Establishing Trust in Remotely Reprogrammable Systems

Thomas Gurriet
Mechanical Engineering Department
thomas.gurriet@gatech.edu

Mark L. Mote
Aerospace Engineering Department
mmote3@gatech.edu

Aaron Ames
Mechanical Engineering Department
ames@gatech.edu

Eric Feron
Aerospace Engineering Department
eric.feron@aerospace.gatech.edu

Georgia Institute of Technology, Atlanta, GA 30332, USA

ABSTRACT

Remote reprogrammability can serve as a powerful and enabling tool, allowing widespread and rapid evolution of cyber-physical systems. Together with the effects of ever increasing automation, this tool is redefining the relationship between the designer, user, and their methods of interaction with the system. The cost of these advancements is a series of new challenges in terms of safety and security. This paper describes the need for a formal framework in which safety, performance, and all operational modes of the system can be described and analyzed efficiently. An approach to establishing trustworthiness of cyber-physical systems that allows for automated and efficient verification of the system is considered in the context of a concrete implementation.

Keywords
Cyber-physical, safety, software reliability, verification

INTRODUCTION

The emergence of new technologies for embedded systems, especially the miniaturization of electronics and the emergence of small and efficient sensors, have enabled a new realm of possibilities in the field of robotics. Machines can sense, they can act, and they are reaching the point where they can even think as we push the embedded computational power to levels never reached before. These robots are not just machines anymore, but truly interactive and aware systems where the "brain" and the "body" are becoming fundamentally entangled. Welcome the world of Cyber-physical Systems (CPS).

Machine intelligence is changing the way that humans interact with CPSs. Increasing automation of low level decisions shifts the control, and thus the burden of safety assurance, from operator to designer. While this frees the user to interact with system in extraordinary new ways, it also introduces significant design challenges in regards to safety and security. Assuring successful integration of these systems requires extensive research in the context of usage and performance, but perhaps more importantly in terms of safety. If we cannot guarantee control integrity, then it is not realistic to expect a satisfactory level of performance. It is necessary that both fronts, safety and performance be pushed in parallel.

Remotely reprogrammable systems - systems that allow some component of their software to be changed by an external entity - are enabling novel innovations in the field of CPS by allowing a component of adaptability in their design. Unfortunately, dynamic and evolving software adds significantly to the challenge of ensuring safety in the system. The already difficult task of verifying the behavior of a highly complex CPS is exacerbated by the need to repeat this verification process for each updated section of code. Traditionally, the flexibility of embedded software systems has been limited by the amount of reliability that can be compromised, however many safety critical systems such as aircraft or space probes do not allow themselves this luxury. New processes need to be developed to allow an expansion of the capabilities of these systems without sacrificing the critically important aspects of safety, performance, or cost.

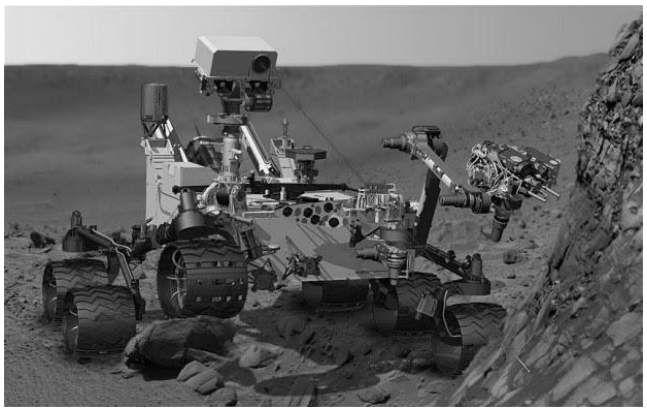
ENABLING INNOVATIONS

Remotely reprogrammable systems are those whose software components are partially or fully open to external reconfiguration via some network connection. Reprogramming a system allows it to either evolve to a new purpose, or become more effective than the current one by replacing or complementing its existing software. The applications of this concept are far-reaching, extending from the simple act of updating a smartphone application, to the complete revision of an aerospace control system. Looking specifically at the case of aerospace systems, an expansion of capability is not only beneficial but often necessary to meet evolving legal standards, such as tightening security requirements [19].

