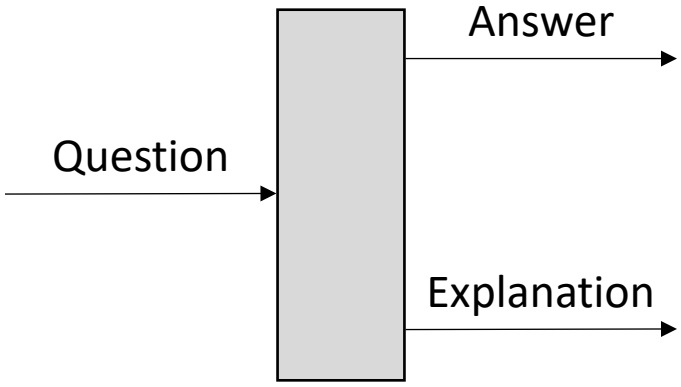
1st International Workshop on Explainability of Real-time Systems and their Analysis

at the IEEE Real-Time Systems Symposium

Houston, Texas, December 5, 2022



Program Chair

Bjorn Andersson, SEI/CMU, USA

Program Committee

Ahlem Mifdaoui, UToulouse, France
Al Mok, UTexas, USA
Anaïs Finzi, TTTech, Austria
Björn Brandenburg, MPI-SWS, Germany
Dakshina Dasari, Bosch, Germany
Hyoseung Kim, UCR, USA
Insup Lee, UPenn, USA
James Anderson, UNC, USA
John Goodenough, SEI/CMU, USA
John Lehoczky, CMU, USA
John Rushby, SRI, USA
Jonathan Preston, LMCO, USA
Mark Klein, SEI/CMU, USA
Raj Rajkumar, CMU, USA
Sanjoy Baruah, WUSTL, USA
Shige Wang, GM, USA
Stefan Mitsch, CMU, USA
Wang Yi, Uppsala, Sweden

It is my pleasure to welcome you to the 1st International Workshop on Explainability of Real-time Systems and their Analysis (ERSA). We started this workshop because we wanted to explore whether the notion of explainability is helpful for the real-time system research community in order to deliver more value to software practitioners—in particular those involved in certification. This document is the workshop proceeding for ERSA.

I thank several individuals without whom this workshop would not have been possible; this includes:

1. The authors for providing technical content;
2. The members of the technical program committee for evaluating the content and providing constructive feedback to the authors;
3. The organizers of RTSS that gave “go-ahead” for ERSA to take place; this includes Dionisio de Niz (Hot-Topics Day Chair of RTSS), Arvind Easwaran (Program Chair of RTSS), and Liliana Cucu-Grosjean (General Chair of RTSS);
4. People working behind the scene to provide (digital and physical) infrastructure, advice, and proof reading.

The papers in this workshop proceeding provide new ideas on explainability. I believe and hope you will find them interesting; and that they will help you and help us all in defining this new area of research.

Sincerely,

Bjorn Andersson
Program chair of ERSA

Certificates of Real-Time Schedulability

Sanjoy Baruah
Washington University in Saint Louis
baruah@wustl.edu

Pontus Ekberg
Uppsala University
pontus.ekberg@it.uu.se

***Abstract*—One method of showing that a hard-real-time system is schedulable is to present a "certificate" of its schedulability — e.g., a (static) schedule which can be verified to always meet all deadlines. We identify some widely-studied real-time systems for which short (i.e., polynomial-sized) certificates of schedulability exist that can be verified in polynomial time, and apply ideas and results from computational complexity theory to identify other systems for which such certificates are unlikely to exist.**

***Index Terms*—Periodic Task Systems; Schedulability; Polynomial-time Verification**

I. Introduction

Cyclic-executive (CE) [1], [2] based approaches have proved to be a successful means of demonstrating that a safety-critical real-time system will meet all its timing constraints. In such approaches, the system developer provides the certification authority (CA) with a lookup table, called the CE, that explicitly enumerates which task will execute at each instant; the CA checks that repeated execution of this table assigns adequate computing to each task to allow all its timing constraints to be met (provided, of course, that no task executes for a duration exceeding its *worst-case execution time* or WCET).

When used in this manner, we can think of the CE as a **certificate** of the schedulability of the system. For periodic task systems [3], [4] the size of the CE is proportional to the hyper-period and may therefore in general be of size exponential in the representation of the task system under consideration. Furthermore, verifying the correctness of such a certificate takes the CA time at least linear in the size of the certificate; i.e., exponential in the representation of the task system.

In this work we adopt a more expansive notion of a certificate than merely an explicit enumeration of a schedule lookup table. As an illustrative problem we consider the preemptive uniprocessor scheduling of synchronous periodic constrained-deadline task systems (see Section II). We discuss alternative certificates of schedulability for such systems, and provide informal assurance arguments as to why these certificates may be considered adequate for the purposes of verifying schedulability. We apply standard results from computational complexity theory in order to identify, in Section III, a particular schedulability analysis problem for which a system developer is able to provide *certificates that can be verified in time polynomial in the size of the representation of the task system*. By defining explainability in terms of the guaranteed existence of such polynomial-time verifiable certificates, we also identify, in Sections IV and V, a pair of schedulability analysis problems that are unlikely to be explainable according to this notion. In Section VI we propose some directions for research upon such a notion of explainability: seeking explainable *sub*problems of problems that are unlikely to be explainable in general.

II. System Model

We consider a synchronous periodic constrained-deadline task system Γ that is to be scheduled upon a single preemptive processor. Each periodic task $\tau_i \in \Gamma$ is characterized by three integer parameters: its worst-case execution time C_i, its relative deadline D_i, and its period T_i; we restrict attention to constrained-deadline systems in which $D_i \leq T_i$. To recap the synchronous periodic task model: each task τ_i releases a job at each time instant $k \times T_i$ for all $k \in \mathbb{N}$; the job released at time-instant $k \times T_i$ has a WCET C_i and a deadline at time-instant $(k \times T_i + D_i)$.

Notation: we will use $H(\Gamma)$ to denote the hyper-period (least common multiple of all the periods) of task system Γ, and $U(\Gamma)$ to denote its utilization, $U(\Gamma) = \sum_{\tau_i \in \Gamma} C_i/T_i$.

Explainability. A system developer that chooses to model their system as a periodic task system for the purposes of obtaining certification would presumably need to justify this choice to the CA. They would, for instance, need to provide some justification for the values they have assigned to the WCET parameters characterizing their tasks[1], and explain why one may model the processor as being preemptive. We will not address this issue of model-justification any further in this note, other than pointing out that safety-critical system design methodologies (e.g., the Rate-Monotonic Analysis methodology [6]) exist that explain how this may be done.

