IET Journals
The Institution of Engineering and Technology

Automated Verification and Synthesis of Cyber-Physical Systems: Advances and Challenges

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Lucas C. Cordeiro[1]* *Eddie B. de Lima Filho*[2] *Iury V. de Bessa*[3]

[1] *Department of Computer Science, University of Oxford, Oxford, United Kingdom*
[2] *TPV Technology, Manaus, Brazil*
[3] *Department of Electricity, Federal University of Amazonas, Manaus, Brazil*
* *E-mail: lucasccordeiro@gmail.com*

Abstract: Dependency on the correct operation 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, telecommunications, internet of things, and consumer electronics. Their structures are becoming more and more complex and now require multi-core processors with scalable shared memory, signal-processing pipelines, and sophisticated software modules, in order to meet increasing computational power, flexibility demands, and adaptation to new scenarios and behaviors. Additionally, interaction with real-world entities and addition of modern communication capabilities further enhance the mentioned features and give rise to a new class of systems: embedded & cyber-physical systems (ECPSs). As a consequence, reliability of ECPSs becomes a key issue during system development, which must be carefully addressed and assured. Normally, state-of-the-art verification methodologies for ECPSs generate test vectors (with constraints) and use assertion-based verification and high-level processor models, during simulation. Nonetheless, other additional challenges have been raised: the need for meeting time and energy constraints, handling concurrent software, dealing with platform restrictions, evaluating implementation-structure choices, validating operation logic, ensuring correct system behavior together with physical plants, providing compliance with target systems, incorporating knowledge about new problems and conditions, and supporting new software architectures and legacy designs. The present survey discusses challenges, problems, and recent advances to ensure correctness and timeliness regarding ECPSs. Reliability issues, in the development of ECPSs, are then considered, as a prominent verification and synthesis application for achieving a *correct-by-construction* design.

1 Introduction

Generally, embedded computer systems perform dedicated functions with high degree of reliability, according to their original design and requirements (*e.g.*, real-time) [1]. They are ubiquitous in modern day information systems and are also becoming increasingly important in our society, due to their use to process, monitor, control, and even replace every human activity: factories, power plants, mobile devices, robotics, traffic control, vehicles, and home duties [2]. Besides, they are also used in a variety of sophisticated applications, which range from entertainment software, such as games and graphics animation, to safety-critical systems, such as nuclear reactors and automotive controllers.

The interaction between embedded software and physical processes created a different class of systems, which are complex, highly integrated, and present a mixture of continuous and discrete dynamics, named as *hybrid systems* (HSs). Indeed, the embedded-system domain allied to the most recent communication revolution caused an intense integration between different and spread physical processes, which are called embedded & cyber-physical systems (ECPSs) [3]. In particular, the presence of modern communication technologies (*e.g.*, internet of things - IoT) causes a revolution in terms of flexibility, scalability, and complexity of those systems and adds a novel class of problems to the challenges found in the embedded systems domain. For example, micro-grids, *i.e.*, small-scale electricity systems that gather different sources of distributed generation (DG) and loads, are emerging ECPSs, where reliability and carbon emission reduction are of paramount importance [2]. One may also notice that each DG element and the majority of current loads already employ embedded software and present several safety-critical requirements; however, the integration of both constitute an ECPS, where additional challenges related to synchronization, stability, control, communication, and reliability of the entire micro-grid arise.

Thus, ECPSs demand short development cycles and high-level of reliability and robustness [3, 4], besides presenting challenges that include but are not limited to the ones already imposed by embedded systems. As a consequence of their popularization, human life has also become more and more dependent on services provided by this type of system and, in particular, their success is strictly related to both service relevance and quality/reliability.

Figure 1 shows some examples of ECPSs, which typically consist of a human-machine interface (*e.g.*, keyboard and display), a processing unit (*e.g.*, real-time computer system), and an instrumentation interface (*e.g.*, sensor, network, and actuator) that can be connected to some physical plant [1]. Indeed, many current ECPSs, such as unmanned aerial vehicles (UAVs) [5] and medical monitoring systems [6], become interesting solutions only if they can reliably perform their target tasks. For instance, UAVs are a trend on military missions and civil applications, since they can easily achieve places that cannot be accessed by humans without on-board pilots with different degrees of autonomy; however, an incorrect plan execution may cost civilian lives, which is unacceptable. In addition, wrong disease diagnosis or condition-evaluation reports have the potential to compromise patients' health and even many aspects of their lives, with serious consequences. On the one hand, portable ECPSs are capable of monitoring and identifying conditions that are very difficult for a physician, mainly when his contact with patients happens only in medical clinics. On the other hand, wrong or incomplete diagnostic data may delay necessary treatments or deteriorate patients' health.

Besides, when physical interaction with the real world is needed, which happens in ECPSs, additional care must be taken, mainly when human action is directly replaced, as in vehicle driving.

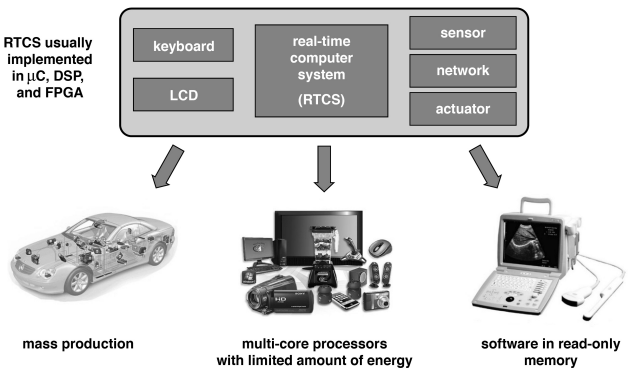

Fig. 1: An ECPS is part of a well-specified larger one (intelligent product).

Regarding the latter, even human-in-the-loop feedback control can be employed [7], which then raises deeper concerns related to 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. Specifically, such models can be used to evaluate and even synthesize a given particular system, by ensuring that all needed functions are correctly implemented and the correct behavior is exhibited, *i.e.*, the system is indeed correct by its method of construction [8].

For instance, one may argue that a Linux device driver [9] was verified and correctly uses the related kernel services (*e.g.*, exported symbols and scheduling); however, not much can be said about correct access to the associated hardware and its use, *i.e.*, if *read* and *write* operations are performed according to what is specified in the device's datasheet and setup periods are respected, for instance. In summary, the main goal would be to ensure that the developed driver is compliant with the underlying hardware, which goes further than code correctness and also tackles interaction with surrounding entities.

Regarding behavior correctness, one important observation is that it is becoming more difficult, since fixed mathematical models are being replaced by machine learning approaches (ML) [10], where future operation may change based on data acquired during previous actions, due to model updating. Indeed, ML algorithms have a dual role on software verification development. With the dissemination of the use of these algorithms, several ECPS have embedded ML software, and as a consequence, system-correctness verification procedures should also adapt somehow and incorporate new checks, which may also rely on ML, when analyzing target models or fed with acquired data, as recently reported in the literature [11, 12], where formal verification techniques are applied to verify deep neural networks. Otherwise, model checking mechanisms have also incorporated ML algorithms to increase the efficiency of verification [13–15] and synthesis [16] procedures.

A number of distinctive characteristics might influence the ECPS verification and synthesis process, which include: mass production and static structure, functionality determined by software in read-only memory, multi-core processors with scalable shared memory, and limited amount of energy. Additionally, increasing computational power and decreasing size and cost, which are common to the area of computer processors, are enabling system designers to move more features to software domain, which consequently leads to difficulties in verifying design correctness, since stringent constraints imposed by the underlying hardware (*e.g.*, real-time, memory allocation, interrupts, interfacing, and concurrency) [17] or even new structures provided with the goal of ensuring more computational capacity [18] must be considered during verification. Such observations expand the addressed issues and even complement what was previously mentioned.

The present article is an survey study whose main goals are described as follows:

- briefly present part of the established research about ECPS modelling, verification and synthesis;
- provide an overview about the technical issues and advances in symbolic formal verification and synthesis methodologies that are currently employed for ECPS;
- present the open challenges on the ECPS verification and synthesis and also clarify current trends, which are mainly driven by ECPSs, IoT, and ML.

The remaining of this survey is organized as follows. We present a brief discussion about models for ECPSs, in Section 2, and then cover preliminaries on bounded model checking, k-induction verification based on invariant inference, counterexample guided inductive synthesis, and system model incorporation, in Section 3. In Section 4, the main challenges regarding verification and synthesis of embedded systems are described, while Section 5 tackles those challenges that are (partially) open in current published research and later describes current achievements and future trends in verification and synthesis of ECPSs, and Section 6 addresses new applications, which take into account the new trends (*e.g.*, ML) mentioned here. Finally, we conclude and describe future work, in Section 7.

2 Automated Verification and Synthesis of Cyber-Physical Systems

Automated formal verification has been employed to ensure correctness and reliability of ECPSs, during the last decade, as well as provide controller synthesis [19–25] for ECPSs and HSs [26–32]. The reliability of an ECPS is related to its ability to maintain the continuity of correct execution of specific function. In particular, the reliability of an ECPSs depends on the correctness of their embedded software and the compatibility with hardware. The correctness is asserted with respect to a given specification, that may be a safety specification or a reachability specification [33]. A safety specification is related to a set of undesirable states that must be avoided by the system, and a reachability is related to a desired state, that must be achieved. In both cases, it is needed a model to compute the reachability of ECPS an than decide about the correctness and reliability.

A general model for a ECPS is defined as follows.

Definition 1. *An ECPS S is a tuple $S = (X, X_0, U, Y, r, v)$, where X is a set of reachable states, X_0 is a set of initial states, U is a set of inputs, Y is the set of outputs, r is a transition function such that $r : X \times U \to X$, and v is an output mapping function such that $v : X \times U \to Y$ [34–36].*

The synthesis of reliable ECPS is related to the synthesis of controllers that are able to handle the complex dynamic of these systems. Indeed, the so-called ECPS can be considered HS, since they present a mixture of continuous and discrete dynamics. Thus, the control synthesis for ECPS is based in the hybrid control system theory, that is still in development, since the control theory for continuous and discrete systems were developed without connexions [37]. The formal synthesis of ECPS includes but is not limited to the control synthesis problem, and must consider the formal verification of correctness and reliability.

Summarizing, the overall automated formal synthesis process for ECPSs and HSs involves three main steps: *modeling*, which is related to obtaining a sound mathematical representation able to capture all relevant aspects of continuous and discrete dynamics and their interaction, as well as formulating properties and invariants that describe a desirable behavior, *verification*, where a ECPS behavior is checked through the proposed mathematical structure and compliance with a desired behavior is verified, and, finally, controller *synthesis*, which produces a controller or finds invariant control laws that allow a ECPS to meet requirements and behave as desired, which is confirmed by a verification oracle.

2.1 A Model for Cyber-physical Systems

Models are largely employed in science and engineering areas, in order to analyze, predict, and modify the behavior of real-world systems. Specially, deterministic and stochastic models are the basis of many recent engineering advances. The former is uniquely determined by parameter values and previous states, while the latter presents inherent randomness which is described by probability distributions, i.e., some of these variable are random variables whose behavior are described by probability density functions [38]. Nonetheless, one can argue that such models are not enough to represent ECPSs, since interaction among different entities (software and hardware components) presents several events, which could not be deterministically or stochastically modeled, and they should be better represented by nondeterministic approaches [4]. Nondeterministic models is able to model different behaviors without providing a measure of probability, i.e., their variables are assumed to be different values but they are not random variables. Indeed, that is even more evident when unknown properties are addressed, such as human behavior.

On the one hand, early representations of hybrid systems, *i.e.*, systems that mix continuous and discrete dynamics, consisted in instantiating discrete components and discretized continuous dynamics by using discrete models and considering non-modeled effects as uncertainties or perturbations exogenous to them, which can even be negligible. On the other hand, that approach is not suitable for complex ECPSs, which present intense nondeterminism, where uncertainties play a central role and might invalidate the entire system behavior. Indeed, ECPSs have been modeled through different structures that present different degrees of nondeterminism and balance between continuous and discrete dynamics, such as timed automata [39], hybrid automata [40, 41], and stochastic hybrid automata [42–44].

Timed automata targets real-time systems and employs finite automata with clocks, *i.e.*, real variables that follow continuous trajectories (behaviors) with constant slope. Hybrid automata bridges digital processing and analog entities and consequently generalizes timed automata, by also employing finite automata with real variables, but whose trajectories follow more general dynamical laws [41]. Timed automata and hybrid automata are deterministic models for hybrid systems that built the basis of HS representations, which were subdivided into different classes of hybrid automata, depending on mapping functions (r and v), *e.g.*, linear hybrid automata [45], rectangular hybrid automata [41, 46], nonlinear hybrid automata [47, 48], integration graphs [49], and linear stochastic hybrid automata [42]. Recently, Rungger and Tabuada [34] and Tabuada *et al.* [50] proposed a general and wide model for ECPSs, which is based on hybrid automata, by providing the concept of robustness and important tools for robust stability analysis. In addition to HA and TA based models, there are a number of other ECPS modeling initiatives based on (exact or approximate) bisimulation [36], simulation [51] and interval analysis [52].

Another approach to ECPS modeling tackles quantized control systems and finite-word length (FWL) effects on the performance of those systems. Differently from bisimulation, quantized models are built from a countable subset of the (quantized) input space [53]. In particular, Keel and Bhattacharyya showed that even robust and optimal controllers may still present an undesired property [54, 55]: fragility or sensitivity to implementation aspects (*e.g.*, FWL representations of digital controllers). Therefore, various studies tried to connect sampled-data theory to hybrid-system models [56–59].

