More adaptive does not imply less safe (with formal verification)

Luca Pulina[1] and Armando Tacchella[2] ⋆

[1] POLCOMING, Università degli Studi di Sassari, Viale Mancini 5 – 07100 Sassari – Italy
[2] DIBRIS, Università degli Studi di Genova, Via Opera Pia, 13 – 16145 Genova – Italy
lpulina@uniss.it — armando.tacchella@unige.it

Abstract. In this paper we provide a concise survey of our work devoted to applying formal methods to check the safety of adaptive cyber-physical systems.

1 Introduction

In the past few years, the notion of cyber-physical system (CPS) emerged to define complex systems intertwining physical processes, hardware, software and communication networks. With respect to "classical" embedded systems, CPSs add elements of complexity including different spatial and temporal scales among components, multiple and distinct behavioral modalities, and context-dependent interaction patterns [1]. When considering *adaptive* (also *reconfigurable*) CPSs, we refer to implements capable of modifying their internal parameters to achieve and maintain a prescribed quality of service even in the face of a partially unknown and mutating environment. The addition of "adaptive" remarks the sharp distinction we draw between systems which only react according to prescribed control policies and systems which can learn and/or update their control policies. While adaptation is a desirable requirement for CPSs in many circumstances, most CPSs are deployed in applications where misbehavior can cause serious damage to the surrounding environment, which makes their safety a mandatory requirement. Unfortunately, adaptivity and safety are two conflicting propositions: safety can be increased by reducing the amount of automatic reconfiguration, while changing internal parameters during operation may yield unsafe control policies.

The vision behind our research is that the trade-off between safety and adaptivity could be reduced substantially by resorting to model-based design (MBD) techniques and formal methods. While MBD tools represents a steadily growing area in CPSs the application of formal methods is still confined to a niche. In our view, the availability of abstract system models from MBD tools is an enabler for analyzing those models in a precise way, and since it is impossible to foresee all the potential adaptations of a system in advance, formal verification is the only practical way to increase confidence in the correct adaptive behavior of the final implement. The research question is thus whether verification techniques conceived with non-adaptive systems in mind, can be borrowed and/or extended to verify (industry-scale) CPSs.

In the following, we divide our attempts to answer such question in two families. In Section 2 we consider the safety of *stateless* models, i.e., models whose purpose is to

⋆ The authors wish to thank their collaborators and colleagues Erika Ábrahám, Nils Jansen, Joost-Pieter Katoen, Francesco Leofante, Giorgio Metta, Lorenzo Natale, Shashank Pathak and Simone Vuotto, who contributed to the research herewith presented.

approximate functional implements. Our main contributions along this research stream involve safety of artificial neural networks [2,3,4] and kernel-based machines [5]. In Section 3 we consider *modal* models, i.e., representations of dynamical systems. Here we consider both hybrid systems [6] augmented with adaptive capabilities, and probabilistic systems [7,8,9,10], wherein models of environments and control policies are acquired through approximate dynamic programming.

2 Stateless models

In applications of CPSs, it is often the case that functional relationships between system variables are to be approximated and possibly updated to maintain optimal performances. Consider, for instance, the relationship between fuel and air intake in electronic injection systems. While interpolation of a fixed look-up table might suffice to determine the correct air intake, an adaptive approach might seek to find the best relationship based, e.g., on fuel quality, air relative humidity, and exhaust gas emissions. Both neural networks and kernel-based machines — see [4,5] for references — have been proved very successful in fulfilling these tasks by "learning" accurate mappings from data. However, in spite of some exceptions, their application is confined to non-safety related implements. The main reason is the lack of general, automated, yet effective safety assurance methods for learning systems.

Introduced for the first time in our work [4], verification of neural networks known as Multi-Layer Perceptrons (MLPs) can be carried out using abstraction-refinement techniques and Satisfiability Modulo Theory (SMT) solvers. The same approach was later extended to consider several safety-related conditions in [2], and to consider kernel-based machines in [5]. The key idea of the approach is that both MLPs and kernel-based machines are linear combinations of non-linear functions. Therefore, it is sufficient to abstract non-linear elements in order to obtain abstract machines whose input-output properties can be checked using quantifier-free linear arithmetic over reals (QF-LRA). Abstract machines are conservative over-approximations of concrete ones. Therefore, safety of abstract machines implies safety of concrete ones, whereas abstract counterexamples must be checked for realization — a process branded Counter-Example Triggered Abstraction Refinement (CETAR) in [4]. To a certain extent, spurious counterexamples can also be used to *repair* the network, i.e., improve its safety. To the best of our knowledge, this is the only contribution in the literature where formal methods are leveraged to improve the quality of a functional approximation.

The results obtained in [2] and [5] show that CETAR based on SMT solvers is applicable to small-to-medium sized networks. However, recent advancements in the machine learning community command for much larger and complex networks known as *Deep Neural Networks*. While the performances of such networks in terms of predictive power on a variety of tasks are impressive, they also feature some unexpected behaviors. For instance, in [1] it is shown that very small perturbations on input instances can cause dramatic effects on output results. This "instability" of deep neural networks was the inspiration behind recent contributions, see, e.g., [11,12]. In spite of these recent advancements, the problem of verifying large and complex networks is still an open question worth of further investigation.