We can distinguish three unique architectures that arise with remotely accessible systems: (1) a central

HCI-Aero '16, September 14-16, 2016, Paris, France
© 2016 ACM. ISBN 978-1-4503-4406-7/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2950112.2964573

"development hub" with access to multiple platforms; (2) a server with access to a single platform; and (3) "open access" of multiple nodes (users) to a central system. The first allows the efficient expansion of a program to multiple platforms, e.g. pushing a software update to an application shared by many users. The second architecture is typically used to access systems whose location makes direct connection infeasible; such is the case with space vehicle design (e.g. satellites, planetary exploration rovers) [24], where remote programming is an absolute necessity. Finally, enabling many users with open access to a central platform is the foundation for many open access research facilities such as DeterLab [12] or SPHERES Laboratory [14]. These laboratories have the potential to become extremely powerful research enablers in the future of robotics and CPS systems by extending valuable hardware resources to external users.



3-D Rendering of Curiosity Rover on Mars

The potential of remotely reprogrammable systems is in general, limited by factors such as implementation cost and reliability. Introducing new software to engineering platforms can jeopardize the safety or functionality, and reliable software updates are often prohibitively expensive to implement. While reprogrammability may be compromising to the safety of the system, remote accessibility is compromising to the security. Specifically, open access opens a window of vulnerability through the possibility of hacking. For example, manipulating radio up-links to satellite services might have significant damaging effects on our communication infrastructure [7]. Remoteness poses additional challenges in terms of time delay and communication integrity.

Looking at the example of the open access laboratory, the addition of accessibility to the user makes the host more vulnerable. A possible solution to protect the local hardware may be to place the computational burden on the user's server, and only allow for a very narrow band of communication to the equipment. The safety of the system would be reduced to the safety of its input communication channel. However, this may pose challenges regarding the restriction in bandwidth and lag time that would arise, making the system less suitable for time-critical applications. For the full potential of remotely reprogrammable systems to be unleashed, we must devise cost effective strategies to mitigate these issues.

SAFETY AT THE CORE OF CPS

Though safety can be defined in a number of ways, it is at its core a characterization of the state of the system and its evolution. If the state space of the system and its subsets can be described easily, possibly at the price of abstraction, then characterizing system safety becomes more feasible.

The essence of safety is the partition of the system behavior in a set of "unsafe" behaviors and a set of "safe" behaviors. The characterization itself is usually based on the states that are harmful to humans, but this definition is purely arbitrary and is highly contextual. It would usually be defined in the system specifications during the early stages of design [11].

The difficulty arises when we try and combine the safety requirements with the performance requirements. Indeed, being in a safe state and not doing anything is certainly a way to assure safety, but it usually does not meet performance requirements either. Nevertheless, it is sometimes used as a last resort solution [22]. The difference between human operated and autonomous systems is that traditionally the responsibility of assuring safety was up to the operator, whereas for autonomous systems these safety requirements need to be infused into the system itself. A compromise between performance and safety must then be defined so as to maximize the end users freedom while assuring constant safety of the system.

Another ambiguity arises when we start to consider the failure modes as states of the system. Failed states should definitely be characterized, but this task is not trivial. Where should one draw the line between a system functioning in a degraded mode of operation and a cloud of debris resulting from being struck by a missile? The definition of the system's integrity should also be clearly stated in restricting the size of the state space.

If the specifications are not complete in the sense that they don't fully define the system and the safety properties we associate to that system, making formal statements about the safety aspect of the system becomes impossible. A formalism that encompasses these three aspects (system definition, performances and safety) needs to be developed in order to carry forward with formal analysis of CPSs. What makes our problem unique is the extent to which the state-space and the state-space transitions can be changed. Today, certification is only done for a fixed transition system but what we actually need are certifications for all possible transition systems. As these systems become more and more complex, a next generation of design tools, that intelligently guide designers, detect and mediate or even anticipate human error, needs to be created.