In addition to ECPS-dynamics modeling, it is often interesting to establish invariants on a ECPS's state space, which can be used for accelerating formal verification and synthesis processes regarding those systems. Such invariants might be synthesized, in order to point out state-space boundaries [60] and determine ECPS stability [61] or reachability [52, 62]. Finally, it is worth to mention that there are still some recent modelling approaches that are mixtures of stochastic and non-deterministic models for ECPS, e.g., stochastic nondeterministic automata [63], and piecewise-deterministic Markov chains [42].

2.2 Automated and Formal Verification of Cyber-Physical Systems

An ECPS verification problem is defined as follows:

Definition 2. *Given a class of ECPSs Ω and a set Φ of desired properties, a class of verification problems is called decidable if there is an algorithm that is able to decide if a finite number of steps $\forall \mathcal{O} \in \Omega$ satisfy every property $\mathcal{L} \in \Phi$ [35].*

Decidability is a key issue in ECPS verification, since there are uncountable states and complex dynamics to be considered, *i.e.*, hybrid automata and their variations (usually employed to model ECPSs) are infinite-state machines [41]. In particular, the safety verification of ECPS is undecidable, except in very specific cases as in systems that can be represented by timed automata and rectangular automata [64]. An alternative to ensure that ECPS trajectories will not pass through unsafe states is the use of barrier certificates [65, 66]. An example of barrier certificate is stability, which ensures the convergence of states to an equilibrium point and consequently that there exists a region from where any trajectory remains inside this region. Stability can be ensured based on Lyapunov theory (for a wide class of model abstractions) [23] or linear system stability theory for discrete simulations [67].

Generally, properties that must hold for ECPSs are related to safety/reachability and liveness, and the earliest studies on HS and ECPS proposed approaches based on computational tree logic and linear temporal logic [40], in order to specify properties and verify those systems. In the last decade, signal temporal logic [68, 69] was proposed and have been used since then for this purpose, thus providing more suitable operators for dynamic systems and signal processing. Therefore, model-checking procedures have been employed to verify those properties, even under undecidability conditions via approximations, *e.g.*, bounding the number of steps from initial states, which leads to bounded model checking (BMC) [70].

Along with CPS modeling and verification techniques purely based on HA, one can also highlight logic-based approaches [71] that may still use HA, but which are not based on a algebraic description of states and transitions. Some examples of logic-based verification are theorem proving techniques, *e.g.* deductive verification (i.e., interactive theorem proving) [72] and automatic theorem proving [73], and reduction-based techniques, among which BMC stands out [74].

Recently, some studies have addressed ECPS formal verification considering discretization, switching, and quantization effects. Dugirala and Viswanathan [75] verified embedded control software considering real time issues, *e.g.*, delayed responses and possible loss of real time deadlines. Anta *et al.* [76] presented a tool called Costan, which finds errors in mathematical-model implementations and verifies whether error is tolerated, considering quantization effects and fixed-point implementation, while focusing on quantization error effects over system stability. Similarly, Ismail *et al.* [77] preseted DSVerifier, a BMC tool for digital systems that employs SMT/SAT solvers as back-ends, in order to provide support for digital system design and verification considering FWL effects. In particular, DSVerifier checks if a designed digital controller and/or closed-loop ECPS presents the desired performance, when it is implemented with a given FWL format [67, 78].

Finally, it is worth mentioning that there are other alternatives for ECPS modeling and analysis, which are not based on verification, but instead on simulators [79–81]. Although such approaches generally fail in ensuring that some properties hold, they present good scalability, which is ideal for large scale and complex systems that are hardly modeled through automata- or logig-based approaches. Modelica [81] and Simulink/MATLAB [80] are important examples of widely used simulators for ECPSs. In order to achieve a balanced trade-off between ECPS-simulation scalability and correctness of verification methods, statistical model checking have been employed to formally verify ECPSs [26].

In addition, Shoukry *et al.* [82] tackled another important problem related to ECPSs: security. Indeed, the essence of ECPSs claims for sophisticated interfacing (*e.g.*, sensors and network connections),

which normally leads to security issues [83], given that wrong control actions can be generated through corrupted measurements. The mentioned authors proposed a methodology for estimating physical-system state, through formal methods, even if sensor inputs are contaminated with erroneous or intentionally corrupted data, in order to still be able to execute suitable control commands. Cyber-security is also a concern of Fiore *et al.*, who aim to ensure secure-state estimation of UAV systems under sparse attacks [84]. Indeed, many situations can be regarded as satisfiability problems and solved, for instance, with approaches based on Satisfiability Modulo Theories (SMT) [85–87], which indicate many links with other research areas and might lead to a unified approach regarding system verification, security enforcement, and interaction handling.

2.3 Control Synthesis of Cyber-physical Systems

Control synthesis for ECPSs consists in finding a controller or an invariant control law $\mathcal{C}$, such that a closed-loop system composed of $(\mathcal{C}, \Omega)$ satisfies every property $\mathcal{L} \in \Phi$.

An possible approach to synthesize reliable ECPSs is Counterexample Guided Inductive Synthesis (CEGIS) [88]. Its key idea is the definition (by a user) of a set of safe states, *i.e.*, the specification of Φ, as well as a generic template for a control system that must satisfy such a specification. Then, a CEGIS engine computes values for the unknown symbols of the chosen template, which hold Φ [89]. Abate *et al.* [8, 90] presented a method for synthesizing stable controllers that are suitable to continuous plants given as transfer functions [8] and state-space representations [90], which exploits bit-accurate verification of software implemented in digital microcontrollers [67, 77, 78], and the resulting systems must ensure non-fragile stability [8] and safety [90]. Ravanbakhsh and Sankaranarayanan [91, 92] also employed CEGIS based on the Lyapunov's function to ensure the robust reach-while-stay property (*i.e.*, a safety specification) [92] and non-zeno behavior [91]. CEGIS was also employed to synthesize model predictive controllers for a wide varieties of applications, *e.g.*, heating ventilation and air conditioning systems, autonomous vehicles control, and aircraft electric power system [43, 87].

CEGIS is an important, but not the unique, example of symbolic control synthesis for ECPSs. The main advantage of using symbolic synthesis techniques (*e.g.*, SMT- and SAT-based) is the possibility of encoding (via logic formulae) reachability/safety specifications on original concrete systems. Indeed, there are many studies [51, 93–102] on symbolic control synthesis for ECPS, available in the literature.

A common procedure for controller synthesis for ECPS is based on finite-state approximations of infinite-states systems, *i.e.*, the HA. Such an approach allows the fully automated synthesis of ECPS control systems, which ensure complex specifications realization; however, the main weakness of this approach is that it often produces complex controllers with high implementation cost. In order to avoid that problem, refined controllers have been proposed, which aim to provide some invariant properties for close-loop systems, *e.g.*, safety [102–104], reachability [100, 105], stability [95, 99], and safety/reachability properties, considering quantization effects [106].

3 SAT- and SMT-based Verification and Synthesis Techniques

Due to the advent of sophisticated SMT solvers built over efficient SAT solvers, an increasingly number of software verifiers as well as synthesizers have emerged in the literature, in order to verify design correctness and produce reliable implementations for ECPS, which was later reinforced by the introduction of knowledge regarding their behavior. Here, we describe some achievements in symbolic model checking of software, including BMC, *k*-induction, "property-based reachability" (or IC3), and Craig interpolation, which are currently used to the safety verification of ECPS. This section also overviews software synthesis techniques that are applicable to write correct-by-construction implementations of ECPS.

3.1 Bounded Model Checking (BMC)

Bounded Model Checking (BMC) based on SAT was originally proposed to verify hardware designs [74, 107]. Indeed, Biere *et al.* were able to successfully verify large digital circuits with approximately 9510 latches and 9499 inputs, leading to BMC formulae with 4×10^6 variables and 1.2×10^7 clauses to be checked by standard SAT solvers. BMC based on SMT [108], in turn, was originally proposed to deal with increasing software verification complexity [109]. In general, BMC techniques aim to check the violation of a given (safety) property at a given system depth, as shown in Figure 2.

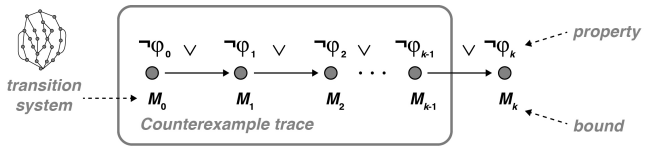

Fig. 2: Bounded Model Checking.

Indeed, given a transition system M, which is derived from the control-flow graph of a program, a property ϕ, which represents program correctness and/or a system's behavior, and an iteration bound k, which limits loop unrolling, data structures, and context-switches, BMC techniques thus unfold a system k times, in order to convert it into a verification condition ψ, which is expressed in propositional logic or in a decidable-fragment of first-order logic, such that ψ is *satisfiable* if and only if ϕ has a counterexample of depth less than or equal to k. The propositional problem associated with SAT-based BMC is formulated by constructing the following equation [107]:

$$\psi_k = I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \neg\phi_k \tag{1}$$

Here, ϕ_k represents a safety property ϕ in step k, I is the set of initial states of M, $R(s_i, s_{i+1})$ is the transition relation of M at time steps i and $i+1$. Hence, the equation $\bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})$ represents the set of all executions of M of length k and $\neg\phi_k$ means the condition that ϕ is violated in state k, which is reached by a bounded execution of M of length k. Finally, the resulting (bit-vector) equation is translated to conjunctive normal form in linear time and passed to a SAT solver for checking satisfiability. Eq. (1) can be used to check safety properties [110]. Liveness properties (*e.g.*, starvation, deadlock) that contain the Linear-time Temporal Logic (LTL) operator F are checked by encoding $\neg\phi_k$ in a loop within a bounded execution of length at most k, such that ϕ is violated on each state in the loop [111]. In that case, Eq. 1 can be rewritten as:

$$\psi_k = I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \left(\bigvee_{i=0}^{k} \neg\phi_i \right) \tag{2}$$

where ϕ_i is the propositional variable ϕ at time step i. Thus, this equation can be satisfied if and only if for some i ($i \leq k$) there exists a reachable state at time step i in which ϕ is violated.

One may notice that BMC analyzes only bounded program runs, but generates verification conditions (VCs) that reflect the exact path in which a statement is executed, the context in which a given function is called, and the bit-accurate representation of expressions. A verification condition is a logical formula (constructed from the bounded program and desired correctness properties) whose validity implies that the program's behaviour agrees with its specification [112]. Correctness properties in programs can be specified by the user via *assert* statements or automatically generated from a specification language [113]. If all of a bounded program's VCs are valid, then a program is in compliance with its specification up to the given bound.

As an example, consider a simple C program (slightly modified from [114]) with an exponential number of paths, as shown in Figure 3(a); the corresponding C program, in single static assignment (SSA) form [115], is shown in Figure 3(b). The SSA form

is an intermediate representation, which is used by compilers to facilitate optimizations and transformations of source code. The common property in SSA form is that every variable state has only one definition in the program text. This is achieved by introducing a fresh variable from the original name (*e.g.*, with a subscript), such that every assignment has a unique left-hand side, as shown in Figure 3(b).

```
#include <assert.h>
int x[N], a;
...
int main(void) {
a=N;
for(int i=0; i<N; i++)
if (x[i]>1)
a--;
assert(a<=N);
return 0;
}
```

(a)

```
a1 = N
a2 = (x[0] > 1) ? a1 - 1 : a1
a3 = (x[1] > 1) ? a2 - 1 : a2
a4 = (x[2] > 1) ? a3 - 1 : a3
...
an+1 = (x[N-1] > 1) ? an - 1 : an
```

(b)

Fig. 3: (a) A simple C program with a loop *for*. (b) The corresponding unwound C program of (a) converted into SSA form.

Apart from that, this program has an exponential number of paths, since each element of x can be either greater than one or less than or equal to one. Despite the large number of paths through that program, BMC unwinds it up to a bound k and translates it into a VC ψ, such that ψ is satisfiable if and only if the assertion $a <= N$ fails. One may also notice that BMC still encodes program stages with a size that grows linearly with N. More precisely, the program in Figure 3(a) is converted into ψ using a decidable fragment of first-order logic, as follows [112]:

$$\psi := \left[\begin{array}{l} a_1 = N \\ \wedge\ a_2 = ite\,(x[0] > 1, a_1 - 1, a_1) \\ \wedge\ a_3 = ite\,(x[1] > 1, a_2 - 1, a_2) \\ \wedge\ a_4 = ite\,(x[2] > 1, a_3 - 1, a_3) \\ \wedge \ldots \\ \wedge\ a_{N+1} = ite\,(x[N-1] > 1, a_N - 1, a_N) \\ \wedge \neg\,(a_{N+1} \leq N) \end{array} \right]. \quad (3)$$

The ternary operator $f\ ?\ t_1 : t_2$, shown in Figure 3(b), is converted into the conditional expression $ite(f, t_1, t_2)$ that takes as its first argument the Boolean formula f and, depending on its value, selects either the second (i.e., t_1) or the third argument (i.e., t_2). In order to verify that the assertion $a <= N$ holds, its negation is added to ψ and it is checked whether the entire formula is satisfiable, through an off-the-self SMT solver. Formula (3) can be represented simply as a Boolean logic circuit, which can be further transformed into a (equisatisfiable) conjunctive-normal-form formula over propositional variables, by Tseitin's transform [116] in linear time and by introducing at most a linear number of fresh variables; however, checking the validity of a first-order logic formula, in a given background theory, is an $\mathcal{NP}$-complete problem [117].

