SMT-based Control Safety Property Checking in Cyber-Physical Systems under Timing Uncertainties

Anand Yeolekar[1,2], Ravindra Metta[1,2], and Samarjit Chakraborty[3]
[1]TCS Research, Tata Consultancy Services, Pune, India
[2]Technical University of Munich, Germany
[3]The University of North Carolina at Chapel Hill, USA
Email: {anand.yeolekar, r.metta}@tcs.com, samarjit@cs.unc.edu

***Abstract*—This paper outlines formal methods and design automation techniques for *exact* checking of control safety and reachability properties of cyber-physical systems (CPS), under timing uncertainties (common deadline miss handling and control update policies). While such checking is often fraught with state-space explosion problems and is hence not scalable. This paper discusses a new joint encoding of control and scheduling behaviors as a satisfiability-modulo-theory (SMT) formulation and a novel abstraction-refinement procedure with incremental solving, to scale the analysis. In addition, we also outline empirical performance results of multiple well-known SMT solvers for this problem. These results can inform practical decision making for large scale control safety verification in the industry.**

***Index Terms*—Cyber-Physical Systems, Real-Time Systems, Formal Verification, SMT Solvers**

I. Introduction

Current model-based design (MBD) approachs [1], [2] for systematic development of embedded software emphasizes on constructing models of the system at various levels of abstraction. A high-level model of the embedded system describes the possibly idealized control law (of the control application under development) and plant dynamics, in some domain-specific language [3]. Such languages are developed keeping in mind the domain expert, in this case the control designer, and generally hide the complexities associated with the software that realizes the control law. The control model is subsequently realized in software as a set of tasks.

The MBD process is naturally layered, beginning with an application or control model, followed by a task model, and so on. During the development, several assumptions are made about the implementation details such as software tasks execute instantaneously, library calls execute in constant time, and resources are not held-up [4], [5]. While these assumptions simplify high-level design, subtle effects can arise from these implementation choices [6], [7].

A central concern in this development process is that of ensuring **timing correctness** [8]–[12]. Each level of modeling may employ different verification and validation strategies, particularly for checking timing behaviour, as appropriate for that level of abstraction. For example, the *control designer* employs control-theoretic analysis in developing the control application, and is concerned with properties such as stability, overshoot, and response time [13]. The *software engineer* employs techniques such as static analysis, dynamic analysis, and assertion checking, for checking properties like input-output correctness, reading stale inputs, runtime crash detection, and time traceability. The *systems engineer* employs schedulability analysis, timing diagrams and is concerned with task execution times, deadline misses, and response times [14], [15]. Checking timing is integral to each level of modeling, yet, due to the wide separation between model levels and their semantics, it is difficult to cohesively and consistently reason about timing across models, or allow automatic feedback of timing information and constraints between models [16].

Often, timing assumptions at design levels do not hold at the implementation level, thereby requiring significant post-implementation re-engineering [17], [18]. For example, tasks mapped to an ECU may face delay or preemption due to scheduling policies, or release timing uncertainties (jitter). A task that is delayed in its execution may output the control action late in time. Similarly, if a message is delayed due to communication constraints, the receptor task, scheduled on time, may read a stale value of the message, resulting in a possibly erroneous output. **Clearly, correctness of the computation depends on inputs and outputs made available at correct time** [19]. When a critical control task misses deadline, say due to transient overload on the processor, the corresponding control computation may be delayed (or skipped), causing a deviation from the expected *ideal* behaviour. To summarize, the combinations of (i) delay arising from low-level timing uncertainties at the task level, (ii) deadline miss handling policy such as KILL or CONTINUE at the scheduler level, and (iii) control update policy such as applying ZERO or HOLD at the control level, can adversely affect the functionality of the embedded system [20], [21].

Based on these observations, and especially towards checking timing properties of CPS, we propose a *joint encoding* of two important layers in MBD, namely control and scheduling [22]. In this work, the specific goal of our encoding is to analyze system behaviour for bugs that occur based on the interaction between control and scheduling layers. Such bugs are often subtle, near-impossible to reproduce by *separately* analyzing control or scheduling. To establish timing correctness with respect to control properties, we propose a formal modeling and analysis that *precisely maps* task runs containing all permitted sequences of deadline misses to the corresponding (erroneous) control behaviour.