In Section V, we will consider an extended version of this problem in which each task τ_i is additionally characterized by a *best case execution time (BCET)* B_i. This model, if used, must also be justified, and a convincing argument provided as to why it is safe to assume that each job of a task will execute for a duration no smaller than the value that is assigned to the BCET parameter value of the task.

III. Fixed-Priority Scheduling

Fixed-priority (FP) scheduling is a priority-based scheduling scheme in which each task in the system is assigned a unique priority, and at each instant in time the highest-priority

[1]Such justification may, for instance, take the form of stating that the values were obtained using tools [5] that have been certified for this purpose.

task needing execution is executed on the shared processor. *Response-time analysis (RTA)* [7], [8] is the standard technique for determining whether a constrained-deadline synchronous periodic task system is schedulable or not under FP scheduling with given priorities. RTA is based on the observation [7] that if a constrained-deadline task system is schedulable under FP, then the maximum possible duration between the release of a job of τ_i and the instant this job completes execution (called the *worst-case response time* of task τ_i) is equal to the smallest positive value of R_i that satisfies the following recurrence:

$$R_i = C_i + \sum_{\tau_j \in \mathrm{hp}(\tau_i)} \left\lceil \frac{R_i}{T_j} \right\rceil \times C_j \qquad (1)$$

(Here $\mathrm{hp}(\tau_i)$ denotes all jobs in the task system that have scheduling priority greater than τ_i's scheduling priority.)

Explainability. A system developer that chooses to use FP scheduling must first have the CA accept the validity of RTA.[2] Assuming the CA accepts this, the certificate for schedulability for a given task system Γ is a value for R_i for each $\tau_i \in \Gamma$ that satisfies Expression 1 and is $\leqslant D_i$'s. The certificate comprises $|\Gamma|$ numbers, and so is polynomial (in fact linear) in the representation of the task system Γ. It is straightforward to observe that each claimed R_i can be verified to be a solution to Equation 1 in linear time.

IV. EDF Scheduling

Earliest-deadline-first (EDF) scheduling is another priority-based scheduling algorithm. In an EDF-scheduled system, at each instance the currently *active* (i.e., needing execution) job with the earliest deadline is executed — ties may be broken arbitrarily. *Processor-demand analysis (PDA)* [11] is the standard technique for determining whether a synchronous periodic task system is schedulable or not under EDF scheduling. PDA asserts that constrained-deadline synchronous periodic task system Γ is EDF-schedulable if (and only if) the following constraint is satisfied for all $t \in [0, H)$:

$$\left(\sum_{\tau_i \in \Gamma} \max \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1, 0 \right) \times C_i \right) \leqslant t \qquad (2)$$

Explainability. Getting the CA to accept the validity of PDA does not yield a means to generate a certificate for a polynomial-time verification algorithm. Indeed, the existence of such a certificate seems highly unlikely since it would then follow (from the definition of the complexity class NP) that EDF schedulability of constrained-deadline synchronous periodic task systems is in NP. But this schedulability problem has previously been shown [12]–[14] to be coNP-hard, hence its membership in NP would immediately imply that coNP = NP, which runs counter to the expectations of most researchers in computational complexity theory.

[2]This has essentially been achieved: there appears to be wide-spread acceptance by most certification authorities that Equation 1 is indeed correct. Additionally in recent years, this RTA (and much more) has been formally proven correct [9] with machine-verified proofs in the Prosa [10] framework.

V. Best-case Response Times

We have seen that there are polynomial-time verifiable certificates for FP-schedulability but it is unlikely that such certificates exist for EDF-schedulability. In showing [non]-existence of polynomial-time verifiable certificates, the devil is very much in the details of the exact question being asked — we demonstrate this below by showing that a problem, which is very closely related to the FP-schedulability problem, is coNP-hard and therefore does *not* allow for polynomial-time verifiable certificates if NP $\neq$ coNP.

The problem we consider is to establish a lower bound on the best-case response-time (BCRT) of a task under FP-scheduling. This is a practically relevant problem since bounding both the BCRT (from below) and the WCRT (from above) allows us to bound the jitter in task responses. For the BCRT problem to be meaningful we need the task model to also include a best-case execution-time B_i of each task. The hardness of the BCRT problem that we will establish here does not stem from any complicated relationship between the B_i and C_i parameters; in the following we show that the BCRT problem is coNP-hard even if $B_i = C_i$ for all tasks.

The Best-Case Response-Time (BCRT) Problem

INSTANCE: An FP-scheduled synchronous periodic task system Γ with each task $\tau_i \in \Gamma$ additionally characterized by a best-case execution time (BCET) B_i, and a positive integer a.

QUESTION: Does each job of the lowest-priority task in Γ have a response time $\geqslant a$?

We will establish the coNP-hardness of the BCRT problem by relating it to the worst-case response-time (WCRT) problem. Determining FP-schedulability is equivalent to determining whether the WCRT R_i (as in Eq. 1) of each task is no larger than its relative deadline D_i. We find it convenient in our derivation below to use the following utilization-restricted variant of the WCRT problem, which has itself been shown [15] to be NP-complete.

The Worst-Case Response-Time (WCRT) Problem

INSTANCE: An FP-scheduled synchronous periodic task system Γ with $U(\Gamma) \leqslant \ln 2$, and a positive integer a.

QUESTION: Does each job of the lowest-priority task in Γ have a response time $\leqslant a$?

We note that the key difference between the above two problem formulations is that we are asked if the given number a is an upper bound to the possible response times in the WCRT case,

and a lower bound in the BCRT case. We will use a simple trick to reduce from the WCRT problem to the complement of the BCRT problem, thereby showing coNP-hardness for the BCRT problem.

We reduce from the WCRT problem to the BCRT problem by copying the task set Γ of the former problem to a task set Γ' for the new problem, but changing the period of the lowest-priority task τ_{low} in Γ' to equal the hyper-period,

$$T_{\text{low}} = H(\Gamma),$$

and assigning best-case execution times

$$B_i = C_i$$

to all tasks $\tau_i \in \Gamma'$.

The change to τ_{low}'s period effectively means that it will only release the first job in every hyper-period in Γ' compared to Γ. It is well-known that if the first job in the hyper-period has a response-time $\leqslant T_{\text{low}}$, then that job has the maximum response time [16]. Since we have $U(\Gamma) \leqslant \ln 2$, the response-time of the first job must be $\leqslant T_{\text{low}}$ by Liu and Layland's utilization bound [3], and so τ_{low}'s WCRT must be the same in Γ and Γ'. But since τ_{low} only releases a single job per hyper-period in Γ', and since all tasks have $B_i = C_i$, it must also be the case that τ_{low}'s WCRT and BCRT are the same in Γ'. In order to answer the WCRT problem for Γ

> "Does each job of the lowest-priority task in Γ have a response time $\leqslant a$?"

we can simply answer the BCRT problem for Γ'

> "Does each job of the lowest-priority task in Γ' have a response time $\geqslant a + 1$?"

and negate the answer. It follows that the BCRT problem is coNP-hard and therefore does not allow polynomial-time verifiable certificates if NP $\neq$ coNP.