ESTABLISHING TRUSTWORTHINESS

Once we have defined our "safe" states, the goal is to design a system that stays in these safe regions. This can be done either mechanically through static measures of

protection between the systems and their environment, or through feedback control of the systems. But for safety-critical systems, we cannot blindly trust these algorithms without some kind of verification of what they produce in terms of the behavior of the system [13].

How to establish trustworthiness of these CPSs is still an open question but we can distinguish two ways of addressing this fundamental problem. On one hand, we can either create a formalism that would capture the entire behavior of the system and prove safety properties about that system. On the other hand if we are not able to extract and exploit meaningful semantics out of the system, we can look at the problem from a probabilistic point of view. Obviously, staying formal from the specifications to the actual hardware would be preferable, but is difficult and we may have to compromise absolute guaranties for probabilistic ones. Also designing the system so as to minimize the impact of parameter uncertainty on the overall behavior of the system is definitely complementary to the verification work [25].

As we make progress toward the completeness of a formal verification framework [8], we must also put safeguards in the system. Ideally, we want this system to be totally robust to what happened upstream and to assure absolute safety of the hardware. A promising tool we are currently developing is Control Barrier Functions (CBF) [3]. The idea is to mathematically define the safe set for the system and to determine in real time the set of inputs that guarantee the forward invariance of that safe set. We then send to the robot an input that is as close as possible to the original command but that satisfies this Barrier Condition to guarantee safety. In practice, one just needs to solve a quadratic optimization problem with the proper inequality constraints for set invariance and input limits. However, this framework still needs to take actuator limits into consideration when synthesizing CBFs to assure set invariance in a realistic context, and still needs to fully characterize the robustness of these CBF based controllers [26]. A formal verification of the algorithms used for the barrier functions will still have to be performed, as these CBF based safety nets are based on complex real time optimization algorithms [17]. We would start with simple safeguard that would have been proven to be safe but that may substantially restrict the user, and then build on these to synthesize optimal safeguards.

CONCLUSION

Robotics has become one of the most fruitful areas of research of the beginning of this millennia, enabling the seamless integration of CPS into our everyday lives. Remotely Reprogrammable systems enable a new realm of possibilities within CPS by allowing systems to adapt. However, in spite of their many advantages, the potential of remotely reprogrammable systems is limited by factors such as safety, security, reliability, and the cost of software updates. For the full potential of remotely reprogrammable systems to be unleashed, we must first innovate toward cost effective solutions to these problems. We have seen that without a formal definition of safety, formal verification of the system is impossible. Working on the overall formalism to address CPSs safety and performances is the way to go for allowing more complex but provably safe capabilities of these systems. If this process is already used at the software level, it usually requires human intensive verification and would need to be much more automated in order to make the design and operation of aerospace systems more efficient and affordable. As the role of the human in the loop evolves, and the burden of safety shifts ever further toward an active designer, the need grows for an efficient and robust framework for formally guaranteeing the safety of a system.

REFERENCES

1. Alexander, D. S., Arbaugh, W., Keromytis, A. D., & Smith, J. M. (1998). Safety and security of programmable network infrastructures. Communications Magazine, IEEE, 36(10), 84-92.
2. Alle, M., Varadarajan, K., Fell, A., Joseph, N., Das, S., Biswas, P., ... & Narayan, R. (2009). Redefine: Runtime reconfigurable polymorphic asic. ACM Transactions on Embedded Computing Systems (TECS), 9(2), 11.
3. Ames, A. D., Grizzle, J. W., & Tabuada, P. (2014, December). Control barrier function based quadratic programs with application to adaptive cruise control. In Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on (pp. 6271-6278). IEEE.
4. ARP4754, S. A. E. (1996). Certification considerations for highly-integrated or complex aircraft systems. SAE, Warrendale, PA.
5. ARP, S. (1996). 4761. Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment, 12.
6. Boy, G. A. (2014). From automation to tangible interactive objects. Annual Reviews in Control, 38(1), 1-11.
7. Fritz, Jason. "Satellite hacking: A guide for the perplexed." Culture Mandala: The Bulletin of the Centre for East-West Cultural and Economic Studies 10.1 (2013): 3.
8. Jeannin, J. B., et al. (2015). A formally verified hybrid system for the next-generation airborne collision avoidance system. In Tools and Algorithms for the Construction and Analysis of Systems (pp. 21-36). Springer Berlin Heidelberg.
9. Lee, E. (2008, May). Cyber physical systems: Design challenges. In Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on (pp. 363-369). IEEE.
10. Leveson, N. G., & Weiss, K. A. (2004, October). Making embedded software reuse practical and safe. In ACM SIGSOFT Software Engineering Notes (Vol. 29, No. 6, pp. 171-178). ACM.