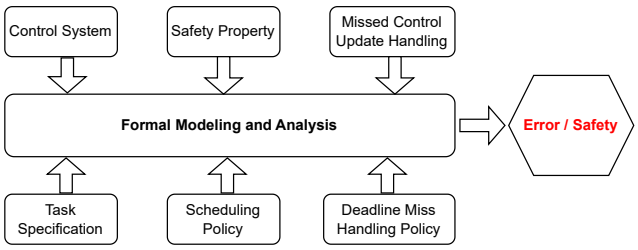

Figure 1: High-level approach

Figure 1 illustrates our approach. The main technical novelty is our proposed use of an abstraction-refinement approach on top of an satisfiability-modulo-theory (SMT) encoding that unifies modeling of control behaviour, scheduling behaviour and timing properties of interest. Formal modeling and analysis has the benefit of precise reasoning but generally suffers from scalability, making industrial-level model checking hard. Preliminary results from our exploration of constructing abstractions for validating timing properties of real-time tasks, together with incremental constraint solving, are encouraging for analysis of small-to-medium size systems. In contrast to existing closed-form analytical approaches to timing analysis, our method enables modeling and checking a diversity of timing properties across layers in MBD. In addition, the encoding enables design space exploration of high-level system parameters, and the useful debugging ability of test cases exposing timing anomalies.

Further, there are several state-of-art SMT solvers all of which employ different heuristics, such as Z3, MathSAT and CVC. It is well known that the performance of any of the solvers for a given SMT encoding cannot be predicted [23], [24]. Therefore, we conducted a preliminary empirical evaluation of our encoding with well-known SMT solvers. These experimental results are intended to aid the designers in practical decision making for large scale control safety verification in the industry.

Related Work: Several works focus on controller verification [25] and on the correctness checking scheduling control tasks [26], [27]. Verification of a combination of control and timing models has been explored in [28], [29]. The impact of deadline misses on control stability has been investigated in [30]–[33], but as observed in [34], a stable control system might still violate safety properties, hence the need of an approach to *rigorously* check safety properties. Alternatively, isolating safety-critical control task from timing interference due to other tasks have also been studied [35], [36]. Many control strategies have complex and conditional control flow, making it difficult to schedule them and safely estimate their timing behaviors [37]; scheduling techniques to address this have also been explored [38]. Further, previous approaches to measuring control performance [39], [40] and adapting the control system [41] to guarantee performance [42], [43] require a bound to be specified on the number of consecutive deadline misses. These approaches assume worst-case task-level behaviour that leads to imprecise modeling of deadline misses and generally report pessimistic results. In contrast, our approach jointly and faithfully models task runs and control evolution, for *precise* analysis.

II. System Model and Encoding

We briefly describe the system model and encoding, along with the implemented tool architecture, that can check control safety property violations. Detailed description of the full encoding with proofs is presented in [44].

A. Control system model and evolution

A *discrete* control system describing the plant model is defined as:

$$x_{k+1} = Ax_k + Bu_k \qquad u_k = R - Kx_k \tag{1}$$

where $x \in \mathbb{R}^{n\times 1}$ is the (discrete) state vector, $n \geq 1$ is the control system dimension; $k \in \mathbb{N}$ denotes the discrete steps of evolution; $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times 1}$ matrices specify the plant with timestep δ, the control action $u \in \mathbb{R}$ is computed using a state feedback vector $K \in \mathbb{R}^{1\times n}$. The initial state x_0 lies in a box $[\underline{X}_0, \overline{X}_0]$.

We *unroll* the system in Eqn. 1 up to a *user-specified* bound h by introducing *symbolic variables* $x_{j,k}$ corresponding to the control states, and u_k corresponding to the control update, where j ranges over the dimension of the control system $1, 2, \ldots, n$, and k ranges over the discrete steps of evolution $0, 1, \ldots, h$. Trajectories are encoded as:

$$\phi_{traj} \coloneqq \bigwedge_{k=0}^{h} \left(\bigwedge_{i=1}^{n} x_{i,k+1} = \sum_{j'=1}^{j} A_{i,j'} x_{j',k} + B_i u_k \right) \tag{2}$$

The initial plant state is encoded as:

$$\phi_{init} \coloneqq \forall j : \ \underline{X}_{j,0} \leq x_{j,0} \leq \overline{X}_{j,0} \tag{3}$$

The control safety property *violation* over the trajectory is encoded as:

$$\phi_{prop} \coloneqq \neg(\underline{X}_{j,k} \leq x_{j,k} \leq \overline{X}_{j,k}) \ , \ 0 \leq k \leq h \tag{4}$$

where $[\underline{X}_{j,k}, \overline{X}_{j,k}]$ denotes the user-specified safety interval (or safety pipe) for j-th dimension.