As an immediate corollary, we may conclude that the following problem for bounding the response time within an interval is both NP-hard and coNP-hard, and therefore is unlikely to have either polynomial-time verifiable certificates or counterexamples.

The Response-Time Jitter Problem

INSTANCE: An FP-scheduled synchronous periodic task system Γ with each task $\tau_i \in \Gamma$ additionally characterized by a best-case execution time (BCET) B_i, and positive integers a, b.

QUESTION: Does each job of the lowest-priority task in Γ have a response time in interval $[a, b]$?

Explainability. Despite being so closely related to the WCRT problem, and hence the FP-schedulability problem, we have seen that there are likely no polynomial-time verifiable certificates for the BCRT problem (if there are, then NP = coNP).

We consider the coNP-hardness of the BCRT problem an interesting result in itself; this section additionally demonstrates how computational complexity may change with only small variations of the questions asked, and how this can determine the existence of efficiently-verifiable certificates.

VI. Summary & Discussion

One effective means of "explaining" that a system is schedulable has been by presenting verifiable certificates of its schedulability, as is evidenced by the prevalence of cyclic-executive based scheduling approaches in important safety-critical application domains such as avionics. A more general notion of certificate than the explicit schedule as provided by cyclic executives is some more abstract *proof* of schedulability that can be independently verified by, say, a certification authority. A very formal (and very interesting) approach to this are the machine-checkable certificates generated by the foundational response-time analysis of Maida et al. [17].

The notion of explainability that we focus upon in this note is this: *is a certificate guaranteed to exist for all schedulable task systems, that can be verified in time polynomial in the representation of the task system whose schedulability is to be verified?* Under this interpretation, explainability implies membership in the computational complexity class NP; and as a contrapositive, if a schedulability analysis problem is $\notin$ NP, then the problem is not explainable in general – it is not the case that all instances have polynomial-sized certificates. Showing, as we have done in this note, that a problem is coNP-hard offers very strong evidence that it is $\notin$ NP, since we otherwise would have NP = coNP.

We have demonstrated this equivalence between computational complexity and this notion of explainability via examples upon some commonly-studied preemptive uniprocessor schedulability analysis problems. Under FP scheduling, we noted that upper bounds on worst-case response time are explainable, but showed that lower bounds on best-case response times are likely not. For EDF scheduling, we noted that determining schedulability is unlikely to be explainable.

We can of course apply this thinking to other complexity results in real-time scheduling theory. To exemplify with a few complexity results for some *multiprocessor* schedulability analysis problems:

- Partitioned FP scheduling of constrained-deadline synchronous periodic task systems is in NP, this follows for example from an ILP formulation by Zheng et al. [18]. (In fact it is NP-complete even for unrelated heterogeneous processors, see [19, Sec. VII] for an overview on the complexity of partitioned schedulability problems.) Hence partitioned FP scheduling of constrained-deadline synchronous periodic task systems is explainable.
- In contrast, it is currently *unknown* if the global FP-schedulability problem of constrained-deadline synchronous periodic task systems has polynomial-sized certificates. This problem is NP-hard (which follows directly from the hardness of the single-processor case [15]), but to the best of our

knowledge no better lower bounds on its complexity are known, and it is not known to be in NP.

- Multiprocessor schedulability for a single *conditional DAG (C-DAG)* [20]–[22] under restricted processor assignment is unlikely to be explainable, since this schedulability analysis problem is known [23] to be PSPACE-complete (and since it is generally believed that NP $\neq$ PSPACE).

Directions for future research. Although this note (and indeed, this workshop) deals with explainability, the traditional focus of real-time scheduling theory research has primarily been on devising efficient algorithms for determining, rather than explaining, schedulability. In this traditional context if a schedulability analysis problem is shown to be computationally hard, one approach has been to try to identify sub-problems that are solvable in polynomial time (for example, FP and EDF schedulability analysis of *harmonic* task systems may be looked upon as such sub-problems of the schedulability analysis problems considered in Sections III and IV, for which exact polynomial-time schedulability tests are known [24], [25]).

This flavor of prior research suggests a promising future research direction on the approach to explainability that we have investigated in this note: If some schedulability analysis problem that arises frequently in practice is $\notin$ NP and therefore unlikely to have polynomial-sized certificates, there may be sub-problems of it that are in NP and hence possess polynomially-verifiable certificates of schedulability. Since P $\subseteq$ NP, there is an obvious possibility that there are more or larger practically relevant sub-problems of this type than there are sub-problems that are efficiently solvable. In other words, if explainability is a main concern it may be meaningful to search for practically relevant sub-problems not only for their efficient solvability, but also for their explainability

References

[1] T. P. Baker and A. Shaw. The cyclic executive model and Ada. In *Proceedings of the 9th Real-Time Systems Symposium (RTSS)*, pages 120–129, 1988.

[2] T. P. Baker and A. Shaw. The cyclic executive model and Ada. *Real-Time Systems*, 1(1):7–25, 1989.

[3] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[4] Joseph Y.-T Leung and M. Merrill. A note on the preemptive scheduling of periodic, real-time tasks. *Information Processing Letters*, 11:115–118, 1980.

[5] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. The worst-case execution-time problem – overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems*, 7(3):36:1–36:53, May 2008.

[6] Mark H. Klein, Thomas Ralya, Bill Pollak, Ray Obenza, and Michael González Harbour. *A Practitioner's Handbook for Real-time Analysis – Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[7] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, October 1986.

[8] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the 10th Real-Time Systems Symposium (RTSS)*, pages 166–171, Santa Monica, California, USA, December 1989. IEEE Computer Society Press.

[9] Sergey Bozhko and Björn B. Brandenburg. Abstract Response-Time Analysis: A Formal Foundation for the Busy-Window Principle. In Marcus Völp, editor, *Proceedings of the 32nd Euromicro Conference on Real-Time Systems (ECRTS)*, volume 165 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:24, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[10] Felipe Cerqueira, Felix Stutz, and Björn B. Brandenburg. PROSA: A case for readable mechanized schedulability analysis. In *Proceedings of the 28th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 273–284, 2016.

[11] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium (RTSS)*, pages 182–190, Orlando, Florida, 1990. IEEE Computer Society Press.

