Automated Verification and Synthesis of Embedded Systems using Machine Learning

Lucas Cordeiro
Department of Computer Science, University of Oxford, UK
E-mail: lucas.cordeiro@cs.ox.ac.uk

ABSTRACT

The dependency on the correct functioning of embedded systems is rapidly growing, mainly due to their wide range of applications, such as micro-grids, automotive device control, health care, surveillance, mobile devices, and consumer electronics. Their structures are becoming more and more complex and now require multi-core processors with scalable shared memory, in order to meet increasing computational power demands. As a consequence, reliability of embedded (distributed) software becomes a key issue during system development, which must be carefully addressed and assured. The present research discusses challenges, problems, and recent advances to ensure correctness and timeliness regarding embedded systems. Reliability issues, in the development of micro-grids and cyber-physical systems, are then considered, as a prominent verification and synthesis application. In particular, machine learning techniques emerge as one of the main approaches to learn reliable implementations of embedded software for achieving a *correct-by-construction* design.

1. INTRODUCTION

Generally, embedded computer systems perform dedicated functions with a high degree of reliability. They are used in a variety of sophisticated applications, which range from entertainment software, such as games and graphics animation, to safety-critical systems, including nuclear reactors and automotive controllers [1]. Embedded systems are ubiquitous in modern day information systems, and are also becoming increasingly important in our society, especially in micro-grids, where reliability and carbon emission reduction are of paramount importance [2], and in cyber-physical systems (CPS), which demand short development cycles and again a high-level of reliability [3]. As a consequence, human life has also become more and more dependent on the services provided by this type of system and, in particular, their success is strictly related to both service relevance and quality.

Figure 1 shows embedded systems examples, which typically consist of a human-machine interface (*e.g.*, keyboard and LCD), a processing unit (*e.g.*, real-time computer system), and an instrumentation interface (*e.g.*, sensor, network, and actuator) [1]. Indeed, many current embedded systems, such as unmanned aerial vehicles (UAVs) [4] and medical monitoring systems [5], become interesting solutions only if they can reliably perform their target tasks. Besides, when physical interaction with the real world is needed, which happens in CPS, additional care must be taken, mainly when human action is directly replaced, as in vehicle driving. Regarding the latter, even human-in-the-loop feedback control can be employed, which raises deeper concerns w.r.t. reliability of human behavior modeling and system implementation.

Consequently, it is important to go beyond design correctness and also address behavior correctness, which may be performed by incorporating system models. In particular, these models can be
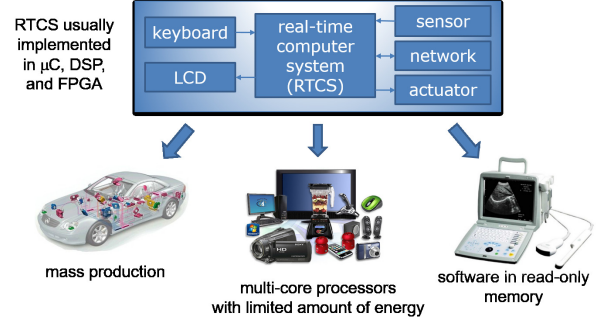

Figure 1: An embedded system is part of a well-specified larger system (intelligent product).

used for synthesizing a given system, ensuring that all needed functions are correctly implemented and the correct behavior exhibited, i.e., the system is indeed correct by its method of construction [6]. Here, machine learning emerges as a powerful technique to automatically learn the correct behavior of the system, which must provably satisfy a given correctness specification σ. Specifically, synthesizers can use σ as starting point and then incrementally produce a sequence of candidate solutions that satisfy σ, by integrating deductive methods with inductive inference (learning from counterexamples) [7]. As a result, a given candidate solution can be iteratively refined to match the specification σ based on a counterexample-guided learning approach.

2. VERIFICATION AND SYNTHESIS CHALLENGES FOR EMBEDDED SYSTEMS

State-of-the-art verification methodologies for embedded systems generate test vectors (with constraints) and use assertion-based verification and high-level processor models, during simulation [8], as shown in Figure 2. Here, the main challenges regarding the verification of embedded systems lie on improving coverage, pruning the state-space exploration during verification, and incorporating system models, which allow specific checks regarding system behavior and not only code correctness. Additionally, embedded system verification raises additional challenges, such as: (1) time and energy constraints; (2) handling of concurrent software; (3) platform restrictions; (4) legacy designs; (5) support to different programming languages and interfaces; and (6) handling of non-linear and non-convex optimization functions.