B. Task specification

The controller is realized in software via a set of tasks $\mathcal{T}$. We assume non-preemptive earliest-deadline-first (NP-EDF) scheduling policy, scheduled on a unicore processor. A task $\tau_i \in \mathcal{T}$ is defined as $\left(O, J, \underline{E}, \overline{E}, P\right)$, where i is a unique task id, O is the task offset, J denotes release jitter faced by each task instance, $\underline{E}$ and $\overline{E}$ denote the best- and worst-case execution times of the task, and P denotes the period. We assume τ_0 corresponds to the controller task with period set to the discretization timestep: $P^0 = \delta$.

Task instances, termed jobs, are spawned up to the horizon h, and symbolic variables r, s, e, d (corresponding to release, start, end and deadline) are introduced for each spawned job. Due to jitter, the instant of job release lies in the interval $[kP^i + O^i,\ kP^i + O^i + J^i]$. Task deadlines are specified as $d^i_k = O^i + (k+1)P^i$, and a deadline *miss* occurs when $e^i_k > d^i_k$. Under CONTINUE policy, jobs are eventually scheduled even if they miss deadline, and under KILL, jobs are aborted

in case of *conservative* deadline miss (*i.e.*, a job is aborted if its execution does not begin by $d_k^i - \overline{E^i}$). All times are assumed to be integers, scaled to the appropriate unit *e.g.* milliseconds.

C. Task runs

A *run* of the task set is a timed sequence of jobs, $\langle \ldots, (i, k, s_k^i, e_k^i), \ldots \rangle$, respecting the scheduling policy, work conservation (*i.e.*, processor can't idle in the presence of ready-to-run jobs) and deadline miss policy. Runs of the task set are encoded as:

$$\begin{aligned} \phi_{runs} \coloneqq \forall\ (i,k):\ & r_k^i \le s_k^i \\ \wedge\ & kP^i + O^i \le r_k^i \le kP^i + O^i + J^i \\ \wedge\ & \underline{E}^i \le e_k^i - s_k^i \le \overline{E}^i \ \text{(CONTINUE)} \\ \wedge\ & (s_k^i + \overline{E}^i \le d_k^i \Rightarrow \underline{E}^i \le e_k^i - s_k^i \le \overline{E}^i)\ \text{(KILL)} \\ \wedge\ & (s_k^i + \overline{E}^i > d_k^i \Rightarrow e_k^i = s_k^i)\ \text{(KILL)} \end{aligned} \tag{5}$$

We assume the scheduling is *work-conserving i.e.* a ready job must be scheduled as soon as the processor is available. Observe that multiple runs of the task set are possible due to (i) release jitter experienced by each job, (ii) variable execution budget leading to non-deterministic termination time for each job, and (iii) arbitrary selection of equal-priority ready jobs. These runs can have varying impact on the control performance and need to be analyzed rigorously.

D. Control update modeling

We admit ZERO and HOLD policies for control update u, where ZERO applies $u = 0$ when the corresponding control task instance misses deadline, and HOLD applies the previously computed value. Control updates are encoded as:

$$\begin{aligned} \phi_{u\ rt} \coloneqq & \forall k : u_k = 0,\ \text{if } k = 0 \\ \wedge\ & e_{k-1}^0 \le d_{k-1}^0 \Rightarrow u_k = R - \sum_{j=1} K_j x_{j,k-1}\ \text{(CONTINUE)} \\ \wedge\ & s_{k-1}^0 + \overline{E}^0 \le d_{k-1}^0 \Rightarrow u_k = R - \sum_{j=1}^{n} K_j x_{j,k-1}\ \text{(KILL)} \\ \wedge\ & s_{k-1}^0 + \overline{E}^0 > d_{k-1}^0 \Rightarrow u_k = u_{k-1}\ \text{(HOLD-KILL)} \\ \wedge\ & s_{k-1}^0 + \overline{E}^0 > d_{k-1}^0 \Rightarrow u_k = 0\ \text{(ZERO-KILL)} \end{aligned} \tag{6}$$