[12] Friedrich Eisenbrand and Thomas Rothvoß. EDF-schedulability of synchronous periodic task systems is coNP-hard. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2010.

[13] P. Ekberg and W. Yi. Uniprocessor feasibility of sporadic tasks with constrained deadlines is strongly coNP-complete. In *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 281–286, 2015.

[14] P. Ekberg and W. Yi. Uniprocessor feasibility of sporadic tasks remains coNP-complete under bounded utilization. In *Proceedings of the 36th Real-Time Systems Symposium (RTSS)*, pages 87–95, 2015.

[15] Pontus Ekberg and Wang Yi. Fixed-priority schedulability of sporadic tasks on uniprocessors is NP-hard. In *Proceedings of the 38th Real-Time Systems Symposium (RTSS)*, pages 139–146. IEEE Computer Society, 2017.

[16] J.P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of the 11th Real-Time Systems Symposium (RTSS)*, pages 201–209, 1990.

[17] Marco Maida, Sergey Bozhko, and Björn B. Brandenburg. Foundational Response-Time Analysis as Explainable Evidence of Timeliness. In Martina Maggio, editor, *Proceedings of the 34th Euromicro Conference on Real-Time Systems (ECRTS)*, volume 231 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:25, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[18] Wei Zheng, Qi Zhu, Marco Di Natale, and Alberto Sangiovanni Vincentelli. Definition of task allocation and priority assignment in hard real-time distributed systems. In *Proceedings of the 28th Real-Time Systems Symposium (RTSS)*, RTSS '07, page 161–170, USA, 2007. IEEE Computer Society.

[19] Pontus Ekberg and Sanjoy Baruah. Partitioned scheduling of recurrent real-time tasks. In *Proceedings of the 42nd Real-Time Systems Symposium (RTSS)*, pages 356–367, 2021.

[20] Jose Fonseca, Vincent Nelis, Gurulingesh Raravi, and Luis Miguel Pinho. A Multi-DAG model for real-time parallel applications with conditional execution. In *Proceedings of the ACM/ SIGAPP Symposium on Applied Computing (SAC)*, Salamanca, Spain, April 2015. ACM Press.

[21] Sanjoy Baruah, Vincenzo Bonifaci, and Alberto Marchetti-Spaccamela. The global EDF scheduling of systems of conditional sporadic DAG tasks. In *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS)*, ECRTS '15, pages 222–231, Lund (Sweden), 2015. IEEE Computer Society Press.

[22] Alessandra Melani, Marko Bertogna, Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, and Giorgio Buttazzo. Response-time analysis of conditional DAG tasks in multiprocessor systems. In *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS)*, ECRTS '15, pages 222–231, Lund (Sweden), 2015. IEEE Computer Society Press.

[23] Sanjoy Baruah and Alberto Marchetti-Spaccamela. Feasibility Analysis of Conditional DAG Tasks. In Björn B. Brandenburg, editor, *Proceedings of the 33rd Euromicro Conference on Real-Time Systems (ECRTS)*, volume 196 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[24] V. Bonifaci, A. Marchetti-Spaccamela, N. Megow, and A. Wiese. Polynomial-time exact schedulability tests for harmonic real-time tasks. In *Proceedings of the 34th Real-Time Systems Symposium (RTSS)*, pages 236–245, Dec 2013.

[25] Thi Huyen Chau Nguyen, Werner Grass, and Klaus Jansen. Exact polynomial time algorithm for the response time analysis of harmonic tasks. In Cristina Bazgan and Henning Fernau, editors, *Combinatorial Algorithms*, pages 451–465, Cham, 2022. Springer International Publishing.

Understanding Safety of Linear Real-Time Systems from Lyapunov Theory and Quadratic Boundedness.

Raffaele Romagnoli

***Abstract*— In this note Lyapunov's stability theory and quadratic boundedness are presented to understand the safety of real-time control systems with missing deadlines. By modeling missing deadlines as bounded external disturbances, quadratic boundedness can be applied. This formulation can also model the presence of uncertainties in the computed control action.**

I. Introduction

Lyapunov's stability theory for dynamical systems is based on the existence of a non-negative and non-increasing function along the trajectories of the system. This function is called Lyapunov's function V [1]. The closest concept to explain V is the energy E of a system [2] (e.g. mechanical, electrical, etc.). Considering the system starting from an initial condition which is associated with an initial level of energy c, the system is *stable* if the energy is not increasing while the system is evolving during the time $E \leq c$. In general, if E decreases, the system is going to stop at a point where $E = 0$. In this case, the system is *asymptotically stable*. A classical example is a pendulum with friction. The amplitude of the oscillations around the downright position reduces over time due to the effect of friction. The position where the pendulum starts has an initial energy $E = c$ that will never be reached again since the energy decreases, this means that the pendulum can reach only the trajectories where $E \leq c$. The set of all the trajectories where $E \leq c$ is called *positively invariant set*. For example, safety can be formulated as a set where any trajectory of the system can be driven to the resting point ($E = 0$). A positively invariant set is suitable to approximate the safety set since it offers the property that any trajectory inside the set will never go outside of it. Hence, safety can be ensured. Assuming that there is an external disturbance that acts as a force on the pendulum then, the energy of the entire system can increase. Hence, the positively invariant property for the set $E \leq c$ is no longer satisfied since the energy may increase and the trajectories of the system may go outside of that set. In this way, safety cannot be ensured anymore. However, if the disturbance is bounded then its effects on the system are limited. If the system starts from a level of energy enough to reject the effects of the disturbance then it defines a positively invariant set. This explains the concept of *quadratic boundedness* [3], [4].

This paper aims to present Lyapunov's theory and quadratic boundedness and their connections with safety in a more general way than the notion of energy used before. Specifically, this paper shows how to represent the missing deadlines in real-time control systems as an external disturbance. If this disturbance is bounded then quadratic boundedness can be used to find an approximation of the safety set.

In general control theory only considers the physical system for the controller design without taking into account the implementation aspects. The controller is implemented as an algorithm that exchanges and processes information in the discrete-time domain. In general, the control algorithm is much faster than the inertia of the system, hence the digital implementation of the controller does not have a serious effect on the properties of the designed controller.

In more complex systems such as cyber-physical systems (CPSs) [5], the digital implementation of the controller can affect its properties. In fact, in a CPS the controller can be distributed over the network, and the packet loss may have negative consequences on the networked control system. Not only is missing updating the control input a problem for networked control systems, but it is also an issue for real-time control systems. In [6], weakly hard real-time (WHRT) constraints are used to model both packet loss and the missing deadline for linear control systems. This work considers the stabilization of linear sampled-data systems under WHRT constraints. The extension to nonlinear systems is in [7]. Another work oriented on the stability analysis for linear systems under extended weakly hard constraints is presented in [8].