From the practical point of view, SAT- or SMT-based BMC procedures have been successfully applied to verify a large number of hardware and software systems, including digital circuits and single- and multi-threaded programs. Those BMC techniques were able to find subtle bugs in real digital and embedded software systems, as reported in the available literature [118–122]. Nonetheless, the main criticism with respect to BMC techniques relies on completeness, since they are able to prove system correctness only if an upper bound k is known, *i.e.*, a bound that unfolds all loops and recursive functions to their maximum possible depth.

Due to that limitation, BMC tools are typically susceptible to exhaustion of time or memory limits, when checking complex circuit-implementations or programs with loops, whose bounds are too large or cannot be statically determined. Indeed, such an issue has pushed researchers to overcome the depth limitation and propose extensions capable of proving global correctness. In particular, we can adopt two possible strategies to prove properties using BMC: *(i)* compute the *completeness threshold*, which can be smaller than or equal to the maximum number of loop-iterations occurring in the program or *(ii)* determine the high-level worst-case execution time (WCET), which also gives a bound on the maximum number of loop-iterations [74, 123, 124]. However, in practice, complex software systems involve large data-paths and complex expressions. Therefore, the verification conditions that arise from BMC of programs become harder to solve and require substantial amounts of memory to build. Finally, another strategy is to use a complementary technique, with the potential to fill the gap between the adopted depth and all subsequent ones, such as induction, which is explained below [125, 126].

3.2 Induction-based Verification of C Programs

One promising approach to achieve completeness, in BMC techniques, is to prove that an invariant (assertion) is k-inductive [125, 126]; however, the main challenge regarding such an approach relies on computing and strengthening inductive invariants from programs. In particular, loop invariants, which are computed from programs under verification, must be inductive (and not just invariant), in order to check the corresponding VCs, *i.e.*, induction cannot determine invariance of a non-inductive assertion [112].

For instance, consider the C-code fragment shown in Figure 4, where the star-notation indicates nondeterminism. Suppose that one wants to prove that $P : x > 0$ is invariant and, in order to do that, induction can be applied:

- It holds initially because

$$\underbrace{x = 2}_{\text{initial condition}} \implies \underbrace{x > 0}_{\text{P}}.$$

- Whenever P holds for k loop unwindings, it also holds for $k + 1$ steps, given that

$$\underbrace{x > 0}_{\text{P}} \wedge \underbrace{x = 2 \wedge x' = 2 * x - 1}_{\text{transition relation}} \implies \underbrace{x' > 0}_{\text{P'}}.$$

If we consider the IEEE floating-point standard (IEEE 754) [127, 128], then the invariant $x > 0$ holds initially and after each iteration and x tends to infinity after 128 iterations. Nonetheless, this invariant is not inductive, given that $x > 0$ before an initial iteration does not ensure that $x > 0$ after each iteration. In particular, if we initially assign $x = 0.9$, then $x < 0$ after the fourth iteration. As a consequence, even if invariant generation procedures successfully compute such assertions, which are indeed invariant, those must be inductive, so that k-induction verifiers can automatically prove correctness. In that specific example, an inductive invariant would be

$x > 1$, given that if $x > 1$ before the initial iteration, then also $x > 1$ after k iterations.

```
#include <assert.h>
int main(void) {
float x=2;
while (*) {
x = ((2*x) - 1);
}
return 0;
}
```

Fig. 4: Motivating example for inductive invariants.

There are several invariant-generation algorithms that discover linear and polynomial relations among integer and real variables, in order to provide loop invariants and also discover the memory "shape", in programming languages with pointers [129, 130]. The current literature also reports successful applications of k-induction based verification algorithms for hardware and software systems, using invariant generation and strengthening, mostly based on interval analysis [131].

Novel verification algorithms for proving correctness of (a large set of) C programs, by mathematical induction and in a completely automatic way (*i.e.*, users do not need to provide loop invariants), were recently proposed [131–135]. Additionally, k-induction based verification was also applied to ensure that (restricted) C programs (1) do not contain violations related to data races [136], considering the Cell BE processor, and (2) do respect time constraints, which are specified during system design phases [125]. Apart from that, k-induction is also a well-established technique in hardware verification, where it is easily applied, due to the monolithic transition relation present in such designs [125, 126, 137].

It is worth noticing that k-induction with invariants has the potential to be directly integrated into existing BMC approaches, given that the induction algorithm itself can be seen as an extension after k unwindings and it is possible to generate program invariants with other software modules, which are then translated and instrumented into an input program [135].

Nonetheless, there is little evidence, in the available literature, that model checking hardware and software systems through k-induction (and invariants) can be efficiently exploited in ECPS verification and synhesis. That happens due to the distinctive characteristics mentioned earlier, which influence ECPS development and also verification processes. Additionally, there is still a lack of studies for software verifiers to exploit the combination and configuration of different invariant generation and strengthening algorithms, including analysis to discover linear inequalities, polynomial equalities and inequalities, and invariants related to memory and variable aliasing [112].

3.3 Property-based Reachability (or IC3)

Bradley *et al.* introduced the "property-based reachability" (or IC3) procedure for the safety verification of systems [138, 139] and have shown that IC3 can scale on certain benchmarks, where k-induction fails to succeed. In particular, the success of IC3 over k-induction procedures is due to the ability of the former to guide the search for inductive instances with counterexamples for the inductiveness (CTIs) of a given property [140]. Consider again the C-code fragment shown in Figure 4, where $P : x > 0$ is an invariant.

$$\underbrace{x > 0}_{\text{P}} \wedge \underbrace{x = * \wedge x' = 2 * x - 1}_{\text{transition relation}} \Rightarrow \underbrace{x > 0}_{\text{P'}}.$$

In this specific example, a CTI returned by an SMT solver is $x = 0$. If this state is not eliminated, then the invariant P cannot be established. The generated inductive assertion should establish that the CTI $x = 0$ is unreachable. If no such inductive assertion exists, then other CTIs can be examined instead (*e.g.*, $0.1, 0.2, \ldots, 0.9$). As a result, the lemmas should be strong enough that consecutively revisiting a finite set of CTIs will eventually end up in an assertion, which is inductive relative to them, thus eliminating the CTI. In this example, the instance $x > 1$ ($a = 1, b = 0$) is inductive and eliminates all possible CTIs.

Jovanović *et al.* [141] presented a reformulation of IC3, by separating reachability checking from inductive reasoning. They further replace the regular induction algorithm by the k-induction one and show that it provides more concise invariants. Those authors implemented the mentioned algorithm in the SALLY model checker using Yices2, in order to perform a forward search, and MathSAT5, which executes a backward search. They showed that the new algorithm is able to solve a number of real-world benchmarks, at least as fast as other approaches.

3.4 Craig Interpolation

Another feasible alternative to prove properties in BMC is to compute Craig interpolants for inconsistent pairs (or more generally, sets) of formulae [142, 143]. This alternative approach exploits the SAT/SMT solvers' ability to produce refutations, *i.e.*, proofs regarding the nonexistence of counterexamples of depth less than or equal to k. Those proofs do not ensure whether a given property holds in the model, but they contain information about reachable states of a model.

Definition 3. *Given a pair of formulae (A, B), such that $A \wedge B$ is inconsistent, and a proof by resolution for (A, B), an interpolant for (A, B) is a formula F with the following properties [142, 143]:*

- $A \Rightarrow F$;
- $F \wedge B$ *is unsatisfiable;*
- F *expressed only over the common variables (nonlogical symbols) of A and B.*

As an example, consider $A = (x_1 \wedge x_2)$ and $B = (\neg x_2 \wedge x_3)$. Given that $(x_1 \wedge x_2)$ must imply F (or simply that $\neg x_1 \vee \neg x_2 \vee F$ hold) and $F \wedge \neg x_2 \wedge x_3$ must be unsatisfiable, one possible interpolant for the given pair of formulae (A, B) is $F = x_2$, since x_2 is a common part of both A and B.

The use of interpolants allows us to define a complete method for finite-state reachability analysis based on SAT and SMT solvers. In order to show how BMC and interpolation can be combined, we refer to Section 3, where we define Eq. (1) and the terms I, R, and ϕ. Now, suppose that $Q = I$ and Eq. (1) is partitioned, so that the set of initial states I and the first instance of the transition relation R are in set A, while the remaining instances of R and the property ϕ are in set B, as shown in Figure 5 (note that k is unknown).

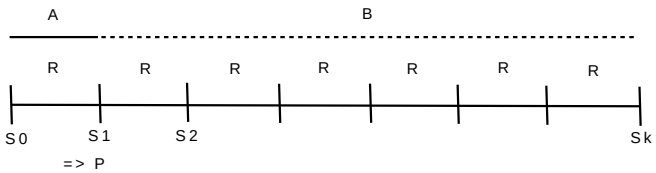

Fig. 5: Computing image by interpolation [142].

Suppose that we use an SMT solver to prove that the $A \wedge B$ is unsatisfiable, *i.e.*, we use an SMT solver to conclude that there is no satisfying assignment to $A \wedge B$.* The internal steps performed by SMT solvers for reaching this conclusion can be used to construct a proof of unsatisfiability Π, from which we can derive an interpolant F for the pair of formulae (A, B), i.e., $F = interpolant\,(\Pi, A, B)$.

Note that if at any stage we can satisfy the property ϕ within k steps from the initial state, then we have found a counterexample.

According to Definition 3, A must imply F and since we defined A to be the set of initial states and the first instance of R (i.e., from Figure 5, $A = s_0 \wedge s_1$), it follows that F is *true* in every state reachable from the initial state in one step. In other words, we can say that F is an over-approximation of the forward image of I [142, 143]. Also according to Definition 3, the formula $F \wedge B$ must be unsatisfiable (from Figure 5, $B = s_2 \wedge s_3 \wedge \ldots \wedge s_k$), which means that there is no state satisfying F that can reach a final state s_k. After computing F, we then check whether F implies Q. If F implies Q, then no reachable state can satisfy the property ϕ and we can thus conclude that the property holds; however, since F is an approximation, we can falsely conclude that the final state is reachable. In this case, we update $Q = F \vee Q$ and $A = F \wedge R_0$, increase the value of $k + 1$ and check whether $A \wedge B$ is unsatisfiable. If $A \wedge B$ is satisfiable, we have found a valid counter-example (i.e., a path from the initial state to the final one); otherwise, we compute the interpolant $F = interpolant\,(\Pi, A, B)$ again and check whether F implies Q. This procedure is stopped when we have found a valid counterexample or proved that the final state is not reachable (i.e., the property holds). The details of the algorithm and further information about the use of interpolants in model checking can be found in [142, 143].

3.5 Program Synthesis via Counter-Example Guided Inductive Synthesis (CEGIS)

The basic idea of program synthesis is to automatically construct a program P that satisfies a correctness specification σ. In particular, program synthesis is automatically performed by engines that use a correctness specification σ, as starting point, and then incrementally produce a sequence of candidate solutions that satisfy σ [8, 144]. As a result, a given candidate program p is iteratively refined, in order to match the specification σ more closely. Counter-Example Guided Inductive Synthesis (CEGIS) represents one of the most popular approaches to program synthesis that are currently used in practice [144]. Figure 6 shows the basic architecture of the CEGIS approach, which has close connections to algorithmic debugging using counterexamples and abstraction refinement [145].

The correctness specification σ provided to the program synthesizer is of the form $\exists\vec{F}.\forall\vec{x}.\sigma(\vec{x},\vec{F})$, where $\vec{F}$ ranges over functions, $\vec{x}$ ranges over ground terms, and σ is a quantifier-free formula typically supported by SMT solvers. The ground terms are interpreted over some finite domain $\mathcal{D}$, where $\mathcal{D}$ can be encoded using the SMT's bit-vectors part. The phases SYNTHESIZE and VERIFY interact via a finite set of test vectors INPUTS that is incrementally updated. Given the correctness specification σ, the SYNTHESIZE procedure tries to find an existential witness $\vec{F}$ satisfying the specification $\sigma(\vec{x},\vec{F})$, for all $\vec{x}$ in INPUTS (as opposed to all $\vec{x} \in \mathcal{D}$). If SYNTHESIZE succeeds in finding a witness $\vec{F}$, this witness is a candidate solution to the full synthesis formula, which is passed to the phase VERIFY, in order to check whether it is a proper solution (*i.e.*, $\vec{F}$ satisfies the specification $\sigma(\vec{x},\vec{F})$ for all $\vec{x} \in \mathcal{D}$). If this is the case, then the algorithm terminates; otherwise, additional information is provided to the phase SYNTHESIZE, in the form of a new counterexample that is added to the INPUTS set and the loop iterates again.

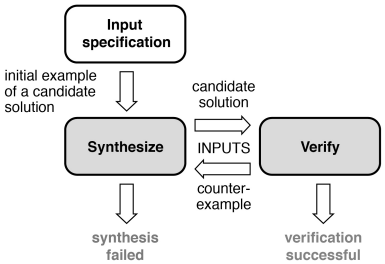

Fig. 6: CounterExample Guided Inductive Synthesis (CEGIS).

One may notice that each iteration of the CEGIS loop adds a new input to the finite set INPUTS that is used for synthesis. Given that the full set of inputs $\mathcal{D}$ is finite, this means that the refinement loop can only iterate over a finite number of times; however, $\mathcal{D}$ can represent a large number of elements for the finite set INPUTS. In order to avoid exploring all possible values, machine learning techniques can be used in the phase SYNTHESIZE, with the goal of learning from experience (input-output samples), *i.e.*, learning from counterexamples provided by a verification oracle [145].