E. Abstraction

The constraints listed in the equations through Sections II-A to II-D admit more than the set of permissible control and scheduling behaviours of the system, thus forming an *abstraction*. For example, Eqn. 5 admits spurious behaviour such as overlapping jobs and out-of-schedule orderings between jobs within task runs. This is because these constraints are not present upfront in the initial encoding. Similarly, constraints from Sec. II-D do not specify the computation of u in the case of a deadline miss under CONTINUE policy, thereby admitting more trajectories than permissible. Thus the joint encoding needs to be (iteratively) *refined* on detecting spurious counterexamples reported by the backend SMT solver. In case the solver returns *unsat*, indicating the constraints conflict, we can conclude the property is safe over the original system.

III. REFINEMENT

A *solution* to the set of constraints reported by an SMT solver assigns concrete values to all the symbolic variables in the formula, consisting of the task run and trajectory, including the initial state. If either of the two is *spurious*, we proceed to block such a trace from the abstraction, by identifying the causes of spuriousness. The task run can be spurious due to presence of overlapping jobs, scheduling policy violation, or work conservation violation; and the trajectory can be spurious due to presence of unconstrained control update resulting from a deadline miss. The abstraction is refined by constructing an implication that *blocks* the spurious counterexample or solution.

A. Overlapping jobs

Suppose job (i, j) overlaps with (i', j') as observed in the counterexample, with $s_j^i \le s_{j'}^{i'}$. This is possible as the abstraction does not prevent overlaps *upfront*. To block this overlap as witnessed in this trace, we construct:

$$B_{ov} \coloneqq (s_j^i \le s_{j'}^{i'} \ \wedge\ s_{j'}^{i'} < e_j^i) \Rightarrow e_j^i \le s_{j'}^{i'} \tag{7}$$

B. Schedule violation

Suppose job (i, j) precedes (i', j') in the run but this precedence violates the scheduling policy. Under NP-EDF, (i, j) can precede (i', j') if and only if the deadline of (i, j) is *no later than* that of (i', j'), or (i, j) is scheduled strictly before (i', j') is released. Thus we construct the blocking implication:

$$B_{sv} \coloneqq s_j^i < s_{j'}^{i'} \Rightarrow (d_j^i \le d_{j'}^{i'} \ \vee\ s_j^i < r_{j'}^{i'}) \tag{8}$$

C. Work conservation violation

Here, the processor is found idling in the presence of a ready-to-run job. The basic idea behind blocking this spurious behaviour is to *statically* enumerate the (conservative) set of jobs that could precede this job across all task runs, construct an implication based on how these jobs are ordered with respect to the idle gap, and "reschedule" this job such that the idling is blocked.

D. Unconstrained control updates

The control update is left unspecified in the event of a control task instance missing deadline, under CONTINUE policy. The basic idea for refining control updates is to locate the (relatively) latest control job that was scheduled in the task run, compute the control update issued as a result of the execution of this job, and use this as the "freshest" value. The two cases to be considered here are whether the previously scheduled job itself met its deadline (in which case the corresponding u was correctly computed via Eqn. 6), or it too missed deadline (in which case the corresponding u value has to be constrained by identifying control states at its start time). Finally, we construct an implication to constrain the u value based on these two cases.

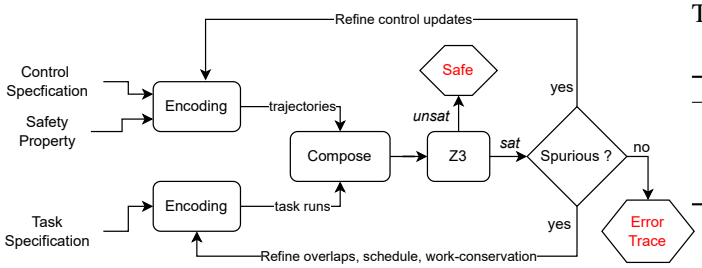

Figure 2: Tool Architecture

Table I: Synthetic task set for F1Tenth car model

id	offset	period	[BCET, WCET]	jitter
τ_0	0	20	$[4, 6]$	2
τ_1	0	20	$[4, 6]$	2
τ_2	0	20	$[4, 6]$	2
τ_3	5	40	$[5, 10]$	2

IV. Tooling and Experiments

The tool design is shown in Fig. 2. The tool has been implemented using Python3.0 scripts and the Z3 [45] SMT solver API. The initial set of constraints forms an abstraction of the set of system behaviours, and several iterations of refinement are generally needed. Each refinement step constructs and adds a number of blocking implications to the existing set of clauses in the current Z3 context. The iterative refinement procedure leverages incremental solving abilities of Z3, helping scale the symbolic analysis.