A state feedback controller is used for the general problem of the stability of control systems under WHRT constraints, and the system's current state is perfectly known. In general, this assumption may be difficult to satisfy in a real application. For example, the measurement noise generates uncertainties. In CPSs the state of the system can be elaborated by different software modules that implement a state estimator that considers remote sensors. These are all possible sources of uncertainties that in the end will affect the control input. This work also shows how quadratic boundedness can consider in the problem of missing deadlines and the uncertainties in the computed control inputs.

II. Preliminaries

The behavior of a physical system can be captured by ordinary differential equations (ODEs) of the form

$$\dot{x} = f(x) \tag{1}$$

R. Romagnoli is with the Dept. of Electrical and Computer Engineering, Carnegie Mellon University (CMU), Pittsburgh, PA, USA 15235. Email: {rromagno}@andrew.cmu.edu

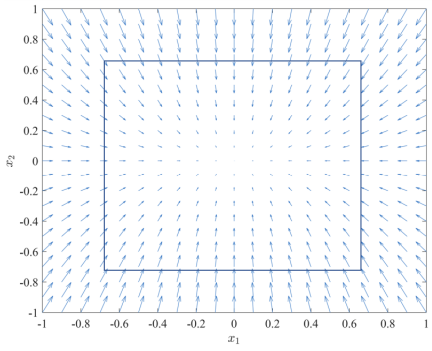


Fig. 1. Example of a vector field $f(x)$ with $x \in \mathbb{R}^2$. The square set is positively invariant where the state of the system can enter from outside but it will never leave it.

where $x \in \mathbb{R}^n$ is the state vector where each component represents a specific physical variable that evolves during the time. The function $f : D \to \mathbb{R}^n$ is locally Lipschitz on $D \subset \mathbb{R}^n$, and from a given initial condition $x_0 \in D$, the solution of (1) is indicated with $x(t)$ where $x : I \to \mathbb{R}^n$ and $I \subseteq \mathbb{R}$. If the system is linear then

$$\dot{x} = f(x) = Ax \tag{2}$$

where $A \in \mathbb{R}^{n \times n}$, and the solution $x(t)$ is defined for all $t \geq 0$. A more general formulation of (1) is $\dot{x} = f(x, u)$ where $u \in \mathbb{R}^p$ is the input vector. Considering a state feedback control input of the form $u = \gamma(x)$, the general system can be represented again as in (1). In the case of a linear system

$$\dot{x} = Ax + Bu, \tag{3}$$

a state feedback controller can assume the form of $u = -Kx$, where $K \in \mathbb{R}^{p \times n}$. Defining $A_f \triangleq (A - BK)$, the closed loop system is $\dot{x} = A_f x$ which is expressed in the same form of (2).

The stability analysis of (1) considers the behavior of the system around an equilibrium point which is a point $x_{eq} \in D$ such that $f(x_{eq}) = 0$. Without loss of generality it can be considered the origin of the state space $x_{eq} = 0$[1]. The function $f(x)$ is a vector field, where each point $x \in D$ has associated a vector that represents the intensity and the direction of the velocity of the solution of (1) at that specific point. Assuming that $x \in \mathbb{R}^2$, the vector field associated to the system (1) can be represented as in Fig. 1. Observing the direction of the arrows that represent the vector field $f(x)$, one can say that any solution starting from any point in the domain D is approaching the origin. Moreover, considering the square set $\mathcal{S}$, all the trajectories starting inside $\mathcal{S}$ will stay in $\mathcal{S}$. Also, all the trajectories starting outside $\mathcal{S}$ will enter in $\mathcal{S}$. Considering the boundary of $\mathcal{S}$, the entire vector field is pointing inside $\mathcal{S}$. Hence, $\mathcal{S}$ is a *positively invariant* set.

For the dynamical system (1), *safety* can be defined as a region of the state space $\mathcal{C} \subseteq D$ where the state $x \in \mathcal{C}$ satisfies specific requirements. To ensure safety $\mathcal{C}$ has to be an *invariant* namely, no solutions starting in $\mathcal{C}$ leave $\mathcal{C}$ for any $t \in \mathbb{R}$. In Fig. 1, the square $\mathcal{S}$ which is positively invariant can represent a good approximation of the safety set, if $\mathcal{S} \subseteq \mathcal{C}$.

III. Lyapunov Stability Theory and Safety

Lyapunov stability theory [2] provides a way to analyze the stability of an equilibrium point in the general case where $x \in \mathbb{R}^n$. The idea is represented in Fig. 1. Given a positive

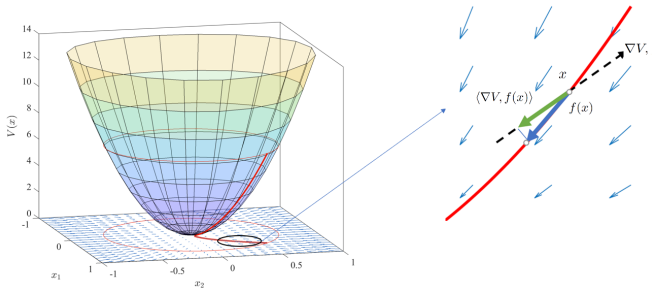


Fig. 2. A possible trajectory of the system is moving accordingly to the vector field $f(x)$ (red line). If the projection of $f(x)$ along the gradient of $V(x)$ is negative then, $V(x)$ decreasing along any trajectory $x(t)$. $V(x)$ generalizes the system's energy concept.

definite function $V : D \to \mathbb{R}$, where $V(0) = 0$ and $V(x) > 0$ for $x \neq 0$, the idea is to project the vector field $f(x)$ along the gradient of V:

$$\langle \nabla V, f(x) \rangle = \nabla V^T f(x) = \frac{\partial V}{\partial x} f(x) = \dot{V}(x) \tag{4}$$

where

$$\frac{\partial V}{\partial x} = \begin{bmatrix} \frac{\partial V}{\partial x_1} & \cdots & \frac{\partial V}{\partial x_n} \end{bmatrix} = \nabla V^T.$$

If (4) is non-positive ($\dot{V}(x) \leq 0$ for all $x \in D$) then $x = 0$ is *stable*. Instead, if $\dot{V}(x)$ is strictly negative, $x = 0$ is *asymptotically stable*. In both cases, given an initial point $x(0) = x_0 \in D$, the value of $V(x) \leq V(x_0) = c$. Defining

$$\Omega_c = \{x \in D \mid V(x) \leq c\} \tag{5}$$

any solution starting in Ω_c remains in Ω_c, since $V(x)$ cannot increase. The set (5) is positively invariant w.r.t the solution of (1), and it is called *Lyapunov level set*.

