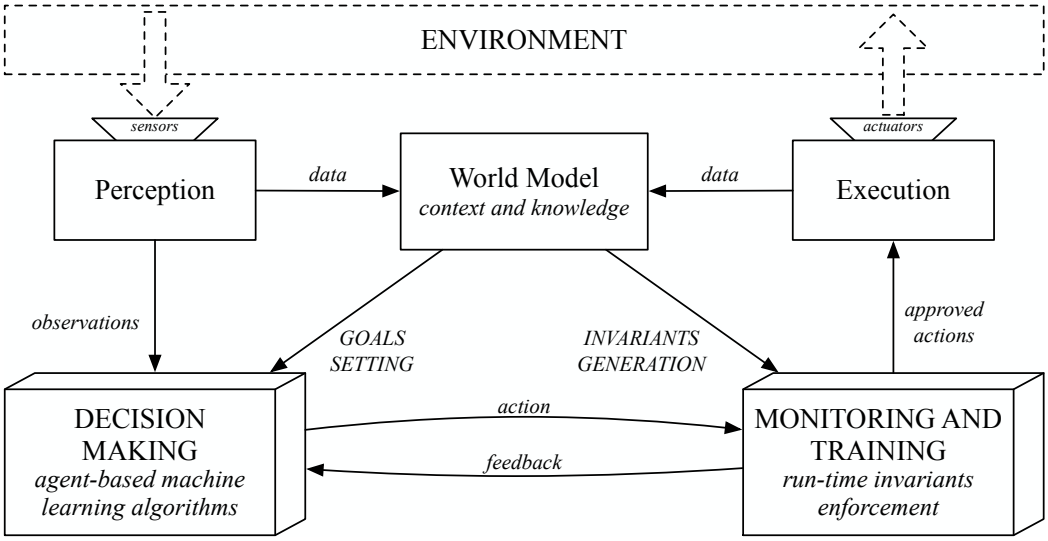

Fig. 1: Proposed method.

- *World Model* This component keeps track of both the external environment and the internal state of the system. At any time it also sets the *goals* to be achieved and the *invariants* to be maintained by the autonomous system.

- *Decision Making* It generates actions from the observations it receives from the perception component and from the goals it gets from the world model. The algorithms that run here are not pre-programmed but learned from the data. Furthermore, they continuously improve by receiving feedback for the generated actions from the monitoring and training component.

- *Monitoring and Training* This component has a double purpose. It *monitors* the actions taken by the decision-making component preventing those that violate system's invariants from being executed. It *trains* the machine-learning algorithms by sending feedback for each produced action so that they can learn and continuously improve.

Our approach utilizes reinforcement learning agents for decision-making [4]. If the actions produced by a decision-making agent violate any safety-critical invariant the monitoring process prevents these actions to be executed and *tame* the agent by sending feedback. This feedback has to be processed and properly expressed in terms of reward function. The reinforcement learning agent will get points for its correct actions and loose points for dangerous actions that also break some system's invariance.

III. Vision and road ahead

Our research aims to combine machine learning approaches with run-time strong assurances of the autonomous systems safety-critical requirements. We will focus on the integration of different state-of-the-art methods with the vision to build an autonomous system that can make safety-certifiable decisions.

We have started by analyzing today's safety standards such as the ISO26262 for the automotive industry and applying traditional approaches for verification such as formal methods. We have modeled different behaviours of semi-autonomous vehicles in a platooning formation using model-checking to verify its invariants [5]. However, this is not flexible enough as all the scenarios that we want to verify with model-checking must be modeled at design-time.

Now our goal is to model a system that continuously evolves using machine learning techniques to drive the system's adaptations. At the same time, we want to keep the preservation of the system's invariants by continuously monitoring how the system reacts to the changes in the environment. This is performed by a continuous run-time monitoring of the machine-learning generated actions. The outcome of the monitoring process also serves the purpose of *training* the decision-making agents so that they can continuously learn and provide better actions in the future.

Acknowledgment

This work was supported by the Wallenberg Autonomous Systems Program (WASP).

References

[1] N. Hiblot, D. Gruyer, J.-S. Barreiro, and B. Monnier, "Pro-sivic and roads. a software suite for sensors simulation and virtual prototyping of adas," in *Proceedings of DSC*, 2010, pp. 277–288.

[2] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009.

[3] "REVERE Lab," https://www.chalmers.se/safer/EN/projects/pre-crash-safety/projects/revere-research-vehicle, 2015.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[5] P. Mallozzi, M. Sciancalepore, and P. Pelliccione, "Formal verification of the on-the-fly vehicle platooning protocol," pp. 62–75, 2016.