Indeed, the first two aspects are of extreme relevance in micro-grids and cyber-physical systems, in order to ensure reliability, which is a key issue for (smart) cities, industries, and consumers, and the third one is essential in systems that implement device models, such as digital filters and controllers, which present a behavior that is highly dependent on signal inputs and outputs and whose deployment may be heavily affected by hardware restric-
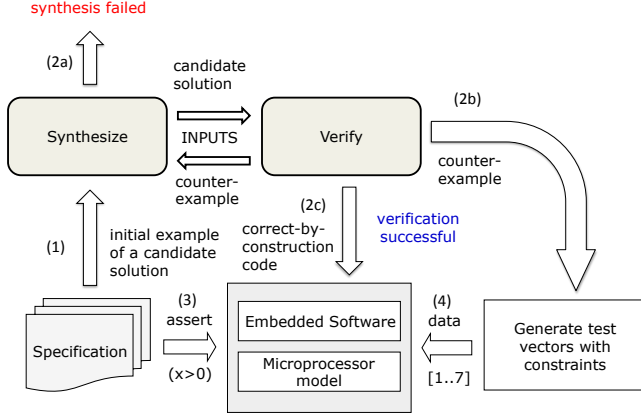
Figure 2: Verification and synthesis methodologies for embedded systems.

tions. The fourth aspect is inherent to a large number of embedded systems from telecommunications, control systems, and medical devices. In particular, software developed for those systems has been extensively tested and verified, and also optimized for efficiency over years. Therefore, when a new product is derived from a given platform, a lot of legacy code is usually reused for reducing development time and improving code quality. The fifth aspect is related to the evolution of development processes and technologies, which may delay the application of suitable verification and synthesis approaches if verifiers and synthesizers do not support different programming languages and interfaces. The last one is related to the widespread use of embedded systems in autonomous vehicle navigation systems, which demand optimization solving during their execution for a wide range of functions, including non-linear and non-convex optimization functions.

Those challenges place difficulties for developing (reliable) synthesizers for embedded systems, especially for CPS and micro-grids, where the controlled object (*e.g.*, physical plant) typically exhibits continuous behavior whereas the controller (usually implemented by a real-time computer system) operates in discrete time and over a quantized domain (cf. intelligent product in Figure 1). In particular, synthesizers for those systems need to consider the effects of the quantizers (A/D and D/A converters), when a digital equivalent of the controlled object is considered, *i.e.*, a model of their physical environment. Additionally, finite-precision arithmetic and their related rounding errors need to be considered when correct-by-construction code is generated for embedded systems. The main challenge lies on exploiting effectively and efficiently counterexamples provided by verifiers to automatically learn reliable embedded software implementations (cf. Figure 2).

3. RESEARCH PROBLEM (RP)
This research statement tackles six major problems in computer-aided verification and synthesis for embedded systems, which are (partially) open in current published research.

(RP1) provide suitable encoding into SMT, which may extend the background theories typically supported by SMT solvers, with the goal of reasoning accurately and effectively about realistic embedded (control) software.

(RP2) exploit SMT techniques to leverage bounded model checking of multi-threaded software, in order to mitigate the state-explosion problem due to thread interleaving.

(RP3) prove correctness and timeliness of embedded systems, by taking into account stringent constraints imposed by hardware.

(RP4) incorporate knowledge about system purpose and associated features to detect system-level and behavior failures.

(RP5) provide tools and approaches capable of addressing different programming languages and application interfaces, with the goal of reducing the time needed to adapt current verification techniques to new developments and technologies.

(RP6) develop automated synthesis approaches that are algorithmically and numerically sound, in order to handle embedded (control) software that is tightly coupled with the physical environment by considering uncertain models and FWL effects.

4. CURRENT ACHIEVEMENTS AND FUTURE TRENDS
In order to support SMT encoding **(RP1)**, Cordeiro, Fischer, and Marques-Silva proposed the first SMT-based BMC for full C programs, called Efficient SMT-Based Context-Bounded Model Checker (ESBMC) [9], which was later extended to support C++, CUDA, and Qt-based consumer electronics applications. This approach was also able to find undiscovered bugs related to arithmetic overflow, buffer overflow, and invalid pointer, in standard benchmarks, which were later confirmed by the benchmarks' creators (*e.g.*, NOKIA, NEC, NXP, and VERISEC). Other SMT-based BMC approaches have also been proposed and implemented in the literature [10, 11], but the coverage and verification time of all existing ones are still limited to specific classes of programs, especially for those that contain intensive floating-point arithmetic and dynamic memory allocation. One possible research direction is to bridge the gap between BMC tools and SMT solvers to propose background theories and develop more efficient decision procedures to handle specific classes of programs.

The SMT-based BMC approach proposed by Cordeiro, Fischer, and Marques-Silva was further developed to verify correct lock acquisition ordering and the absence of deadlocks, data races, and atomicity violations in multi-threaded software based on POSIX and CUDA libraries **(RP2)** [12, 13], considering monotonic partial-order reduction and state-hashing techniques, in order to prune the state-space exploration. Recent advances for verifying multi-threaded C programs have been proposed to speed up the verification time, which significantly prune the state-space exploration; however, the class of concurrent programs (*e.g.*, OpenCL and MPI) that can be verified is still very limited. One possible research direction is to further extend BMC of multi-threaded programs via Sequentialization [14] and also analyze interpolants to prove non-interference of context switches [15].