Given a safety set $\mathcal{C}$, a Lyapunov level set Ω_c can be used to guarantee safety if $\Omega_c \subseteq \mathcal{C}$, and $x_0 \in \Omega_c$.

For an asymptotically stable linear system (2), there exists a symmetric positive-definite matrix P that solves Lyapunov's equation

$$A^T P + PA = -Q \tag{6}$$

where Q can be any given symmetric positive definite matrix. The function

$$V(x) = x^T P x \tag{7}$$

is a Lyapunov function for (2) [2]. The Lyapunov level set Ω_c

[1]Any equilibrium point can be translated to the origin [2]. If the equilibrium point changes during the time, the analysis remains the same.

[2]The Lyapunov function (7) is a quadratic form, and each Lyapunov level set is an ellipsoid that is easy to represent and visualize.

is an ellipsoid centered at $x = 0$

$$\mathcal{E}_c = \left\{ x \in D \mid x^T P x \leq c \right\}. \tag{8}$$

In the case of linear systems, the safety set can be approximated with a positively invariant ellipsoidal set $\mathcal{E}_c \subseteq \mathcal{C}$, where the safety can be ensured $\forall x_0 \in \mathcal{E}_c$.

Remark 1: Lyapunov's analysis transforms a multidimensional-state problem into a scalar one. For example, the evaluation of $V(t)$ during experiments or simulations can show how far the system can be from the unsafe region [9].

IV. Quadratic boundedness

Let us consider a slightly different version of (3)

$$\dot{x} = Ax + Dd, \quad \|d\| \leq \delta \tag{9}$$

where $D \in \mathbb{R}^{n \times r}$, and $d \in \mathbb{R}^r$ is a bounded disturbance that may represent uncertainties, external disturbances, etc. The nominal model (2) is assumed to be asymptotically stable, hence there exists a symmetric definite-positive matrix P that satisfies (6) for a given symmetric definite-positive matrix Q, and (7) is the associated Lyapunov function.

Here is the following question, is still it possible to use Ω_c to approximate the safety set $\mathcal{C}$? Assuming, for example, that the system is at $x = 0$ then, the nominal system does not move because $\dot{x} = 0$. Instead, the disturbance pushes $x(t)$ away from $x = 0$ because $\dot{x} = Dd \neq 0$. This means that Ω_c computed for the nominal system cannot be positively invariant anymore, and $\Omega_c \subseteq \mathcal{C}$ for all $x_0 \in \Omega_c$ may not guarantee safety. This problem can be addressed by considering *quadratic boundedness*.

Definition 1:[4] The system (9) is quadratically bounded if there exists a scalar $\eta > 0$ and a symmetric positive-definite matrix P such that

$$x^T P x > \eta \rightarrow x^T P(Ax + Dd) < 0 \quad \forall \|d\| \leq \delta \tag{10}$$

Assuming that P is computed from the nominal system, and considering (6), (11) can be reformulated as

$$V(x) > \eta \rightarrow \dot{V}(x) < 0 \quad \forall \|d\| \leq \delta \tag{11}$$

where $\dot{V} = \nabla V^T (Ax + Dd)$. This means that for any $\eta' > \eta$, the set

$$\mathcal{E}_{\eta'} = \left\{ x \in D \mid x^T P x \leq \eta' \right\}. \tag{12}$$

is *robustly positively invariant* for (9). For all $x_0 \in \mathcal{E}_{\eta'}$ and outside of $\mathcal{E}_\eta$, any solution $x(t)$ is approaching $\mathcal{E}_\eta$ since $V(x)$ is strictly decreasing.

For a bounded disturbance d, and a given symmetric positive matrix P computed for the nominal system (2) that satisfies (6), it is possible to compute η such that (11) is satisfied [10]. If there exists $\eta' > \eta$ such that $\mathcal{E}_{\eta'} \subseteq \mathcal{C}$ then, for all $x_0 \in \mathcal{E}_{\eta'}$, safety is ensured.

V. Sampled-Data systems

The Lyapunov theory and quadratic boundedness presented in the previous sections refer to the continuous time case which in general represents the time domain where the physical system evolves end this is fundamental for CPSs. To analyze control systems with real-time constraints, the sampled data version of the continuous-time system (1) is considered

$$x_{k+1} = f_d(x_k) \tag{13}$$

The Lyapunov stability theory is based on the same idea but instead of the derivative of the Lyapunov function $V(x)$, the following difference [1] is evaluated for all $k \geq 0$

$$V(x_k) - V(x_{k-1}) \leq 0 \text{ or } V(x_k) - V(x_{k-1}) < 0 \tag{14}$$

A linear discrete-time system is described by

$$x_{k+1} = A_d x_k, \tag{15}$$

where A_d is the discretized version of the matrix A in (2). If (15) is asymptotically stable then there exists a symmetric positive definite matrix P that satisfies

$$A^T P A - P = -Q \tag{16}$$

where Q is a given symmetric positive-definite matrix. The function $V(x) = x^T P x$ is a Lyapunov function for (15). Since the Lyapunov function for the discrete-time is the same as the continuous-time case, a Lyapunov level set $\mathcal{E}_c$ is an ellipsoid defined as in (8).

The principles of the quadratic boundedness do not change concerning the continuous-time case, the difference is that the condition changes because now the system is in discrete-time:

$$x_{k+1} = A_d x_k + D d_k \quad \|d\| \leq \delta \tag{17}$$

Definition 2:[11] System (17) is quadratically bounded if there exists a scalar $\eta > 0$ and a symmetric positive definite matrix P such that

$$\begin{aligned} &x_k^T P x_k > \eta \quad \rightarrow \\ &(A_d x_k + Dd)^T P (A_d x_k + D d_k) - x_k^T P x_k < 0, \end{aligned} \tag{18}$$

$\forall \|d\| \leq \delta$, and $\forall k \geq 0$.
Similarly to the continuous-time case, P is computed solving (16) for the nominal system, (18) can be reformulated as

$$V(x) > \eta \rightarrow V(x_k) - V(x_{k-1}) < 0, \tag{19}$$

$\forall \|d\| \leq \delta$, and $\forall k \geq 0$.

A. Safety

Note that, ensuring safety for the sampled-data system may not be sufficient to guarantee the safety of the real physical system in the continuous-time domain [12]. In this note, it is assumed that there is a safety set for the sampled-data system $\mathcal{C}_d \subset \mathcal{C}$ which ensures the safety of the continuous-time system. At this point, all the considerations on the safety for the continuous-time system can be used for the discrete-time system by considering $\mathcal{C}_d$ as the safety set.