We present a case study of checking a control safety property from a CPS, the F1Tenth [46] model car, suitably adapted for our setting (linearized, discretized at 20ms), with controller adapted from [47]. The system dynamics are specified as:

$$x_{k+1} = \begin{bmatrix} 1 & 0.13 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0.02559 \\ 0.3937 \end{bmatrix} u_k$$

$$u_k = \begin{bmatrix} 0.2935 & 0.4403 \end{bmatrix} x_{k-1}$$

The (synthetic) task set implementing this controller is described in Table I. The control safety property of interest is that the steering angle should not deviate by more than 0.2 units from the ideal behaviour: $-0.2 \leq x_2 - x_{ideal} \leq 0.2$, to be checked under the four combinations of ZERO-KILL, HOLD-KILL, HOLD-CONT, ZERO-CONT. The tool implementation currently supports refinements only with Z3, hence we evaluate different SMT solvers only for the concluding refinement step. Table II presents the evaluation over 4 different SMT solvers: Z3 [45], CVC4 [48] , CVC5 [49], and MathSAT [50]. We chose these solvers as these are not only some of the most popular SMT solvers, they also support many different kinds of SMTLIB logics, in particular those that allow a mix of integers and reals, required by our encoding.

In this Table, *T* denotes the corresponding solver run-time ceiled to the nearest second and *M* denotes the corresponding solver's memory consumption ceiled to the nearest MB. The column *Policy* denotes the deadline miss and control update policies, and *Safe?* denotes the (expected) result reported by the solvers, where *Yes* means that the control property is safe, and *No* means there exists a violation of the safety property.

Table II: Evaluating different SMT solvers on F1Tenth model

F1TENTH		Z3		CVC4		CVC5		MasthSAT	
Policy	Safe?	T	M	T	M	T	M	T	M
HOLD-CONT	Yes	1	43	1	33	1	39	1	26
HOLD-KILL	Yes	40	48	38	92	68	90	217	65
ZERO-CONT	No	1	42	1	33	1	39	1	28
ZERO-KILL	No	11	49	12	56	18	67	4	34

Notice that all solvers could correctly detect property satisfaction as well as violation, with Z3 offering the overall trade off in time and memory, while CVC4 is the fastest. Interestingly, MathSAT is the fastest on the ZERO-KILL combination, and MatSAT is the slowest on HOLD-KILL variation. These preliminary results clearly indicate that it is hard to predict which solver fares well on a particular problem instance, and needs more investigation.

V. Concluding Remarks

Model-driven approaches have been widely used for the design of embedded and cyber-physical systems, especially in the domain of control and signal processing [51]. They allow verification, and synthesis and have been shown to be useful also for security and fault tolerance [52]–[54]. More importantly, they allow compositional design and thereby help tackle design complexity [55]. But as systems become more heretogeneous and distributed [56], such traditional compositional approaches are breaking down. In particular, assumptions made at the modeling stage, *e.g.*, those related to timing, might not hold during implementation. To address this, we have presented an approach for *joint* encoding of control and scheduling layers of CPS, that can check for control safety violations due to low-level timing uncertainties. The analysis is *precise* and the approach scales to medium-sized systems due to the refinement procedure implemented as incremental solving. Preliminary evaluation with different SMT solvers indicates a significant variation in solver performance in both satisfiable and unsatisfiable instances.

In the future, we would like to extend the approach to analyze industrial-size systems, with richer specifications, such as admitting interrupts, task dependencies and networked control. With industrial CPS increasingly adopting *learning components*, we would like to extend our abstraction-refinement approach to analyze the new setting of interest – interaction between control, scheduling and learning – where the learning strategy impacts both layers.

Acknowledgments: This work is partially funded by the NSF grant 2038960.

References

[1] W. Chang *et al.*, "Model-based design of resource-efficient automotive control software," in *35th International Conference on Computer-Aided Design (ICCAD)*, 2016.

[2] Q. Zhu and A. L. Sangiovanni-Vincentelli, "Codesign methodologies and tools for cyber-physical systems," *Proc. IEEE*, vol. 106, no. 9, pp. 1484–1500, 2018.