Novel approaches to model check embedded software using k-induction and invariants were proposed and evaluated in the literature **(RP3)**, which demonstrate its effectiveness in some real-life embedded-system applications [16]. However, the main challenge still remains open, *i.e.*, to compute and strengthen loop invariants to prove program correctness and timeliness in a more efficient and effective way, in order to be competitive with other model-checking approaches [17]. In particular, invariant-generation algorithms have substantially evolved over the last years, with the goal of discovering inductive invariants of programs or continuously refine them during verification [18]. Yet there is still a lack of studies for exploiting the combination of different invariant-generation algorithms (*e.g.*, interval analysis, linear inequalities, polynomial equalities and inequalities) and how to strengthen them.

State-of-the-art SMT-based context-BMC approaches were ex-

tended to verify overflow, limit cycle, stability, and minimum phase, in digital systems **(RP4)**. Indeed, digital filters and controllers were tackled, in order to verify system-level properties of those systems, specified as linear-time temporal logic [19, 20]. In particular, a specific UAV application was tackled, with the goal to verify its attitude controllers. In general, however, there is still a lack of studies to verify system-level properties related to embedded systems; emphasis should be given to micro-grids and cyber-physical systems, which require high-dependability requirements for computation, control, and communication. Additionally, the application of automated fault detection, localization, and correction techniques to digital systems represents an important research direction to make BMC tools useful for engineers.

Although ESBMC was extended to support C/C++ and some variants **(RP5)**, new application interfaces and programming languages are often developed, which require suitable verifiers. Indeed, it would be interesting if a new programming language model could be loaded, which along with a BMC core could check different programs. Some work towards that was already presented by [21], which employed operational models for checking Qt-based programs from consumer electronics. In summary, the BMC core is not changed, but instead an operational model, which implements the behavior and features of Qt libraries, is used for providing the new code structure to be checked. Such research problem is closely related to the first one **(RP1)** and has the potential to devise a new paradigm in software verification.

State-of-the-art synthesis approaches **(RP6)** for embedded (control) systems typically disregard the platform in which the embedded system software operates and restrict itself to generate code that do not take into account FWL effects. However, the synthesized system must include the physical plant to avoid serious system's malfunctioning (or even a catastrophe) due to the embedded (control) software, *e.g.*, the Mars Polar Lander did not account for leg compressions prior to landing [22]. Research in this direction has made some progress to design, implement, and evaluate an automated approach for generating correct-by-construction digital controllers that is based on state-of-the-art inductive synthesis techniques [6]. However, there is still little evidence whether that approach can scale for larger systems modeled by other types of representations (*e.g.*, state-space). Another research direction for synthesizers is to automatically produce UAV trajectory and mission planning code, by taking into account system's dynamics and nonholonomic constraints. As a result, verifiers and synthesizers need to handle a wide range of functions, including non-linear and non-convex optimization problems [23]. Machine learning techniques could be employed here to learn from counterexamples, *i.e.*, in the inductive step, synthesizers could learn the model from raw data, and in the deductive step, the model could be applied to predict the behaviour of new data [24].

5. CONCLUSIONS
This research statement presented the main challenges related to the verification of design correctness, in embedded systems, and also raised some important side considerations about synthesis. Given that software complexity has significantly increased in embedded products, there is still the need for stressing and exhaustively covering the entire system state space, in order to verify low-level properties that have to meet the application's deadline, access memory regions, handle concurrency, and control hardware registers. Besides, there is a trend towards incorporating knowledge about the system to be verified, which may take software verification and synthesis one step further, where not only code correctness will be addressed, but also full system reliability. Finally, it seems interesting to provide behavioral models when new application interfaces or programming language features are used, in order to extend the capabilities of current verification tools,

without changing the core BMC module.

As future perspective, the main goal of this research is to extend BMC as a verification and synthesis tool for achieving correct-by-construction embedded system implementations. Special attention will be given to CPS and modern micro-grids, considering small-scale versions of a distributed system, so that reliability and other system-level properties (*e.g.*, carbon emission reduction in smart cities) are amenable to automated verification and synthesis, probably through behavior models.