B. Linear Weakly-Hard Real-Time Control Systems

In this subsection, a model of a linear control system under weakly hard real-time constraints is presented. This model is inspired by [6]:

$$x_{k+1} = A_d x_k + B_d u_k^a, \tag{20}$$

for all $k \geq 0$ with $k \in \mathbb{N}$. The matrices A_d and B_d are the sampled data version of the matrices in (3). A static state feedback controller that makes (20) asymptotically stable is

$$u_k^c = -Kx_k. \tag{21}$$

The strategies to model the effects of missing deadlines to compute (21) at time k is to provide $u_k^a = 0$ or holding the last provided control input $u_k^a = u_{k-1}^a$. For the latter case, let δ_k be a variable which is 1 if u_k^c is successfully computed at time k and provided as input to the system, or 0 in case of failure.

$$u_k^a = \delta_k u_k^c + (1-\delta_k)u_{k-1}^a \tag{22}$$

for all $k \geq 0$ with $k \in \mathbb{N}$, and for some arbitrary initial condition $u_{-1}^a \in \mathbb{R}^p$. The closed-loop system is now given by

$$x_{k+1} = A_d x_k + B_d\left(\delta_k u_k^c + (1-\delta_k)u_{k-1}^a\right) \tag{23}$$

To get the same problem formulation for the quadratic boundedness (18), the above equation can be rewritten as

$$x_{k+1} = (A_d - B_d K)\, x_k + B_d(1-\delta_k)\left(u_{k-1}^a - u_k^c\right). \tag{24}$$

where the nominal system $x_{k+1} = (A_d - B_d K)x_k$ is asymptotically stable, hence there exists a symmetric positive definite matrix P that solves (16), and $V(x) = x^T P x$ is the associated Lyapunov function. The vector

$$(1-\delta_k)(u_{k-1}^a - u_k^c) = K(x_k - x_{k-1}^a)(1-\delta_k)$$

where x_{k-1}^a represents the last state feedback used to provide the previous control input. Hence,

$$x_{k+1} = (A_d - B_d K)\, x_k + D_d d_k \tag{25}$$

where $D_d = B_d K$, and

$$d_k \triangleq (x_k - x_{k-1}^a)(1-\delta_k) \tag{26}$$

From (26)

$$\|d_k\| \leq \|x_k - x_{k-1}^a\|, \tag{27}$$

if $\|d_k\| \leq \delta$ where δ is a finite and positive number then, the quadratic boundedness theory can be applied to (25) for assessing the safety of the real-time control system with missing deadlines. This note does not address the problem of how to find δ, but it shows that the proposed formulation can also consider uncertainties in the control input. In fact, in a real system, the exact knowledge of the state x_k is very difficult since the presence of the measurement noise, hence filters are deployed to obtain a better state estimation. This means that the actual control input is $u_k^c = -K(x_k + e_k)$ where e_k represents the estimation error and x_k is the exact state of the system:

$$\begin{aligned} x_{k+1} \;&=\; (A_d - B_d K)x_k + \\ &\quad B_d K\left[(1-\delta_k)(x_k + e_k - x_{k-1}^a) - e_k\right], \end{aligned} \tag{28}$$

hence,

$$\begin{aligned} &\|(1-\delta_k)(x_k + e_k - x_{k-1}^a) - e_k\| \\ &\qquad \leq \|(x_k - x_{k-1}^a\| + 2\|e_k\| \leq \delta + 2\delta_e \triangleq \bar{\delta}. \end{aligned} \tag{29}$$

The sampled-data system under weak hard real-time constraints and uncertainties on the current state x_k can be formulated as $x_{k+1} = (A_d - B_d K)\, x_k + D_d \bar{d}_k$ where $\bar{d}_k = (1-\delta_k)(x_k + e_k - x_{k-1}^a) - e_k$. If $\delta, \delta_e > 0$ exists, the safety can be addressed by quadratic boundedness analysis. The error e_k can also model the uncertainties that can be generated in CPS. The current state x_k is generated by other software modules that implement more sophisticated state estimators that fuse information from different sensors that can be distributed over the network. Those modules may be also subject to delays or missing deadlines, which contribute to increased uncertainty on x_k. The quadratic boundedness formulation can take into account those aspects, providing a more complete and general tool to assess safety for complex systems.

VI. Conclusions

This note presents the classical tools of Lyapunov theory to assess safety for complex systems such as CPSs. In particular quadratic boundedness can be used for the analysis of safety for sampled-data systems under weak hard real-time constraints. The theory presented here is quite general and can be used for many different application domains. For future works, one of the main challenges is to extract the main features of a particular application to build a Lyapunov-like function that is more application-oriented.

References

[1] R. Kalman and J. Bertram, "Control system analysis and design via the second method of lyapunov:(i) continuous-time systems (ii) discrete time systems," *IRE Transactions on Automatic Control*, vol. 4, no. 3, pp. 112–112, 1959.

[2] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002.

[3] M. L. Brockman and M. Corless, "Quadratic boundedness of nonlinear dynamical systems," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, vol. 1, pp. 504–509, IEEE, 1995.

[4] M. L. Brockman and M. Corless, "Quadratic boundedness of nominally linear systems," *International Journal of Control*, vol. 71, no. 6, pp. 1105–1117, 1998.

[5] A. Platzer, *Logical foundations of cyber-physical systems*, vol. 662. Springer, 2018.

[6] S. Linsenmayer, M. Hertneck, and F. Allgöwer, "Linear weakly hard real-time control systems: Time-and event-triggered stabilization," *IEEE Transactions on Automatic Control*, vol. 66, no. 4, pp. 1932–1939, 2020.

[7] M. Hertneck, S. Linsenmayer, and F. Allgöwer, "Stability analysis for nonlinear weakly hard real-time control systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2594–2599, 2020.

[8] N. Vreman, P. Pazzaglia, V. Magron, J. Wang, and M. Maggio, "Stability of linear systems under extended weakly-hard constraints," *IEEE Control Systems Letters*, 2022.

[9] R. Romagnoli, B. H. Krogh, d. N. Dionisio, H. Anton D., and B. Sinopoli, "Software rejuvenation for safe operation of cyber-physical systems in the presence of run-time cyber attacks," *IEEE Transaction on Control Systems Technology*, 2022. under review.

[10] R. Romagnoli, B. H. Krogh, and B. Sinopoli, "Robust software rejuvenation for CPS with state estimation and disturbances," in *2020 American Control Conference (ACC)*, pp. 1241–1246, IEEE, 2020.

[11] A. Alessandri, M. Baglietto, and G. Battistelli, "On estimation error bounds for receding-horizon filters using quadratic boundedness," *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1350–1355, 2004.