[3] D. Roy *et al.*, "Waterfall is too slow, let's go agile: multi-domain coupling for synthesizing automotive cyber-physical systems," in *37th International Conference on Computer-Aided Design (ICCAD)*, 2018.

[4] M. Lukasiewycz *et al.*, "System architecture and software design for electric vehicles," in *50th Annual Design Automation Conference (DAC)*, 2013.

[5] D. Goswami *et al.*, "Challenges in automotive cyber-physical systems design," in *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation(SAMOS)*, 2012.

[6] M. Broy *et al.*, "Cross-layer analysis, testing and verification of automotive control software," in *11th International Conference on Embedded Software (EMSOFT)*, 2011.

[7] G. Tibba *et al.*, "Testing automotive embedded systems under X-in-the-loop setups," in *35th International Conference on Computer-Aided Design (ICCAD)*, 2016.

[8] P. Axer *et al.*, "Building timing predictable embedded systems," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4, pp. 82:1–82:37, 2014.

[9] J. Oetjens et al., "Safety evaluation of automotive electronics using virtual prototypes: State of the art and research challenges," in *51st Design Automation Conference (DAC)*, 2014.

[10] G. Georgakos *et al.*, "Reliability challenges for electric vehicles: from devices to architecture and systems software," in *50th Annual Design Automation Conference (DAC)*, 2013.

[11] L. Guo, Q. Zhu, P. Nuzzo, R. Passerone, A. L. Sangiovanni-Vincentelli, and E. A. Lee, "Metronomy: A function-architecture co-simulation framework for timing verification of cyber-physical systems," in *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2014.

[12] D. Roy *et al.*, "Timing debugging for cyber-physical systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021.

[13] D. Goswami *et al.*, "Time-triggered implementations of mixed-criticality automotive software," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012.

[14] P. Pazzaglia, Y. Sun, and M. D. Natale, "Generalized weakly hard schedulability analysis for real-time periodic tasks," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 1, 2020.

[15] R. Schneider *et al.*, "Multi-layered scheduling of mixed-criticality cyber-physical systems," *Journal of Systems Architecture - Embedded Systems Design*, vol. 59, no. 10-D, pp. 1215–1230, 2013.

[16] S. Chakraborty *et al.*, "Automotive cyber-physical systems: A tutorial introduction," *IEEE Des. Test*, vol. 33, no. 4, pp. 92–108, 2016.

[17] D. Goswami, R. Schneider, and S. Chakraborty, "Re-engineering cyber-physical control applications for hybrid communication protocols," in *Design, Automation and Test in Europe (DATE)*, 2011.

[18] ——, "Co-design of cyber-physical systems via controllers with flexible delay constraints," in *16th Asia South Pacific Design Automation Conference (ASP-DAC)*, 2011.

[19] W. Chang and S. Chakraborty, "Resource-aware automotive control systems design: A cyber-physical systems approach," *Found. Trends Electron. Des. Autom.*, vol. 10, no. 4, 2016.

[20] C. Hobbs, B. Ghosh, S. Xu, P. S. Duggirala, and S. Chakraborty, "Safety analysis of embedded controllers under implementation platform timing uncertainties," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4016–4027, 2022.

[21] S. Xu *et al.*, "Safety-aware flexible schedule synthesis for cyber-physical systems using weakly-hard constraints," in *28th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023.

[22] D. Roy *et al.*, "Multi-objective co-optimization of FlexRay-based distributed control systems," in *22nd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016.

[23] T. Weber *et al.*, "The SMT competition 2015-2018," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 11, no. 1, 2019.

[24] "The SMT competition," 2023, https://smt-comp.github.io/2023/participants.html.

[25] P. Kumar *et al.*, "A hybrid approach to cyber-physical systems verification," in *Design Automation Conference (DAC)*, 2012.

[26] D. Roy *et al.*, "Timing debugging for cyber-physical systems," in *DATE*, 2021.

[27] L. Zhang *et al.*, "Task- and network-level schedule co-synthesis of Ethernet-based time-triggered systems," in *19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2014.

[28] G. Frehse *et al.*, "Formal analysis of timing effects on closed-loop properties of control software," in *RTSS*, 2014.

[29] ——, "SpaceEx: Scalable verification of hybrid systems," in *CAV*, 2011.