6. REFERENCES
[1] H. Kopetz: *Real-Time Systems - Design Principles for Distributed Embedded Applications*. Real-Time Systems Series, Springer, ISBN 978-1-4419-8236-0, pp. 1–376, 2011.
[2] Xua X., Jiaa H., Wanga D., Yub D., Chiangc H.: *Hierarchical energy management system for multi-source multi-product microgrids*. Renewable Energy, v. 78, pp. 621–630, 2015.
[3] Lee E.: *The Past, Present and Future of Cyber-Physical Systems: A Focus on Models*. Sensors 15(3): pp. 4837–4869, 2015.
[4] Groza A., Letia I., Goron A., Zaporojan S.: *A formal approach for identifying assurance deficits in unmanned aerial vehicle software*. Progress in Systems Engineering, Springer, pp. 233–239, 2015.
[5] Cordeiro L., Fischer B., Chen H., Marques-Silva J.: *Semiformal Verification of Embedded Software in Medical Devices Considering Stringent Hardware Constraints*. ICESS, pp. 396–403, 2009.
[6] Abate A., Bessa I., Cattaruzza D., Cordeiro L., David C., Kesseli P., Kroening D.: *Sound and Automated Synthesis of Digital Stabilizing Controllers for Continuous Plants*. HSCC, 2017 (to appear).
[7] Sanjit A. Seshia: *Combining Induction, Deduction, and Structure for Verification and Synthesis*. Proc. of the IEEE 103(11): pp. 2036–2051, 2015.
[8] Behrend J., Lettnin D., Gruenhage A., Ruf J., Kropf T., Rosenstiel W.: *Scalable and Optimized Hybrid Verification of Embedded Software*. J. Electronic Testing 31(2): pp. 151–166, 2015.
[9] Cordeiro L., Fischer B., Marques-Silva J.: *SMT-based Bounded Model Checking for Embedded ANSI-C Software*. IEEE Trans. Software Eng. 38(4), pp. 957–974, 2012.
[10] Armando A., Mantovani J., Platania L.: *Bounded Model Checking of Software Using SMT Solvers Instead of SAT Solvers*. SPIN, LNCS 3925, pp. 146-162, 2006.
[11] Merz F., Falke S., Sinz C.: *LLBMC: Bounded Model Checking of C and C++ Programs using a Compiler IR*. VSTTE, LNCS 7152, pp. 146–161, 2012.
[12] Cordeiro L., Fischer B.: *Verifying Multi-threaded Software using SMT-based Context-Bounded Model Checking*. ICSE, pp. 331–340, 2011.
[13] Pereira P. Albuquerque H., Marques H., Silva I., Carvalho C., Santos V., Ferreira R., Cordeiro L.: *SMT-Based Context-Bounded Model Checking for CUDA Programs*. Concurrency and Computation: P&E., 2017 (to appear).
[14] Inverso O., Tomasco E., Fischer B., La Torre S., Parlato G.: *Bounded Model Checking of Multi-threaded C Programs via Lazy Sequentialization*. CAV, LNCS 8559, pp. 585-602, 2014.
[15] K. McMillan: *Widening and Interpolation*. SAS, LCNS 6887, pp. 1, 2011.
[16] Gadelha M., Ismail H., Cordeiro L.: *Handling Loops in Bounded Model Checking of C Programs via k-Induction*. STTT 19(1), pp. 97–114, 2017.
[17] Rocha, W., Rocha, H., Ismail H., Cordeiro, L., Fischer, B.: *DepthK: A k-Induction Verifier Based on Invariant Inference for C Programs (Competition Contribution)*. TACAS, 2017 (to appear).
[18] Beyer D., Dangl M., Wendler P.: *Boosting k-Induction with Continuously-Refined Invariants*. CAV, LNCS 9206, pp. 622–640, 2015.
[19] Morse J., Cordeiro L., Nicole D., Fischer B.: *Model Checking LTL Properties over ANSI-C Programs with Bounded Traces*. Software and System Modeling 14(1): pp. 65–81, 2015.
[20] Bessa I., Ismail H., Palhares, R., Cordeiro L., Chaves Filho J.: *Formal Non-Fragile Stability Verification of Digital Control Systems with Uncertainty*. IEEE Trans. Computers 66(3), pp. 545–552, 2017.
[21] Sousa F., Garcia M., Cordeiro, L., Lima Filho E.: *Bounded Model Checking of C++ Programs based on the Qt Cross-Platform Framework*. Softw. Test., Verif. Reliab., 2017 (to appear).
[22] Jackson D., Vaziri M.: *Correct or usable? the limits of traditional verification*. ESEC/SIGSOFT FSE, pp. 11, 2016.
[23] Araujo, R., Albuquerque, H., Bessa, I., Cordeiro, L., Chaves Filho, J.: *Counterexample Guided Inductive Optmization*. Sci. Comput. Program., 2017 (under review).
[24] Alur R., Bodík R., Juniwal G., Martin M., Raghothaman M., Seshia S., Singh R., Solar-Lezama A., Torlak E., Udupa A.: *Syntax-guided synthesis*. FMCAD, pp. 1–8, 2013.