11. Leveson, N. G. (1998, April). Intent specifications: An approach to building human-centered specifications. In Requirements Engineering, 1998. Proceedings. 1998 Third International Conference on (pp. 204-213). IEEE.
12. Mirkovic, Jelena, and Terry Benzel. "Teaching cybersecurity with DeterLab." Security & Privacy, IEEE 10.1 (2012): 73-76.
13. Mitsch, S., & Platzer, A. (2014, January). ModelPlex: Verified runtime validation of verified cyber-physical system models. In Runtime Verification(pp. 199-214). Springer International Publishing.
14. Miller, David, et al. "SPHERES: a testbed for long duration satellite formation flying in micro-gravity conditions." Proceedings of the AAS/AIAA Space Flight Mechanics Meeting, Clearwater, FL, Paper No. AAS 00-110. 2000.
15. Nancy G. Leveson The role of software in spacecraft accidents. AIAA Journal of Spacecraft and Rockets, Vol 41, No. 4, July 2004.
16. Owre, S., Rushby, J. M., & Shankar, N. (1992). PVS: A prototype verification system. In Automated Deduction—CADE-11 (pp. 748-752). Springer Berlin Heidelberg.
17. Roozbehani, Mardavij, Eric Feron, and Alexandre Megrestki. "Modeling, optimization and computation for software verification." Hybrid Systems: Computation and Control. Springer Berlin Heidelberg, 2005. 606-622.
18. RTCA/DO-178B. Software Considerations in Airborne Systems and Equipment Certification.
19. SAE committee developing standards to tackle cybersecurity threat. 2015. Aviation week. Web. 29 January 2016.
20. Saenz-Otero, A. (2005). Design principles for the development of space technology maturation laboratories aboard the International Space Station.Massachusetts Institute of Technology, Department of Aeronautics and Astronatucis, Ph. D. Thesis, Cambridge, MA.
21. Tomlin, C. J., Mitchell, I., Bayen, A. M., & Oishi, M. (2003). Computational techniques for the verification of hybrid systems. Proceedings of the IEEE, 91 (7), 986-1001.
22. The 9/11 commission report: Final report of the national commission on terrorist attacks upon the United States. Government Printing Office, 2011.
23. Wang, T., Jobredeaux, R., Pantel, M., Garoche, P. L., Féron, É., & Henrion, D. (2014). Credible Autocoding of Convex Optimization Algorithms. arXiv preprint arXiv: 1403.1861.
24. Weiss, Kathryn Anne. "The Mars Science Laboratory: Flight Software-A Platform for Science and Mobility." (2012).
25. Wu, Y., & Wu, A. (2000). Taguchi methods for robust design. American Society of Mechanical Engineers.
26. Xu, X., et al. (2015). Robustness of control barrier functions for safety critical control. IFAC-PapersOnLine, 48(27), 54-61.
27. Zimmerman, M., Rodriguez, M., Ingram, B., Katahira, M., De Villepin, M., & Leveson, N. (2000). Making formal methods practical. In Digital Avionics Systems Conference, 2000. Proceedings. DASC. The 19th (Vol. 1, pp. 1B2-1). IEEE.