Nowadays, program synthesis engines that implement the CEGIS approach [146] can automatically produce solutions for a large variety of specifications, due to the combination of automated testing, genetic algorithms, and SMT-based automated reasoning [147].

3.5.1 CEGIS for Embedded & CPS:

A typical synthesizer for Embedded & CPS iterates between two main phases, an inductive synthesis phase, that is, SYNTHESISE, and a validation one, that is, VERIFY [148]. The two phases interact via a finite set of test vectors, which is updated incrementally. Given a specification of a ECPS (*e.g.*, stability and safety), the inductive synthesis procedure tries to find a candidate solution satisfying that specification, for the given set of test inputs. If the synthesis phase succeeds in finding a witness, then the latter is a candidate solution for the full synthesis formula. This candidate is passed to the validation phase, which checks whether it is a proper solution (*i.e.*, it satisfies the specification for all possible inputs). If this is the case, then the algorithm terminates; otherwise, additional information is provided to the inductive synthesis phase, in the form of a new counterexample, C-ex, which is added to the set of test inputs, and the loop iterates again.

One possible architecture for a synthesiser is presented in Fig. 7, considering the stability and safety specification. It starts by devising a digital controller that stabilizes the physical model, while remaining safe for a pre-selected time horizon (k) and a single initial state. Then, it employs a multi-staged verification process as follows: (i) The first verification stage (SAFETY) checks if the candidate solution, which we synthesized to be safe for at least one initial state, is safe for *all* possible initial states, *i.e.*, it does not reach an unsafe state within k steps. (ii) The second verification stage (PRECISION) restores soundness with respect to plant precision, by using interval arithmetic [149] to validate the operations performed by the previous stage. (iii) The third verification stage (COMPLETE) checks if the current k is large enough to ensure safety for any $k'>k$. Here, we compute the completeness threshold $\overline{k}$ for the current candidate controller K [90], *i.e.*, the number of iterations required to sufficiently unwind the closed-loop state-space model, such that the boundaries are not violated for any larger number of iterations, and check if $k \geq \overline{k}$.

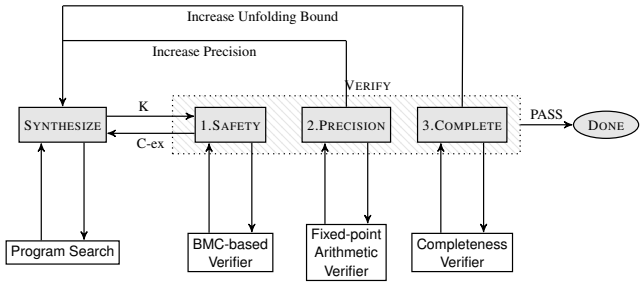

Fig. 7: CEGIS with multi-staged verification for ECPS.

3.6 Incorporating System Models to Automated Verification and Synthesis Procedures

Currently, SMT-based BMC approaches check code properties in real programs, which basically address programing-language issues and general correctness, without taking into account target applications or system behavior. Such a statement is important, since, as already mentioned, many system features are being moved to software domain, which then requires schemes that do not only check if source code is correctly written, but also if it will properly respond in

real environments or under external problems or corrupted data. For instance, the anti-lock braking system software of a vehicle model can be bug free, but it may not work correctly if a sensor is damaged or even intentionally tampered [82].

Indeed, research in software verification is now incorporating such considerations, during checking processes, and some schemes already use knowledge about the system to be verified and the underlying hardware. Recently, a verification tool for digital systems was proposed, which is called digital system verifier (DSVerifier) [77] and is able to aid engineers to check overflow, limit cycle, output error, timing, stability, and minimum phase, considering FWL effects. Additionally, DSVerifier checks closed-loop systems with uncertain models considering FWL effects, which are typically represented as hybrid systems, *i.e.*, the controller is digital but the controlled agent (plant) is a physical and continuous system [67]. That ultimately leads to the use of analog-to-digital (A/D) converters, which are one of the most important aspects to be considered, given that data loss (quantization) is inevitable. As a consequence, verification procedures have to consider the interaction between a continuous plant and a digital (and sampled) controller with FWL effects, which can be connected using different control system configurations. Additionally, the latter may present a different behavior, regarding what was specified in analog domain, due to the inherent discret-time operation.

In summary, DSVerifier is a useful test tool, which takes into account different representations (*e.g.*, transfer-function and state-space), realization forms (*e.g.*, direct, delta, and transposed forms), and other implementation restrictions, in order to explore the design space. Finally, if a system's requirements are not met with a given configuration, an analysis of the provided error report may suggest another setup, which can even be incorporated into a given system development procedure, by providing both system verification and model refinement.

In this respect, DSVerifier has been extended to automatically synthesize digital stabilizing controllers for continuous plants represented as transfer functions or state-space equations [8, 90]. In particular, a CEGIS-based approach, implemented in a tool called digital system synthesizer (DSSynth), uses inductive synthesis in conjunction with the DSVerifier's algorithm for verifying robust closed-loop stability that addresses plant variations as interval sets, as well as FWL uncertainties in digital controllers. DSSynth is able to successfully synthesize stable digital controllers for a set of intricate plant models, taken from the control literature, within minutes. Nonetheless, system correction is of paramount importance and, even with such a synthesizing tool, a final verification procedure should be executed, with the goal of ensuring that the generated solution still complies with the provided specs and other issues, which may have not been initial tackled and have the potential to compromise system reliability, are not present.

Apart from the control system domain, Scratch is another example software model-checking tool, which uses knowledge about the system to be verified and the underlying hardware, with the goal of detecting races related to direct memory access (DMA), in the Cell BE processor [136]. That tool also uses BMC, in order to detect DMA races, and BMC with k-induction, which aims to prove the absence of races. If support to other DMA operations were added, Scratch could be adapted to different architectures, *i.e.*, the same techniques would be employed, but with a different system behavior/knowledge.

One may also notice that such a paradigm, *i.e.*, verification and synthesis based on an expected system behavior, is not restricted to digital filters and controllers or DMA races, but it can also be applied to possibly any real system, as long as the desired behavior can be expressed as properties in BMC frameworks. For instance, regarding self-driving cars, an important property could be the detection of pedestrians, animals, bicycles, or any obstacle present on urban and rural roads. Indeed, bicycles are considered one of the most difficult problems, due to the myriad of possible shape and colors [150]. In addition, the mentioned problem is closely related to machine learning, from which refinement strategies can be devised, incorporated to BMC approaches [151], and directly applied to ECPS verification.

In summary, current approaches already tackle ECPS characteristics (*e.g.*, interaction with real world entities and digital processing), which are included in verification and synthesis methodologies. Finally, the final complement may come from ML techniques, which have the potential to fill gaps (*e.g.*, lack of complete models including nondeterminism), bridge tools and models (*e.g.*, analog and digital domains in ECPSs and verification schemes), and provide model and methodology evolution.

4 Verification and Synthesis Challenges for Embedded & Cyber-physical Systems

Generally, state-of-the-art verification methodologies for embedded systems (including ECPSs) generate test vectors (with constraints) and use some assertion-based verification and high-level processor models, during simulation [152, 153], as shown in Figure 8.

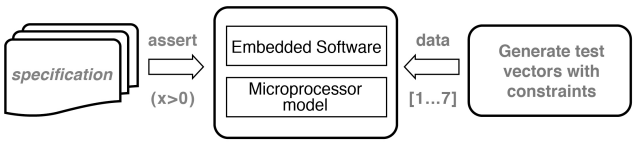

Fig. 8: Verification methodologies for embedded systems.

In particular, the main challenges regarding verification of embedded systems lie on improving coverage, where more system functions are verified, reducing verification time, *i.e.*, pruning state-space exploration during verification, providing completeness, *i.e.*, if all possible states can be reached and evaluated, and incorporating system models, which allows 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. Verification of code pieces or modules that rely on larger structures;
5. Legacy designs;
6. Support to different programming languages, frameworks, and interfaces;
7. Correct code instrumentation;
8. Handling of non-linear and non-convex optimization problems;
9. Incorporation of new and adapted checks, due to system or environment change;
10. Provision of a form of system evolution, in such a way that a version more adapted to a given scenario is obtained.

Indeed, the first two aspects are of extreme relevance in microgrids 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 restrictions.

The fourth aspect deals with code that relies on existing infrastructures and must be compliant with those, such as Linux kernel modules. The fifth aspect, in turn, is inherent to a large number of embedded systems from telecommunications, control systems, and medical devices, including ECPSs. In particular, software developed for those types of embedded system has been extensively tested and verified, and also optimized for efficiency over years of development. Therefore, when a new product is derived from a given platform, a lot of legacy code is usually reused for improving development time and code quality.

The sixth aspect is related to evolving 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 seventh aspect highlights a subjective task in verification: where one must insert an evaluation point or assertion, in order to correctly address a property. The eighth one is related to the widespread use of embedded systems in autonomous-vehicle navigation systems [154], which demands optimization solving during their execution for a wide range of functions, including non-linear and non-convex problems using fixed- and floating-point arithmetic.

The ninth aspect is a consequence of changes in environment or different applications and scenarios, which may present new problems to embedded systems that did not exist or were not tackled during their development. Such a challenge is also closely related to ML techniques, given that they may be integrated into commercial systems and have the potential to devise new and refined control strategies, which must then be identified, checked, and validated. For instance, a diagnostic system may employ techniques that make it able to learn new tumor patterns and them warn their presence on patients' bodies, but a wrong updated model may compromise earlier traditional assessments and erroneously perform new ones, which should be verified and validated. Finally, the latter also raises another question: an embedded system must be checked only when it is developed/created or also during its life cycle, in order to ensure that improved models and actions are sound and reliable? That is an issue that will probably concern researchers, when ML becomes mature and widely spread in software verification.

The tenth aspect is related to the current development and test strategies adopted for embedded systems, which aim at simply providing robust structures, instead of antifragile ones [155]. In summary, it is not feasible to account for every error or faulty scenario and then it would be more beneficial to provide a module, architecture, or methodology capable of thriving on problems and getting stronger. Such an aspect shares some similarity with the ninth one and may also be achieved through ML techniques. Indeed, this kind of scheme consists in a completely different approach, which would profoundly change the way we design embedded systems.

Those ten challenges place additional difficulties for developing reliable synthesizers for embedded systems, especially for cyber-physical systems and micro-grids, where controlled objects (*e.g.*, physical plants) typically exhibit continuous behavior (which may eventually change), whereas controllers (usually implemented by real-time computer systems) operate in discrete time and over a quantized domain. In particular, synthesizers for those systems need to consider the effects of 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 (cf. intelligent product in Figure 1). Additionally, finite-precision arithmetic and their related rounding errors need to be considered, when correct-by-construction code is generated for ECPS.

5 Current Achievements and Future Trends

In the preceding sections, we identified and discussed many aspects regarding ECPSs, which included design, synthesis, behavior, verification, and bottlenecks. Nonetheless, from the point of view of computer-aided verification and synthesis for ECPS, it is possible to highlight seven major problems, which are tackled here and can be regarded as (partially) open in current published research.

(RP1) provision of suitable encoding into SMT [108], which may extend background theories typically supported by SMT solvers, with the goal of reasoning accurately and effectively about realistic ECPS.

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

(RP3) proof of correctness and timeliness of ECPS, by taking into account stringent constraints imposed by hardware platforms.

(RP4) incorporation knowledge about system purpose and associated features, which aims to detect system-level and behavior failures.

(RP5) provision of 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) development of automated synthesis approaches that are algorithmically and numerically sound, in order to handle (control) software that is tightly coupled with physical environments, by considering uncertain models and FWL effects.

(RP7) provision of a unified framework, which is able to tackle high and low level properties, as well as system behavior, with the potential to lead to another degree of completeness, in such a way that every implementation aspect of ECPS is addressed and evaluated.

Due to the fact that the mentioned research problems are the some of the main issues found in the technology area addressed in this article, many works tackled them and other researchers still perform experiments and suggest new methodologies and schemes for mitigating or solving their consequences. As a result, we can cite the following contributions toward them, which already provide some answers or at least present promissing approaches.

(RP1) Cordeiro *et al.* proposed the first SMT-based BMC for full C programs, called Efficient SMT-Based Context-Bounded Model Checker (ESBMC) [122], which was later extended to support C++98 programs [156], CUDA programs [18, 157], and Qt-based consumer electronics applications [158]. 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) [120, 122]. Other SMT-based BMC approaches have also been proposed and implemented [119], but the coverage and verification time of all existing ones are still limited to specific program classes, especially for those that contain intensive floating-point arithmetic and dynamic memory allocation [159, 160]. One possible research direction is to bridge the gap between BMC tools and SMT solvers, propose background theories, and develop more efficient decision procedures, in order to handle specific program classes. In this particular research direction, the European Project SC^2 (Satisfiability Checking and Symbolic Computation: uniting two communities to solve real problems) aims to further extend background theories supported by SMT solvers, in order to solve problems related to security and safety of computer systems (or large mathematical problems), through the development of radically improved software tools [161].