3 Modal models

Modeling CPSs as a whole usually requires modal models. Furthermore, due to the interaction with physical processes, discrete-time finite-state models are not sufficient to capture all the subtleties of a CPS. Hybrid and/or probabilistic models are to be considered instead. With respect to the classical tasks of controller verification and synthesis, such models introduce additional computational issues which might make formal approaches untenable in practical applications. Adaptivity, i.e., learning parameters and/or control strategies, thickens the plot even further. Our research has focused on applicable formal methods for verification, synthesis and repair of controllers, considering the robotic domain as benchmarks for realistic, yet reasonably sized CPSs.

In [6] we considered a robot learning to play defense in the air hockey game. This setup is paradigmatic since the robot must see, decide and move fastly, but, at the same time, it must learn and guarantee that the control system is safe throughout the process. The (multi-agent) control system is comprised of a *vision* agent devoted to visual perception, a *motion control* agent sending position commands to the manipulator and a *coordination* agent converting stimuli into commands. The parameters of the coordination agent change over time, possibly improving on the robot's ability to intercept the puck. The system is unsafe if the manipulator moves too close to the table's edges. Agents are modeled as hybrid automata, and execution traces are checked for safety with HYSAT [13]. Because of learning, the whole system must be (re)verified eventually. The key idea is to preserve safety at all times by keeping safe – and possibly ineffective – parameters of the coordination agent in place, until a more effective – and definitely safe – setting is available. Experimental analysis in the air hockey setup shows that this approach can yield safety without heavily compromising on effectiveness.

In a series of papers started with [9], we considered the problem of synthesizing safe controllers using probabilistic models. In these works, we assume that the interaction between the robot and the environment can be modeled as a Markov Decision Process (MDP), and that a control strategy for the task at hand can be acquired by approximate dynamic programming — also known as Reinforcement Learning (RL). Here, the focus is on safety at the deliberative level, enabling a discrete-time, discrete state abstraction of the problem domain, where probabilistic effects account for noise in sensing and acting.[3] Since RL acquires (an implicit) system model and an (explicit) control strategy by trial and error, we postulate that learning is performed in a simulator and then the control strategy is deployed on the actual robot. The key idea is that, given a control strategy, an MDP becomes a Markov chain so that safety properties can be expressed using probabilistic temporal logic and verified using model checkers. In [9] we consider a task wherein a humanoid should grasp some object while avoiding others, whereas in [8] we consider a standing-up task for a small (19 degrees-of-freedom) robot. In both contributions we consider both the problem of verifying that a learned control strategy fulfills some requirements, and the problem of repairing it until it does. Moreover, in [8], we consider also the problem of monitoring that the (verified, repaired) control strategy maintains its properties once deployed on the actual robot.

[3] It is however assumed that the state can be detected with sufficient precision, i.e., we postulate full observability.

References

1. Edward A. Lee. Cyber physical systems: Design challenges. In *11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2008), 5-7 May 2008, Orlando, Florida, USA*, pages 363–369, 2008.
2. Luca Pulina and Armando Tacchella. Challenging SMT solvers to verify neural networks. *AI Commun.*, 25(2):117–135, 2012.
3. Luca Pulina and Armando Tacchella. NeVer: a tool for artificial neural networks verification. *Ann. Math. Artif. Intell.*, 62(3-4):403–425, 2011.
4. Luca Pulina and Armando Tacchella. An Abstraction-Refinement Approach to Verification of Artificial Neural Networks. In *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, pages 243–257, 2010.
5. Francesco Leofante and Armando Tacchella. Learning in Physical Domains: Mating Safety Requirements and Costly Sampling. In *AI*IA 2016: Advances in Artificial Intelligence - XVth International Conference of the Italian Association for Artificial Intelligence, Genova, Italy, November 29 - December 1, 2016, Proceedings*, pages 539–552, 2016.
6. Giorgio Metta, Lorenzo Natale, Shashank Pathak, Luca Pulina, and Armando Tacchella. Safe and effective learning: A case study. In *IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3-7 May 2010*, pages 4809–4814, 2010.
7. Shashank Pathak, Luca Pulina, and Armando Tacchella. Evaluating probabilistic model checking tools for verification of robot control policies. *AI Commun.*, 29(2):287–299, 2016.
8. Francesco Leofante, Simone Vuotto, Erika Ábrahám, Armando Tacchella, and Nils Jansen. Combining Static and Runtime Methods to Achieve Safe Standing-Up for Humanoid Robots. In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th Int.l Symp., ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part I*, pages 496–514, 2016.
9. Shashank Pathak, Luca Pulina, Giorgio Metta, and Armando Tacchella. Ensuring safety of policies learned by reinforcement: Reaching objects in the presence of obstacles with the iCub. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, pages 170–175, 2013.
10. Shashank Pathak, Luca Pulina, and Armando Tacchella. Verification and Repair of Control Policies for Safe Reinforcement Learning. *To appear in Applied Intelligence*, 2017.
11. Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. *arXiv preprint arXiv:1610.06940 – To appear as invited paper at CAV 2017*, 2016.
12. Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *arXiv preprint arXiv:1702.01135 – To appear in the proc. of CAV 2017*, 2017.
13. Martin Fränzle and Christian Herde. Hysat: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in System Design*, 30(3):179–198, 2007.