[30] N. Vreman, A. Cervin, and M. Maggio, "Stability and performance analysis of control systems subject to deadline misses," in *ECRTS*, 2021.

[31] M. Maggio *et al.*, "Control system stability under consecutive deadline misses," in *ECRTS*, 2020.

[32] A. Minaeva *et al.*, "Control performance optimization for application integration on automotive architectures," *IEEE Trans. Computers*, vol. 70, no. 7, pp. 1059–1073, 2021.

[33] N. Vreman and C. Mandrioli, "Evaluation of Burst Failure Robustness of Control Systems in the Fog," in *Workshop on Fog-IoT*, ser. OASIcs. Schloss Dagstuhl, 2020.

[34] A. Abate *et al.*, "Automated formal synthesis of digital controllers for state-space physical plants," in *CAV*, 2017.

[35] A. Masrur *et al.*, "VM-based real-time services for automotive control applications," in *16th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2010.

[36] J. Freitag, S. Uhrig, and T. Ungerer, "Virtual timing isolation for mixed-criticality systems," in *30th Euromicro Conference on Real-Time Systems (ECRTS)*, 2018.

[37] S. Chakraborty and L. Thiele, "A new task model for streaming applications and its schedulability analysis," in *Design, Automation and Test in Europe Conference ((DATE)*, 2005.

[38] S. Chakraborty, T. Erlebach, and L. Thiele, "On the complexity of scheduling conditional real-time code," in *7th International Workshop on Algorithms and Data Structures (WADS)*, 2001.

[39] P. Pazzaglia *et al.*, "Beyond the Weakly Hard Model: Cost of Deadline Misses," in *ECRTS*, vol. 106, 2018, pp. 10:1–10:22.

[40] L. Zhang *et al.*, "Real-time attack-recovery for CPS using linear-quadratic regulator," *ACM TECS*, vol. 20, 2021.

[41] D. Goswami, R. Schneider, and S. Chakraborty, "Relaxing signal delay constraints in distributed embedded controllers," *IEEE Trans. Contr. Sys. Techn.*, vol. 22, no. 6, pp. 2337–2345, 2014.

[42] N. Vreman, C. Mandrioli, and C. Anton, "Deadline-miss-adaptive controller implementation for real-time control systems," in *RTAS*, 2022.

[43] X. Dai and A. Burns, "Period adaptation of real-time control tasks with fp scheduling in cyber-physical systems," *Journal of Sys. Arch.*, vol. 103, 2020.

[44] A. Yeolekar *et al.*, "Checking scheduling-induced violations of control safety properties," in *ATVA*, 2022.

[45] "Z3 Solver," 2023, https://github.com/z3prover/z3.git.

[46] M. O'Kelly *et al.*, "F1tenth: An evaluation environment for continuous control and reinforcement learning," in *NeurIPS*, 2019.

[47] K. N. Murphy, in *Analysis of Robotic Vehicle Steering and Controller Delay*, 1994.

[48] "CVC4 Solver," 2023, https://cvc4.github.io/.

[49] "CVC5-website," 2023, https://cvc5.github.io/.

[50] "MathSAT Solver," 2023, https://mathsat.fbk.eu/.

[51] S. Chakraborty, L. T. X. Phan, and P. S. Thiagarajan, "Event count automata: A state-based model for stream processing systems," in *26th IEEE Real-Time Systems Symposium (RTSS)*, 2005.

[52] P. Waszecki *et al.*, "Automotive electrical and electronic architecture security via distributed in-vehicle traffic monitoring," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 36, no. 11, pp. 1790–1803, 2017.

[53] P. Mundhenk *et al.*, "Security analysis of automotive architectures using probabilistic model checking," in *52nd Annual Design Automation Conference (DAC)*, 2015.

[54] ——, "Security in automotive networks: Lightweight authentication and authorization," *ACM Trans. Design Autom. Electr. Syst.*, vol. 22, no. 2, pp. 25:1–25:27, 2017.

[55] L. T. X. Phan *et al.*, "Composing functional and state-based performance models for analyzing heterogeneous real-time systems," in *28th IEEE Real-Time Systems Symposium ((RTSS)*, 2007.

[56] E. Fraccaroli *et al.*, "Timing predictability for SOME/IP-based service-oriented automotive in-vehicle networks," in *Design, Automation & Test in Europe Conference (DATE)*, 2023.