(RP2) The SMT-based BMC approach proposed by Cordeiro *et al.* was further developed to verify correct lock acquisition ordering and absence of deadlocks, data races, and atomicity violations in multi-threaded software based on POSIX and CUDA libraries [120, 157], considering monotonic partial-order reduction [162] and state-hashing techniques, in order to prune state-space exploration [163]. Recent advances for verifying multi-threaded C programs have been proposed to speed up verification times, which significantly prune state-space exploration [164, 165], and also extensions to verify the Debian GNU/Linux distribution, based on a context- and thread-sensitive abstract interpretation framework [166], were made available; however, the set of concurrent-program classes (*e.g.*, CUDA, OpenCL, and MPI) that can be verified is still very limited. One possible research direction is to further extend BMC of multi-threaded C programs via Lazy Sequentialization [164], in order to analyze unsatisfiability cores [167], with the goal of removing redundant behaviour or analyzing interpolants [168] and then prove non-interference of context switches.

(RP3) Novel approaches to model check embedded software using k-induction and invariants were proposed and evaluated in literature, which demonstrate its effectiveness in some real-life embedded-system applications [132, 133, 135, 136]; however, the main challenge remains still open, *i.e.*, to compute and strengthen loop invariants for proving program correctness and timeliness in a more efficient and effective way, in order to be competitive with other model-checking approaches. In particular, invariant-generation algorithms have substantially evolved over the last years, with the goal of discovering inductive invariants of programs [129, 130] or continuously refine them during verification [131]; however, 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 during verification, in order to ensure system robustness w.r.t. implementation aspects. Finally, optimal configurations for invariant generators are still of paramount importance, given that applications with code based on different aspects may benefit from different setups.

(RP4) State-of-the-art SMT-based context-BMC approaches were extended to verify overflow, limit cycle, time constraints, stability, and minimum phase, in digital systems. Indeed, digital filters and controllers [77, 169, 170] were tackled, in order to specify system-level properties of those systems, using linear-time temporal logic [111]. In particular, a specific UAV application was tackled, with the goal of verifying its attitude controllers [78, 171]. More recently, other implementation aspects were added to the existing verification framework, including magnitude, phase, poles, and zeros, which provide deeper analysis regarding digital systems, by tackling frequency-domain parameters, permissible deviation, and associated natural response [172]. In general, however, there is still a lack of studies to verify system-level properties related to ECPS; emphasis should be given to micro-grids [2], 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 ECPS engineers [173].

(RP5) Although ESBMC [122] was extended to support C/C++ and some variants, new application interfaces and programming languages are often developed, which require suitable software verification tools. 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 Sousa *et al.* [174], which employed operational models for checking Qt-based programs from consumer electronics. In summary, the BMC core (in that case, ESBMC) is not changed (it is still C/C++ code), but instead an operational model, which implements the behavior and features of Qt libraries, is used to provide the new code structure to be checked. This approach could adopt the clang compiler [175], which is already widely used in industry [176], in order to produce an Abstract Syntax Tree (AST) for a given programming language (*e.g.*, C/C++/ObjectiveC/ObjectiveC++). This AST can then be converted either to an intermediate representation (IR) of the underlying verifier or to LLVM bitcode that is already produced by clang. Obviously, a completely new language may present new structures and resources; however, if different inputs are parsed and converted into a form that preserves every aspect of their structure, that can then be analyzed and verified in an unified way. Such research problem is closely related to the first one **(RP1)** and has the potential to devise a new paradigm in ECPS verification.

(RP6) State-of-the-art synthesis approaches for embedded (control) systems typically disregard the platform in which the embedded system software operates and restrict themselves to generate code that do not take into account FWL effects. The synthesized system, however, 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 [177]. 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 [8, 90]. Nonetheless, there is still little evidence whether that approach can scale for larger systems modeled by other types of representation (*e.g.*, Multiple-Input Multiple-Output). In addition to that, another research direction for synthesizers is to generate code for UAV trajectory and mission planning by taking into account system's dynamics and nonholonomic constraints [85]. As a consequence, verifiers and synthesizers need to handle a wide range of functions, including non-linear and non-convex optimization problems based on both fixed- and floating-point arithmetic.

(RP7) Embedded systems are typically implemented in low- and medium-level programming languages (*e.g.*, ANSI-C/C++/Java). Given the availability of different programming languages for writing code for ECPS, the provision of a unified framework able to tackle low- and high-level properties, as well as system behavior, is needed to achieve code coverage in real applications. The clang compiler [175] seems to offer a simple and strong approach to lead software verification to another degree of completeness so that every implementation aspect of an ECPS can be addressed and evaluated. In particular, by using the clang compiler as front-end for a software verification tool, we avoid the need for maintaining a proprietary front-end (a real challenge nowadays, given that the C and C++ standards are rapidly evolving), and then focus on the main objective: formal program verification in ECPS. Here, we provide only operation models to tackle low- and high-level system properties via an abstract representation of the associated libraries, which conservatively approximates their semantics. The clang compiler offers an industrial quality analyzer and an application programming interface to access and traverse its internal AST, which can then be used by software verifiers to generate their IR or alternatively analyze the LLVM bitcode produced by clang.

Lastly, yet importantly, BMC tools like CBMC [118], ESBMC [178, 179], SMACK [180], and LLBMC [119] represent the most prominent approaches for verifying C programs, as observed in the Intl. Competition on Software Verification [159, 160, 181], where verifiable (correctness and violation) witnesses are of extreme importance for evaluating software verifiers [182, 183]. The increasing number of verification tasks being added every year to this competition allows developers to further improve their verifiers by implementing new types of abstraction and simplification techniques. As a consequence, the continuous development and improvement of verifiers can leverage more efficient synthesizers and further increase the reliability of verified ECPS.

6 New Applications: Beyond Code Correctness

As already mentioned in section 2, problems that are not directly related to code correctness may be tackled with formal methods, as long as they are conveniently modeled. Araújo *et al.* [85] employed SMT for handling non-convex optimization problems, with the goal of ensuring optimal solutions. Indeed, it is possible to recursively re-constrains a model checking procedure, by using the current counterexample as a seed for the next run, until a global minimum is achieved. Nonetheless, elapsed execution times are still a bottleneck for some applications, such as changes in UAV trajectories, but that can be improved, through the use of simplification techniques and problem constraints.

In the same fashion, other problems may benefit from the recursive-refinement approach provided by SMT-based verification schemes. For instance, de Jesus *et. al* [184] developed a simplified methodology for antenna alignment that makes use of many modules already available in satellite TV receivers, along with a suitable structure for performing position correction. In summary, a reference signal is evaluated, until an (approximately) error-free reception is acquired, which can be measured through a simple transport stream (TS) error-indication interface and further refined with other parameters, such as carrier-to-noise ratio and signal level. This way, by using an SMT-based verification method, one may map the TS error indication and other performance figures as properties and then recursively perform realignment procedures, until a counterexample is not output. It is worth noticing that associated timing requirements are more relaxed than what was mentioned in the work developed by Araújo *et al.* [85], which can be seen as an advantage for such an implementation.

Regarding behavior correctness, many other applications can be tackled, such as the ones related to automatic classification. Amoedo *et al.* [185] proposed a modulation classification methodology, with the goal of spectrum sensing for opportunistic transmissions. Besides a correct software implementation, *i.e.*, correct source code, which is already covered by formal methods already available in literature [118, 122, 170], one may argue if the proposed methodology is able to recognize modulations under some reception condition, such as severe analog interference. This way, it would be interesting to develop a verification module for transmission and reception software, which could be capable of creating inputs related to sampled radio frequency signals, under a variety of conditions, and then unifying source code and behavior verification. In addition, that kind of

implementation usually employs digital filters with specific parameters, which could also be addressed. Indeed, communication aspects of ECPSs could be tackled, as properties in system verification procedures.

ML techniques are already being used in the software verification area [186–189]. For instance, Hamel [186] proposed to induce a theory, through ML, that fits a test set for a program, in such a way that it implements the respective test set. Phuc [187], in turn, applied decision tree C4.5 and multilayer perceptron to annotated programs (metrics related to structure and semantics) and tackled the generation of test sets satisfying desired constraints. Bridge, Holden, and Paulson [] studied the automation of heuristic selection based on features, regarding first-order logic theorem proving, and Hutter *et. al* [189] showed that ML can be used for improving SAT solvers regarding huge verification tasks, when applied to parameter tuning [189]. Seshia *et al.* [190], in turn, showed that control syntheses based on ML is also a trend, since it has been applied to different applications and model classes [191, 192]. Finally, ML can also be used for invariant computation and strengthening, which is a major bottleneck in k-induction verification based on invariant algorithms. As a consequence of what was presented, it seems worthwhile to pursue integration of ML into BMC-based checkers, given that it has the potential to tackle many open problems or at least enhance existing solutions.

In addition, another aspect closely related to ML techniques, as mentioned in Section 4, is system fragility. Currently, most design strategies try to produce robust systems, which are able to deal with a number of errors that happen when they are operating. Nonetheless, that is effective only if problems foreseen during design phases happen; otherwise, operation may be severely compromised and even cause loss of lives.

Indeed, depending on the employed point of view, fragility and robustness may be regarded as the same property. A system that is more robust is less fragile, but it is still prone to problems, which depends on operating scenarios, use cases, and uncertainties. Keel and Bhatactacharyya [55] showed that robust and optimal control system can still be fragile with respect to the implementation issues. In summary, if the chosen model does not tackle every possible problem or error that might occur in a given environment, system-failure events are still possible, which is true even for the Ariane 5's failure [155].

As a result, it may be more interesting to develop methodologies that are not simply robust, regarding predetermined conditions, but instead thrive on problems and grow stronger (as the human body, which gets stronger when exposed to germs), which leads to antifragile systems. In engineering, we are only beginning to develop solutions that can be classified as antifragile, that can be achieved via analytical or physical redundancy to achieve performance enhancement [193]. For example, multiple-input/multiple-output (MIMO) systems, which use multiple signal copies together with multipath signal copies would initially result in a destructive effect; however, with the use of a methodology that takes advantage on them, the net result is performance improvement.

Systems that could greatly benefit from being antifragile are the cyber-physical ones, which interact with other external elements and use their internal representations to interpret physical properties. As faults occur, which may be treated with some sort of system elasticity, and external environments are normally dynamic, given that it naturally changes with time, variations in operating conditions happen and should be tolerated, in order to keep behavior quality [155]. Besides, when discussing the fundamentals of antifragility, it is difficult not to tackle machine learning [155], which has the potential to provide some wisdom regarding correct behavior and internal representations, in order to keep a system working in a volatile environment.

For instance, we could think of a system including an antifragility module composed by two elements: a machine learning engine, capable of capturing current behaviors and identifying different trends, and a SMT-based verification method, capable of recursively performing retuning procedures, which could test new parameters estimated from the data provided by the machine learning module and then test new behaviors based on them, through counterexamples. In addition, there could also be a measurement module, which would also judge some metrics regarding the new behavior (e.g., signal-to-noise-error). As a consequence, we would be able to design systems with the potential of being antifragile, given that they would learn from new challenges and then modify its behavior model, in order to account for new errors and threats and even benefit from them.

7 Conclusions

This paper presented the main challenges related to the verification of design correctness, in ECPS, and also raised some important side considerations about synthesis. In particular, it emphasizes that stringent constraints imposed by the underlying hardware (*e.g.*, real-time, memory allocation, interrupts, and concurrency), along with system behavior models, must be considered during verification and synthesis. Additionally, there is little evidence that model checking ECPS using k-induction (and invariants), which extends BMC-based approaches from falsification to verification, can be applied to formally verify correctness and timeliness of ECPS.

Given that software complexity has significantly increased in ECPS, there are still some (recent) advances to stress and exhaustively cover 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. 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. In addition, it seems interesting to provide behavioral models when new application interfaces or programming language features are used, in order to extend capabilities of current verification tools, without changing the core BMC module. Finally, there are some bottlenecks in software verification that are possible to be handled with machine learning, given some initial studies carried out in the current literature, which may represent another step towards a unified and complete approach that combines code verification, behavior correctness, environment adaptation, and system evolution.

As future work, the main goal of this research is to extend BMC as a design, verification, and synthesis tool for achieving correct-by-construction ECPS implementations. Special attention will be given to 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. Additionally, ML techniques will be incorporated to BMC frameworks, in order to provide flexibility and specific-problem tuning, or even allow system adaptation and behavior tuning.

8 References

1 Kopetz, H.: 'Real-Time Systems: Design Principles for Distributed Embedded Applications'. Real-Time Systems Series. (Springer US, 2011)

2 Xu, X., Jia, H., Wang, D., Yu, D.C., Chiang, H.D.: 'Hierarchical energy management system for multi-source multi-product microgrids', *Renewable Energy*, 2015, **78**, pp. 621 – 630

3 Lee, E.A. 'Cyber physical systems: Design challenges'. In: 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing. (, 2008. pp. 363–369

4 Lee, E.A.: 'The past, present and future of cyber-physical systems: A focus on models', *Sensors*, 2015, **3**, (15), pp. 4837–4869

5 Groza, A., Letia, I.A., Goron, A., Zaporojan, S. 'A formal approach for identifying assurance deficits in unmanned aerial vehicle software'. In: Progress in Systems Engineering. (Springer, 2015. pp. 233–239

6 Cordeiro, L., Fischer, B., Chen, H., Marques.Silva, J. 'Semiformal verification of embedded software in medical devices considering stringent hardware constraints'. In: International Conference on Embedded Software and Systems. (, 2009. pp. 396–403

7 Munir, S., Stankovic, J.A., Liang, C.J.M., Lin, S. 'Cyber physical system challenges for human-in-the-loop control'. In: 8th International Workshop on Feedback Computing. (San Jose, CA: USENIX, 2013.

8 Abate, A., Bessa, I., Cattaruzza, D., Lucas, C., David, C., Kesseli, P., et al. 'Sound and automated synthesis of digital stabilizing controllers for continuous plants'. In: 20th ACM International Conference on Hybrid Systems: Computation and Control (HSCC). (, 2017. pp. 197–206

9 Witkowski, T., Blanc, N., Kroening, D., Weissenbacher, G. 'Model checking concurrent linux device drivers'. In: 22nd IEEE/ACM International Conference on Automated Software Engineering. ASE '07. (New York, NY, USA: ACM, 2007. pp. 501–504

10 Virtanen, S., Virtanen, S.: 'Advancing Embedded Systems and Real-Time Communications with Emerging Technologies'. 1st ed. (Hershey, PA, USA: IGI Global, 2014)
11 Huang, X., Kwiatkowska, M., Wang, S., Wu, M. 'Safety verification of deep neural networks'. In: CAV. vol. 10426 of *LNCS*. (, 2017. pp. 3–29
12 Katz, G., Barrett, C.W., Dill, D.L., Julian, K., Kochenderfer, M.J. 'Reluplex: An efficient SMT solver for verifying deep neural networks'. In: CAV. vol. 10426 of *LNCS*. (, 2017. pp. 97–117
13 Buccafurri, F., Eiter, T., Gottlob, G., Leone, N.: 'Enhancing model checking in verification by ai techniques', *Artificial Intelligence*, 1999, **112**, (1), pp. 57 – 104
14 Bortolussi, L., Milios, D., Sanguinetti, G.: 'Machine learning methods in statistical model checking and system design – tutorial', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, **9333**, pp. 323–341
15 Braázdil, T., Chatterjee, K., Chmelík, M., Forejt, V., Křetínskýy, J., Kwiatkowska, M., et al.: 'Verification of markov decision processes using learning algorithms', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, **8837**, pp. 98–114
16 Jha, S., Seshia, S.A.: 'A theory of formal synthesis via inductive learning', *Acta Informatica*, 2017, **54**, (7), pp. 693–726
17 Kroening, D., Liang, L., Melham, T., Schrammel, P., Tautschnig, M. 'Effective verification of low-level software with nested interrupts'. In: Design, Automation & Test in Europe Conference & Exhibition. DATE '15. (San Jose, CA, USA: EDA Consortium, 2015. pp. 229–234
18 Monteiro, F.R., Alves, E., Silva, I., Ismail, H., Cordeiro, L., Filho, E.L.: 'Esbmc-gpu a context-bounded model checking tool to verify cuda programs', *Science of Computer Programming*, 2017, **xx**, (y)
19 Nilsson, P., Hussien, O., Balkan, A., Chen, Y., Ames, A.D., Grizzle, J.W., et al.: 'Correct-by-construction adaptive cruise control: Two approaches', *IEEE Transactions on Control Systems Technology*, 2016, **24**, (4), pp. 1294–1307
20 Prabhakar, P., García.Soto, M. 'Formal synthesis of stabilizing controllers for switched systems'. In: Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control. HSCC '17. (, 2017. pp. 111–120
21 Esmaeil.Zadeh.Soudjani, S., Majumdar, R. 'Controller synthesis for reward collecting markov processes in continuous space'. In: Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control. HSCC '17. (, 2017. pp. 45–54
22 Ames, A.D., Tabuada, P., Jones, A., Ma, W.L., Rungger, M., SchÃijrmann, B., et al.: 'First steps toward formal controller synthesis for bipedal robots with experimental implementation', *Nonlinear Analysis: Hybrid Systems*, 2017, **25**, pp. 155 – 173
23 Tabuada, P.: 'Verification and Control of Hybrid Systems: A Symbolic Approach'. (Springer US, 2009)
24 Hasuo, I.: 'Metamathematics for systems design', *New Generation Computing*, 2017, **35**, (3), pp. 271–305
25 Zamani, M., Abate, A., Girard, A.: 'Symbolic models for stochastic switched systems: A discretization and a discretization-free approach', *Automatica*, 2015, **55**, pp. 183 – 196
26 Clarke, E.M., Zuliani, P. 'Statistical model checking for cyber-physical systems'. In: Proceedings of the 9th International Conference on Automated Technology for Verification and Analysis. ATVA'11. (, 2011. pp. 1–12
27 'Model-based testing of cyber-physical systems'. In: Song, H., Rawat, D.B., Jeschke, S., Brecher, C., editors. Cyber-Physical Systems. Intelligent Data-Centric Systems. (Boston: Academic Press, 2017. pp. 287 – 304
28 Zhang, Y., Dong, Y., Xie, F. 'Bounded model checking of hybrid automata pushdown system'. In: 2014 14th International Conference on Quality Software, Allen, TX, USA, October 2-3, 2014. (https://doi.org/10.1109/QSIC.2014.36: IEEE, 2014. pp. 190–195
29 Simko, G., Jackson, E.K. 'A bounded model checking tool for periodic sample-hold systems'. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control. HSCC '14. (, 2014. pp. 157–162
30 Sanwal, M.U., Hasan, O. 'Formal verification of cyber-physical systems: Coping with continuous elements'. In: Proceedings of the 13th International Conference on Computational Science and Its Applications - Volume 1. vol. 7971 of *ICCSA'13*. (Springer, 2013. pp. 358–371
31 Lee, H.Y. 'Towards model checking of simulation models for embedded system development'. In: International Conference on Parallel and Distributed Systems. (, 2013. pp. 452–453
32 Li, T., Tan, F., Wang, Q., Bu, L., Cao, J.N., Liu, X.: 'From offline toward real time: A hybrid systems model checking and cps codesign approach for medical device plug-and-play collaborations', *IEEE Transactions on Parallel and Distributed Systems*, 2014, **25**, (3), pp. 642–652
33 Jhala, R., Majumdar, R.: 'Software model checking', *ACM Comput Surv*, 2009, **41**, (4), pp. 21:1–21:54
34 Rungger, M., Tabuada, P.: 'A notion of robustness for cyber-physical systems', *IEEE Transactions on Automatic Control*, 2016, **61**, (8), pp. 2108–2123
35 Alur, R., Henzinger, T.A., Lafferriere, G., Pappas, G.J.: 'Discrete abstractions of hybrid systems', *Proceedings of the IEEE*, 2000, **88**, (7), pp. 971–984
36 Girard, A., Pappas, G.J.: 'Approximate bisimulation: A bridge between computer science and control theory', *European Journal of Control*, 2011, **17**, (5), pp. 568 – 578
37 Lunze, J., Lamnabhi.Lagarrigue, F.: 'Handbook of Hybrid Systems Control: Theory, Tools, Applications'. (Cambridge University Press, 2009)
38 Bargmann, H.: 'The role of stochastic modelling in engineering science', *Acta Mechanica*, 1997, **125**, (1), pp. 63–71
39 Alur, R., Dill, D.L.: 'A theory of timed automata', *Theor Comput Sci*, 1994, **126**, (2), pp. 183–235
40 Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H. 'Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems'. In: Hybrid Systems. (London, UK, UK: Springer-Verlag, 1993. pp. 209–229
41 Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P. 'What's decidable about hybrid automata?'. In: 27th Annual ACM Symposium on Theory of Computing. STOC '95. (New York, NY, USA: ACM, 1995. pp. 373–382
42 Julius, A.A., Pappas, G.J.: 'Approximations of stochastic hybrid systems', *IEEE Transactions on Automatic Control*, 2009, **54**, (6), pp. 1193–1203
43 Ghosh, M.K., Bagchi, A. In: Cagnol, J., Zolésio, J.P., editors. 'Modeling stochastic hybrid systems'. (Boston, MA: Springer US, 2005. pp. 269–280
44 Pola, G., Bujorianu, M.L., Lygeros, J., Benedetto, M.D.D.: 'Stochastic hybrid models: An overview', *IFAC Proceedings Volumes*, 2003, **36**, (6), pp. 45 – 50. iFAC Conference on Analysis and Design of Hybrid Systems
45 Lafferriere, G., Pappas, G.J., Yovine, S. 'A new class of decidable hybrid systems'. In: Hybrid Systems: Computation and Control. HSCC '99. (London, UK, UK: Springer-Verlag, 1999. pp. 137–151
46 Puri, A., Varaiya, P. 'Decidability of hybrid systems with rectangular differential inclusion'. In: Computer Aided Verification. CAV '94. (London, UK, UK: Springer-Verlag, 1994. pp. 95–104
47 Henzinger, T.A., Ho, P.H. 'Algorithmic analysis of nonlinear hybrid systems'. In: Computer Aided Verification. (London, UK, UK: Springer-Verlag, 1995. pp. 225–238
48 Broucke, M., Varaiya, P. In: Antsaklis, P., Kohn, W., Nerode, A., Sastry, S., editors. 'Decidability of hybrid systems with linear and nonlinear differential inclusions'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. pp. 77–92
49 Kesten, Y., Pnueli, A., Sifakis, J., Yovine, S. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H., editors. 'Integration graphs: A class of decidable hybrid systems'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 1993. pp. 179–208
50 Tabuada, P., Caliskan, S.Y., Rungger, M., Majumdar, R.: 'Towards robustness for cyber-physical systems', *IEEE Transactions on Automatic Control*, 2014, **59**, (12), pp. 3151–3163
51 Rungger, M., Mazo, M. Jr., Tabuada, P. 'Specification-guided controller synthesis for linear systems and safe linear-time temporal logic'. In: Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control. HSCC '13. (ACM, 2013. pp. 333–342
52 Li, Y., Liu, J. 'An interval analysis approach to invariance control synthesis for discrete-time switched systems'. In: 55th Conference on Decision and Control (CDC). (, 2016. pp. 6388–6394
53 Tabuada, P.: 'Symbolic models for control systems', *Acta Informatica*, 2007, **43**, (7), pp. 477–500
54 Keel, L.H., Bhattacharyya, S.P. In: Istepanian, R.S.H., Whidborne, J.F., editors. 'Stability margins and digital implementation of controllers'. (London: Springer London, 2001. pp. 13–24
55 Keel, L.H., Bhattacharyya, S.P.: 'Robust, fragile, or optimal?', *IEEE Transactions on Automatic Control*, 1997, **42**, (8), pp. 1098–1105
56 Bicchi, A., Marigo, A., Piccoli, B.: 'On the reachability of quantized control systems', *IEEE Transactions on Automatic Control*, 2002, **47**, (4), pp. 546–563
57 Petreczky, M., van Schuppen, J.H.: 'Realization theory for linear hybrid systems', *IEEE Transactions on Automatic Control*, 2010, **55**, (10), pp. 2282–2297
58 Petreczky, M.: 'Realization theory for linear switched systems: Formal power series approach', *Systems & Control Letters*, 2007, **56**, (9), pp. 588 – 595
59 Ye, H., Michel, A.N., Hou, L.: 'Stability theory for hybrid dynamical systems', *IEEE Transactions on Automatic Control*, 1998, **43**, (4), pp. 461–474
60 Ben.Sassi, M.A., Girard, A.: 'Computation of polytopic invariants for polynomial dynamical systems using linear programming', *Automatica*, 2012, **48**, (12), pp. 3114–3121
61 Paul, T., Kimball, J.W., Zawodniok, M., Roth, T.P., McMillin, B., Chellappan, S.: 'Unified invariants for cyber-physical switched system stability', *IEEE Transactions on Smart Grid*, 2014, **5**, (1), pp. 112–120
62 Li, Y., Liu, J. 'Computing maximal invariant sets for switched nonlinear systems'. In: Conference on Computer Aided Control System Design (CACSD). (, 2016. pp. 862–867
63 Fisher, A., Jacobson, C.A., Lee, E.A., Murray, R.M., Sangiovanni.Vincentelli, A., Scholte, E. 'Industrial cyber-physical systems – icyphy'. In: Complex Systems Design & Management. (, 2014. pp. 21–37
64 Alur, R. 'Formal verification of hybrid systems'. In: Proceedings of the 9th ACM International Conference on Embedded Software. EMSOFT '11. (, 2011. pp. 273–278
65 Prajna, S., Jadbabaie, A., Pappas, G.J.: 'A framework for worst-case and stochastic safety verification using barrier certificates', *IEEE Transactions on Automatic Control*, 2007, **52**, (8), pp. 1415–1428
66 Prajna, S., Jadbabaie, A. In: Alur, R., Pappas, G.J., editors. 'Safety verification of hybrid systems using barrier certificates'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. pp. 477–492
67 Bessa, I., Ismail, H., Palhares, R., Cordeiro, L., Filho, J.E.C.: 'Formal non-fragile stability verification of digital control systems with uncertainty', *IEEE Transactions on Computers*, 2017, **66**, (3)
68 Maler, O., Nickovic, D. In: Lakhnech, Y., Yovine, S., editors. 'Monitoring temporal properties of continuous signals'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. pp. 152–166
69 Donzé, A., Maler, O., Bartocci, E., Nickovic, D., Grosu, R., Smolka, S. 'On temporal logic and signal processing'. In: Proceedings of the 10th International Conference on Automated Technology for Verification and Analysis. ATVA'12. (, 2012. pp. 92–106
70 Veanes, M., Bjorner, N., Gurevich, Y., Schulte, W.: 'Symbolic bounded model checking of abstract state machines', *Int J Software Informatics*, 2009, **3/2-3**, pp. 149–170
71 Phan, A.D. 'Modelling and Analysis for Cyber-Physical Systems: An SMT-based approach'. Technical University of Denmark (DTU, 2015
72 Nipkow, T., Wenzel, M., Paulson, L.C.: 'Isabelle/HOL: A Proof Assistant for Higher-order Logic'. (Berlin, Heidelberg: Springer-Verlag, 2002)
73 Platzer, A. 'Differential dynamic logic for verifying parametric hybrid systems'. In: Proceedings of the 16th International Conference on Automated Reasoning

with Analytic Tableaux and Related Methods. TABLEAUX '07. (, 2007. pp. 216–232
74 Biere, A. 'Bounded model checking'. In: Handbook of Satisfiability. (IOS Press, 2009. pp. 457–481
75 Duggirala, P.S., Viswanathan, M. 'Analyzing real time linear control systems using software verification'. In: IEEE Real-Time Systems Symposium. (, 2015. pp. 216–226
76 Anta, A., Majumdar, R., Saha, I., Tabuada, P. 'Automatic verification of control system implementations'. In: Proceedings of the Tenth ACM International Conference on Embedded Software. EMSOFT '10. (, 2010. pp. 9–18
77 Ismail, H., Cordeiro, I.B.L., Filho, E.B.L., ao Edgar.Chaves.Filho, J. 'DSVerifier: A bounded model checking tool for digital systems'. In: SPIN. vol. 9232 of *LNCS*. (Stellenbosch, South Africa: Springer, 2015. pp. 126–131
78 Bessa, I.V., Ismail, H.I., Cordeiro, L.C., Filho, J.E.C.: 'Verification of fixed-point digital controllers using direct and delta forms realizations', *Design Autom for Emb Sys*, 2016, **20**, (2), pp. 95–126
79 Phan, A.D., Hansen, M.R., Madsen, J. In: Iida, S., Meseguer, J., Ogata, K., editors. 'EHRA: specification and analysis of energy-harvesting wireless sensor networks'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. pp. 520–540
80 Nakajima, S., Furukawa, S., Ueda, Y. 'Co-analysis of sysml and simulink models for cyber-physical systems design'. In: International Conference on Embedded and Real-Time Computing Systems and Applications. (, 2012. pp. 473–478
81 Junjie, T., Jianjun, Z., Jianwan, D., Liping, C., Gang, X., Bin, G., et al. 'Cyber-physical systems modeling method based on modelica'. In: 6th International Conference on Software Security and Reliability Companion. (, 2012. pp. 188–191
82 Shoukry, Y., Nuzzo, P., Puggelli, A., Sangiovanni.Vincentelli, A.L., Seshia, S.A., Tabuada, P.: 'Secure state estimation for cyber physical systems under sensor attacks: A satisfiability modulo theory approach', *IEEE Transactions on Automatic Control*, 2017, **PP**, (99), pp. 1–1
83 Liu, Y., Ning, P., Reiter, M.K. 'False data injection attacks against state estimation in electric power grids'. In: 16th ACM Conference on Computer and Communications Security. CCS '09. (New York, NY, USA: ACM, 2009. pp. 21–32. Available from: http://doi.acm.org/10.1145/1653662.1653666
84 Fiore, G., Chang, Y.H., Hu, Q., Benedetto, M.D.D., Tomlin, C.J. 'Secure state estimation for cyber physical systems with sparse malicious packet drops'. In: American Control Conference (ACC). (, 2017. pp. 1898–1903
85 AraÃžjo, R.F., Albuquerque, H.F., de Bessa, I.V., Cordeiro, L.C., Filho, J.E.C.: 'Counterexample guided inductive optimization based on satisfiability modulo theories', *Science of Computer Programming*, 2017,
86 Trindade, A.B., Cordeiro, L.C.: 'Applying SMT-based verification to hardware/-software partitioning in embedded systems', *DES AUTOM EMBED SYST*, 2016, **20**, (1), pp. 1–19
87 Rahman, M.A., Duan, Q., Al.Shaer, E. 'Energy efficient navigation management for hybrid electric vehicles on highways'. In: International Conference on Cyber-Physical Systems (ICCPS). (, 2013. pp. 21–30
88 Solar-Lezama, A., Tancau, L., Bodík, R., Seshia, S.A., Saraswat, V.A. 'Combinatorial sketching for finite programs'. In: ASPLOS. (, 2006. pp. 404–415
89 Riener, H., Könighofer, R., Fey, G., Bloem, R. 'Smt-based CPS parameter synthesis'. In: ARCH@CPSWeek 2016, 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems, Vienna, Austria. (, 2016. pp. 126–133
90 Abate, A., Bessa, I., Cattaruzza, D., Cordeiro, L., David, C., Kesseli, P., et al. 'Automated Formal Synthesis of Digital Controllers for State-Space Physical Plants'. In: CAV. vol. 10426 of *LNCS*. (, 2017. pp. 462–âĂŞ482
91 Ravanbakhsh, H., Sankaranarayanan, S. 'Counterexample-guided stabilization of switched systems using control lyapunov functions'. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control. HSCC '15. (, 2015. pp. 297–298
92 Ravanbakhsh, H., Sankaranarayanan, S. 'Robust controller synthesis of switched systems using counterexample guided framework'. In: Proceedings of the 13th International Conference on Embedded Software. EMSOFT '16. (, 2016. pp. 8:1–8:10
93 Gol, E.A., Lazar, M., Belta, C.: 'Language-guided controller synthesis for linear systems', *IEEE Transactions on Automatic Control*, 2014, **59**, (5), pp. 1163–1176
94 Holub, O., Zamani, M., Abate, A. 'Efficient hvac controls: A symbolic approach'. In: 2016 European Control Conference (ECC). (, 2016. pp. 1159–1164
95 Tabuada, P.: 'An approximate simulation approach to symbolic control', *IEEE Transactions on Automatic Control*, 2008, **53**, (6), pp. 1406–1418
96 Zamani, M., Arcak, M.: 'Compositional abstraction for networks of control systems: A dissipativity approach', *IEEE Transactions on Control of Network Systems*, 2017, **PP**, (99), pp. 1–1
97 Zamani, M., Abate, A.: 'Approximately bisimilar symbolic models for randomly switched stochastic systems', *Systems & Control Letters*, 2014, **69**, pp. 38 – 46
98 Zamani, M., van de Wouw, N., Majumdar, R.: 'Backstepping controller synthesis and characterizations of incremental stability', *Systems & Control Letters*, 2013, **62**, (10), pp. 949 – 962
99 Zamani, M., Pola, G., Mazo, M., Tabuada, P.: 'Symbolic models for nonlinear control systems without stability assumptions', *IEEE Transactions on Automatic Control*, 2012, **57**, (7), pp. 1804–1809
100 Khatib, M.A., Girard, A., Dang, T.: 'Stability verification and timing contract synthesis for linear impulsive systems using reachability analysis', *Nonlinear Analysis: Hybrid Systems*, 2017, **25**, pp. 211 – 226
101 Lesser, K., Abate, A.: 'Controller synthesis for probabilistic safety specifications using observers**this work is supported in part by the european commission iapp project ambi 324432, and by the john fell oxford university press(oup) research fund.', *IFAC-PapersOnLine*, 2015, **48**, (27), pp. 329 – 334. analysis and Design of Hybrid Systems ADHS
102 : 'Low-complexity quantized switching controllers using approximate bisimulation', *Nonlinear Analysis: Hybrid Systems*, 2013, **10**, pp. 34 – 44. special Issue related to IFAC Conference on Analysis and Design of Hybrid Systems (ADHS 12)
103 Dallal, E., Colombo, A., Del.Vecchio, D., Lafortune, S.: 'Supervisory control for collision avoidance in vehicular networks using discrete event abstractions', *Discrete Event Dynamic Systems*, 2017, **27**, (1), pp. 1–44
104 Dallal, E., Colombo, A., Vecchio, D.D., Lafortune, S. 'Supervisory control for collision avoidance in vehicular networks with imperfect measurements'. In: 52nd IEEE Conference on Decision and Control. (, 2013. pp. 6298–6303
105 Habets, L.C.G.J.M., Collins, P.J., van Schuppen, J.H.: 'Reachability and control synthesis for piecewise-affine hybrid systems on simplices', *IEEE Transactions on Automatic Control*, 2006, **51**, (6), pp. 938–948
106 Reissig, G., Weber, A., Rungger, M.: 'Feedback refinement relations for the synthesis of symbolic controllers', *IEEE Transactions on Automatic Control*, 2017, **62**, (4), pp. 1781–1796
107 Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y. 'Symbolic model checking without bdds'. In: 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems. TACAS '99. (London, UK: Springer-Verlag, 1999. pp. 193–207
108 Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C. 'Satisfiability modulo theories'. In: Handbook of Satisfiability. (IOS Press, 2009. p. 825âĂŞ885
109 Armando, A., Mantovani, J., Platania, L.: 'Bounded model checking of software using SMT solvers instead of SAT solvers', *International Journal on Software Tools for Technology Transfer*, 2009, **11**, (1), pp. 69–83
110 Prasad, M.R., Biere, A., Gupta, A.: 'A survey of recent advances in SAT-based formal verification', *STTT*, 2005, **7**, (2), pp. 156–173
111 Morse, J., Cordeiro, L., Nicole, D., Fischer, B.: 'Model checking ltl properties over ansi-c programs with bounded traces', *Softw Syst Model*, 2015, **14**, (1), pp. 65–81
112 Bradley, A.R., Manna, Z.: 'The Calculus of Computation: Decision Procedures with Applications to Verification'. (Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007)
113 Ball, T., Rajamani, S. 'Slic: A specification language for interface checking (of c)'. (, 2002. Available from: https://www.microsoft.com/en-us/research/publication/slic-a-specification-language-for-interface-checking-of-c/
114 Kroening, D., Strichman, O.: 'Decision Procedures: An Algorithmic Point of View'. 1st ed. (Springer Publishing Company, Incorporated, 2008)
115 Appel, A.W.: 'Modern Compiler Implementation in C: Basic Techniques'. (New York, NY, USA: Cambridge University Press, 1997)
116 Tseitin, G.S. In: Siekmann, J.H., Wrightson, G., editors. 'On the complexity of derivation in propositional calculus'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 1983. pp. 466–483
117 Patarin, J., Goubin, L. In: Han, Y., Okamoto, T., Qing, S., editors. 'Trapdoor one-way permutations and multivariate polynomials'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. pp. 356–368
118 Clarke, E., Kroening, D., Lerda, F. In: Jensen, K., Podelski, A., editors. 'A tool for checking ANSI-C programs'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. pp. 168–176
119 Merz, F., Falke, S., Sinz, C. 'LLBMC: bounded model checking of C and C++ programs using a compiler IR'. In: International Conference on Verified Software: Theories, Tools, Experiments. vol. 7152 of *LNCS*. (, 2012. pp. 146–161
120 Cordeiro, L., Fischer, B. 'Verifying multi-threaded software using SMT-based context-bounded model checking'. In: 33rd International Conference on Software Engineering. (, 2011. pp. 331–340
121 Ivančić, F., Shlyakhter, I., Gupta, A., Ganai, M.K. 'Model checking c programs using F-SOFT'. In: International Conference on Computer Design. ICCD '05. (Washington, DC, USA: IEEE Computer Society, 2005. pp. 297–308
122 Cordeiro, L., Fischer, B., Marques.Silva, J.: 'SMT-based bounded model checking for embedded ANSI-C software', *IEEE Trans Software Eng*, 2012, **38**, (4), pp. 957–974
123 Clarke, E., Kroening, D., Strichman, O., Ouaknine, J. 'Completeness and complexity of bounded model checking'. In: VMCAI. vol. 2937 of *LNCS*. (, 2004. pp. 85–96
124 Ganai, M.K., Gupta, A. 'Completeness in SMT-based BMC for software programs'. In: DATE. (, 2008. pp. 831–836
125 Eén, N., Sörensson, N.: 'Temporal induction by incremental sat solving', *Electronic Notes in Theoretical Computer Science*, 2003, **89**, (4), pp. 543–560
126 Sheeran, M., Singh, S., Stålmarck, G. 'Checking safety properties using induction and a sat-solver'. In: Proceedings of the Third International Conference on Formal Methods in Computer-Aided Design. FMCAD '00. (London, UK, UK: Springer-Verlag, 2000. pp. 108–125
127 : 'IEEE standard for floating-point arithmetic', *IEEE Std 754-2008*, 2008, pp. 1–70
128 Goldberg, D.: 'What every computer scientist should know about floating-point arithmetic', *ACM Comput Surv*, 1991, **23**, (1), pp. 5–48
129 'PIPS: Automatic parallelizer and code transformation framework'. (, . accessed 21 February 2016. https://pips4u.org/
130 Henry, J., Monniaux, D., Moy, M.: 'PAGAI: A path sensitive static analyser', *Electron Notes Theor Comput Sci*, 2012, **289**, pp. 15–25
131 Beyer, D., Dangl, M., Wendler, P. In: Kroening, D., Păsăreanu, C.S., editors. 'Boosting k-induction with continuously-refined invariants'. (Cham: Springer International Publishing, 2015. pp. 622–640
132 Gadelha, M.Y.R., Ismail, H.I., Cordeiro, L.C.: 'Handling loops in bounded model checking of c programs via k-induction', *International Journal on Software Tools for Technology Transfer*, 2017, **19**, (1), pp. 97–114
133 Brain, M., Joshi, S., Kroening, D., Schrammel, P. In: Blazy, S., Jensen, T., editors. 'Safety verification and refutation by k-invariants and k-induction'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. pp. 145–161
134 Donaldson, A.F., Haller, L., Kroening, D., Rümmer, P. In: Yahav, E., editor. 'Software verification using k-induction'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. pp. 351–368

135 Rocha, W., Rocha, H., Ismail, H., Cordeiro, L.C., Fischer, B. 'Depthk: A k-induction verifier based on invariant inference for C programs - (competition contribution)'. In: Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II. vol. 10206 of *LNCS*. (Springer, 2017. pp. 360–364

136 Donaldson, A.F., Kroening, D., Ruemmer, P. 'SCRATCH: A tool for automatic analysis of DMA races'. In: 16th ACM Symposium on Principles and Practice of Parallel Programming. PPoPP '11. (New York, NY, USA: ACM, 2011. pp. 311–312

137 Grosse, D., Le, H.M., Drechsler, R. 'Induction-based formal verification of SystemC TLM designs'. In: 10th International Workshop on Microprocessor Test and Verification. (, 2009. pp. 101–106

138 Bradley, A. 'IC3 and beyond: Incremental, inductive verification'. In: Computer Aided Verification. vol. 7358 of *Lecture Notes in Computer Science*. (Springer Berlin Heidelberg, 2012. pp. 4–4

139 Hassan, Z., Bradley, A.R., Somenzi, F. 'Better generalization in IC3'. In: Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013. (, 2013. pp. 157–164

140 Bradley, A.R.: 'Understanding IC3'. (Technical Report. Accessed on February 23rd 2018: ECEE Department, University of Colorado at Boulder,)

141 Jovanović, D., Dutertre, B. 'Property-directed k-induction'. In: FMCAD. (, 2016. pp. 85–92

142 McMillan, K.L. 'Interpolation and sat-based model checking'. In: CAV. vol. 2725 of *LNCS*. (, 2003. pp. 1–13

143 McMillan, K.L. 'Applications of craig interpolants in model checking'. In: TACAS. vol. 3440 of *LNCS*. (, 2005. pp. 1–12

144 David, C., Kroening, D., Lewis, M. 'Using program synthesis for program analysis'. In: 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning. LPAR-20 2015. (New York, NY, USA: Springer-Verlag New York, Inc., 2015. pp. 483–498

145 Alur, R., Bodik, R., Juniwal, G., Martin, M.M.K., Raghothaman, M., Seshia, S.A., et al. 'Syntax-guided synthesis'. In: Formal Methods in Computer-Aided Design. (, 2013. pp. 1–8

146 Solar.Lezama, A.: 'Program sketching', *International Journal on Software Tools for Technology Transfer*, 2013, **15**, (5), pp. 475–495

147 Sharma, R., Aiken, A. 'From invariant checking to invariant inference using randomized search'. In: Computer Aided Verification. (New York, NY, USA: Springer-Verlag New York, Inc., 2014. pp. 88–105

148 Abate, A., Bessa, I., Cattaruzza, D., Chaves, L., Cordeiro, L.C., David, C., et al. 'Dssynth: an automated digital controller synthesis tool for physical plants'. In: Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017. (, 2017. pp. 919–924

149 Moore, R.E.: 'Interval analysis'. vol. 4. (Prentice-Hall, 1966)

150 Fairley, P.: 'Self-driving cars have a bicycle problem [news]', *IEEE Spectrum*, 2017, **54**, (3), pp. 12–13

151 Bortolussi, L., Milios, D., Sanguinetti, G.: 'Smoothed model checking for uncertain continuous-time markov chains', *Information and Computation*, 2016, **247**, (C), pp. 235–253

152 Behrend, J., Lettnin, D., Grünhage, A., Ruf, J., Kropf, T., Rosenstiel, W.: 'Scalable and optimized hybrid verification of embedded software', *J Electron Test*, 2015, **31**, (2), pp. 151–166

153 Lettnin, D., Nalla, P.K., Behrend, J., Ruf, J., Gerlach, J., Kropf, T., et al. 'Semi-formal verification of temporal properties in automotive hardware dependent software'. In: Design, Automation Test in Europe Conference Exhibition. (, 2009. pp. 1214–1217

154 Vilca, J., Adouane, L., Mezouar, Y.: 'Optimal multi-criteria waypoint selection for autonomous vehicle navigation in structured environment', *Journal of Intelligent & Robotic Systems*, 2016, **82**, (2), pp. 301–324

155 De.Florio, V. 'Antifragility = elasticity + resilience + machine learning'. In: Proceedings of the 1st International Workshop âĂIJFrom Dependable to Resilient, from Resilient to Antifragile Ambients and SystemsâĂİ. Procedia Computer Science. (Elsevier, 2014. pp. 834–841

156 Ramalho, M., Freitas, M., Sousa, F., Marques, H., Cordeiro, L., Fischer, B. 'SMT-based bounded model checking of c++ programs'. In: 20th International Conference and Workshops on the Engineering of Computer Based Systems. ECBS '13. (Washington, DC, USA: IEEE Computer Society, 2013. pp. 147–156

157 Pereira, P., Albuquerque, H., da Silva, I., Marques, H., Monteiro, F., Ferreira, R., et al.: 'Smt-based context-bounded model checking for cuda programs', *Concurrency and Computation: Practice and Experience*, 2016, cpe.3934

158 Monteiro, F.R., Cordeiro, L.C., de Lima.Filho, E.B. 'Bounded Model Checking of C++ Programs Based on the Qt Framework'. In: Global Conference on Consumer Electronics. (IEEE, 2015. pp. 179–180

159 Beyer, D. 'Status report on software verification'. In: Tools and Algorithms for the Construction and Analysis of Systems. vol. 8413 of *LNCS*. (Springer Berlin Heidelberg, 2014. pp. 373–388

160 Beyer, D. 'Software verification and verifiable witnesses'. In: Tools and Algorithms for the Construction and Analysis of Systems. vol. 9035 of *LNCS*. (Springer Berlin Heidelberg, 2015. pp. 401–416

161 Ábrahám, E., Abbott, J., Becker, B., Bigatti, A.M., Brain, M., Buchberger, B., et al.: 'Satisfiability checking and symbolic computation', *ACM Comm Computer Algebra*, 2016, **50**, (4), pp. 145–147. Available from: `http://doi.acm.org/10.1145/3055282.3055285`

162 Kahlon, V., Wang, C., Gupta, A. In: Bouajjani, A., Maler, O., editors. 'Monotonic partial order reduction: An optimal symbolic partial order reduction technique'. (Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. pp. 398–413

163 Morse, J. 'Expressive and Efficient Bounded Model Checking of Concurrent Software'. University of Southampton, 2015

164 Inverso, O., Tomasco, E., Fischer, B., La.Torre, S., Parlato, G. 'Bounded model checking of multi-threaded c programs via lazy sequentialization'. In: Computer Aided Verification. (New York, NY, USA: Springer-Verlag New York, Inc., 2014. pp. 585–602

165 Zheng, M., Rogers, M.S., Luo, Z., Dwyer, M.B., Siegel, S.F. 'CIVL: formal verification of parallel programs'. In: 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015. (IEEE/ACM: IEEE Computer Society, 2015. pp. 830–835. Available from: `https://doi.org/10.1109/ASE.2015.99`

166 Kroening, D., Poetzl, D., Schrammel, P., Wachter, B. 'Sound static deadlock analysis for c/pthreads'. In: ASE. (, 2016. pp. 379–390

167 Grumberg, O., Lerda, F., Strichman, O., Theobald, M. 'Proof-guided underapproximation-widening for multi-process systems'. In: 32nd Symposium on Principles of Programming Languages. POPL '05. (New York, NY, USA: ACM, 2005. pp. 122–131

168 McMillan, K.L. 'Widening and interpolation'. In: 18th International Symposium on Static Analysis. SAS'11. (Berlin, Heidelberg: Springer-Verlag, 2011. pp. 1–1

169 Bessa, I., Abreu, R., Filho, J.E., Cordeiro, L. 'Smt-based bounded model checking of fixed-point digital controllers'. In: IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society. (, 2014. pp. 295–301

170 Abreu, R.B., Cordeiro, L.C., Filho, E.B.L. 'Verifying Fixed-Point Digital Filters using SMT-Based Bounded Model Checking'. In: SBrT. (, 2013.

171 Chaves, L., Bessa, I., Cordeiro, L. 'Dsvalidator: An automated counterexample reproducibility tool for digital systems'. (arXiv Document, 2016.

172 Mello, D.P.M., Freitas, M.L., Cordeiro, L., Silva.Júnior, W.S., Bessa, I.V., Lima.Filho, E.B., et al. 'Verification of magnitude and phase responses in fixed-point digital filters'. In: Brazilian Symposium on Telecommunications and Signal Processing (to appear). (, 2017. pp. 1–5

173 d. S..Alves, E.H., Cordeiro, L.C., d. L..Filho, E.B. 'Fault localization in multi-threaded c programs using bounded model checking'. In: Brazilian Symposium on Computing Systems Engineering. (, 2015. pp. 96–101

174 Garcia, M., Monteiro, F., Cordeiro, L., de Lima.Filho, E. In: Bošnački, D., Wijs, A., editors. 'Esbmc^{QtOM}: A bounded model checking tool to verify qt applications'. (Cham: Springer International Publishing, 2016. pp. 97–103

175 Lopes, B.C., Auler, R.: 'Getting Started with LLVM Core Libraries'. (Packt Publishing, 2014)

176 'Why Apple's swift language will instantly remake computer programming'. (, . accessed 21 February 2016. `http://www.wired.com/2014/07/apple-swift/`

177 Jackson, D., Vaziri, M. 'Correct or usable? the limits of traditional verification (impact paper award)'. In: 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. FSE 2016. (New York, NY, USA: ACM, 2016. pp. 11–11

178 Morse, J., Cordeiro, L., Nicole, D., Fischer, B. 'Handling Unbounded Loops with ESBMC 1.20 - (Competition Contribution)'. In: Tools and Algorithms for the Construction and Analysis of Systems. LNCS. (, 2013. pp. 619–622

179 Morse, J., Ramalho, M., Cordeiro, L.C., Nicole, D., Fischer, B. 'ESBMC 1.22 - (competition contribution)'. In: Tools and Algorithms for the Construction and Analysis of Systems. vol. 8413 of *LNCS*. (, 2014. pp. 405–407

180 Haran, A., Carter, M., Emmi, M., Lal, A., Qadeer, S., Rakamarić, Z. 'SMACK+Corral: A modular verifier'. In: Tools and Algorithms for the Construction and Analysis of Systems. (New York, NY, USA: Springer-Verlag New York, Inc., 2015. pp. 451–454

181 Beyer, D.: 'Reliable and reproducible competition results with benchexec and witnesses (report on SV-COMP 2016)', *LNCS*, 2016, **9636**, pp. 887–904

182 Beyer, D., Dangl, M., Dietsch, D., Heizmann, M., Stahlbauer, A. 'Witness validation and stepwise testification across software verifiers'. In: ESEC/SIGSOFT Foundations of Software Engineering. ESEC/FSE 2015. (New York, NY, USA: ACM, 2015. pp. 721–733

183 Rocha, H., Barreto, R., Cordeiro, L., Neto, A.D. 'Understanding programming bugs in ansi-c software using bounded model checking counter-examples'. In: 9th International Conference on Integrated Formal Methods. IFM'12. (Berlin, Heidelberg: Springer-Verlag, 2012. pp. 128–142

184 Jesus, A.S., Rodrigues, R.N., Ferreira, A.N.G., Melo, W.C., Lima.Filho, E.B., Silva.Júnior, W.S. 'Automatic antenna alignment system for satellite receivers operating in c and ku bands'. In: Brazilian Symposium on Telecommunications and Signal Processing (to appear, in Portuguese). (, 2017. pp. 1–5

185 Amoedo, D.A., da Silva.JÃžnior, W.S., de Lima.Filho, E.B. 'Parameter selection for SVM in automatic modulation classification of analog and digital signals'. In: International Telecommunications Symposium. (, 2014. pp. 1–5

186 Hamel, L. 'On the use of machine learning in formal software verification'. (Dept. of Computer Science and Statistics, University of Rhode Island, 2003. technical Report TR03-294

187 Phuc, N.V. 'The Application of Machine Learning Methods in Software Verification and Validation'. University of Texas at Austin, 2010

188 Bridge, J.P., Holden, S.B., Paulson, L.C.: 'Machine learning for first-order theorem proving', *Journal of Automated Reasoning*, 2014, **53**, (2), pp. 141–172

189 Hutter, F., Babic, D., Hoos, H.H., Hu, A.J. 'Boosting verification by automatic tuning of decision procedures'. In: Formal Methods in Computer Aided Design. FMCAD '07. (Washington, DC, USA: IEEE Computer Society, 2007. pp. 27–34

190 Seshia, S.A., Hu, S., Li, W., Zhu, Q.: 'Design automation of cyber-physical systems: Challenges, advances, and opportunities', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017, **PP**, (99), pp. 1–1

191 Jha, S., Gulwani, S., Seshia, S.A., Tiwari, A. 'Synthesizing switching logic for safety and dwell-time requirements'. In: Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems. ICCPS '10. (, 2010. pp. 22–31

192 Sadigh, D., Kim, E.S., Coogan, S., Sastry, S.S., Seshia, S.A. 'A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications'. In: 53rd IEEE Conference on Decision and Control. (IEEE, 2014. pp. 1091–1096

193 Lucky, R.W.. 'Antifragile Systems'. (, 2013. (accessed 13 December 2017). https://spectrum.ieee.org/telecom/ wireless/antifragile-systems