[12] R. Findeisen, T. Raff, and F. Allgöwer, "Sampled-data nonlinear model predictive control for constrained continuous time systems," in *Advanced strategies in control systems with input and output constraints*, pp. 207–235, Springer, 2007.

Formal Artifacts as Explanations for System Correctness in Cyber-Physical Systems

Stefan Mitsch
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
Email: smitsch@cs.cmu.edu

***Abstract*—Autonomous systems, such as self-driving cars and robots, are increasingly often entrusted with operating in safety-critical ways, which makes certification an important tool to ensure that systems are operating as intended. This paper argues that formal methods are a useful basis for certification not only in terms of their rigor, but also as a way of explaining evidence of system correctness at the right level of detail.**

***Index Terms*—formal verification, theorem proving, hybrid systems, refinement, runtime verification, explanations**

I. Introduction

Semi-autonomous and fully autonomous systems—such as driver assistance systems, self-driving cars, aircraft, and robots—are increasingly often entrusted with operating in safety-critical ways. This makes certification an important tool to scrutinize systems before they are deployed and ensure that they are operating as intended. Obtaining evidence of system correctness, however, is not straightforward: assurance cases in combination with testing the system itself is understood well and produces well-understood statistical estimates of the risks of safety violation, but in autonomous systems it requires unrealistically large amounts of test runs to achieve the required confidence [1]. Formal verification of symbolic models, in contrast, provides guarantees for infinitely many situations, but only some methods provably guarantee that the analyzed models are sufficiently accurate [2]. Reliably predicting the behavior of a system and understanding its assurance arguments becomes even more challenging when machine learning components are operating parts of the system [3]. Recently, using formal methods to explain the behavior of learned agents has sparked research interest, e.g., [4], [5].

Even though formal verification is a suggested technique in some certification standards (e.g., the formal methods supplement to DO-178C in avionics), it is still an open question how to best present and explain formal artifacts. In this paper, we argue that *(i)* deductive verification and proofs provide various aspects of direct evidence about system correctness, and that *(ii)* formal techniques are a useful basis for explaining such evidence to a diverse audience with their different situational needs in understanding systems at the right level of detail [5].

II. Models as Explanations

Computing systems that reach out into the real world with actuators to affect physical processes can be understood mathematically as hybrid systems and hybrid games [6], which describe computations plus their physical effects in a uniform modeling language. Deductive verification tools like KeYmaera X [7] implement differential dynamic (game) logic [6] to rigorously analyze the behavior of hybrid systems and games models for safety and liveness properties, for invariants that are preserved throughout system execution, as well as for winning strategies in game settings.

With their unambiguous semantics, formal models can themselves be useful sources for explanation in the following ways: *(i)* the structure of modular formal hybrid systems models with clearly identified assumptions and guarantees may help in understanding the dynamic interaction between the components of a system; *(ii)* formal hybrid systems models can be expressed at varying levels of abstraction, navigating these levels may help in gaining an understanding of system behavior at an abstract level before understanding the details of the full system; and *(iii)* the predictive power of formal hybrid systems models may help in understanding the link between model and true system. We discuss details for each of these three aspects in the subsections below.

A. Decomposition

Component-based verification approaches [8], [9] help manage proof complexity by allowing users to decompose a system into separate components with local responsibilities. In that process, we identify contracts in terms of assumptions that components make about their environment and verified guarantees about their outputs under these assumptions. In hybrid systems models, the physical dynamics of components is expressed with differential equations, and for contracts additionally summarized using first-order real-arithmetic ($\text{FOL}_\mathbb{R}$) abstractions. We construct $\text{FOL}_\mathbb{R}$ abstractions from the hybrid systems models when conducting a formal proof, rather than purely modeling them as requirements. Still, when creating architecture models, there is significant freedom in how responsibility is shared between components: the proofs linking differential equations with their $\text{FOL}_\mathbb{R}$ abstractions can serve as explanations for archictural choices.

Additionally, the interaction between components in a cyber-physical system takes on a variety of different forms, such as direct communication, sensing, or physical manipulation, all with their own assumptions and guarantees (e.g., about sensor uncertainty). When composing component proofs to a full system proof, the composition arguments about the compatibility between assumptions and guarantees may serve as explanations about the dynamic interaction between the components of a system.

For example, in [10] an ego-agent controller safely navigates among mobile environment agents. The discrete abstraction of the environment agent motion describes a region of positions that is reachable from the current position in some bounded time. Using such an abstraction, we separate the local responsibilities of the ego-agent controller and the mobile environment agents with contracts. The components become connected through sensors that obtain position measurements.

B. Model Refinement and Instantiation

Mathematical arguments about cyber-physical systems are often more crisply formulated at a higher level of abstraction and in formal models with fully symbolic parameters. This makes the resulting proofs applicable to a wide range of actual systems, but may obfuscate the assurance argument for each particular one of them. Formal refinement arguments (e.g., in Hybrid Event-B [11], differential refinement logic [12], [13]) or formal instantiation/substitution arguments [14] create a hierarchy of models at various levels of abstraction, which can be navigated to gain an increasingly detailed understanding of the system behavior while preserving provable correctness along the refinement/instantiation hierarchy.

Crucially, however, in cyber-physical systems the final refinement step (from model to true system) is not solvable through refinement or code synthesis, because a vital part of the system is formed by non-engineered real-world dynamics. Next, we discuss how we can obtain explanations for this final refinement step even in cyber-physical systems.

C. Model Validation

For a comprehensive assurance argument, we want evidence about the correctness of the true system (not just about models of it). Neither testing alone nor formal verification alone can provide a comprehensive explanation of system correctness. Testing can only cover a finite amount of the infinitely many possible scenarios in a cyber-physical system. Formal methods, in contrast, provide strong correctness guarantees about all of the possible scenarios in a *formal model* of the system, e.g., in the form of proofs. This poses an inherent limitation for refinement and instantiation arguments: *there always exists a gap to the true system behavior, no matter how detailed we express formal models*. The remaining question, therefore, is whether the analyzed formal model accurately reflects the modeled system, a question tackled by *model validation*. Only few methods, however, combine offline proofs and model validation in a provably correct way: ModelPlex [2] transforms *by proof* hybrid systems models into monitoring conditions. The monitoring conditions inherit the predictive power of models to describe expected behavior, which makes the monitors capable of flagging discrepancies between models and true system execution. The monitoring conditions serve as a provably correct link between formal verification and model validation/testing and can be useful as explanations in the following ways. ModelPlex monitoring conditions can provide

- validation evidence about how accurately a model reflects the modeled system (are there discrepancies between collected data and the formal model?);
- validation evidence about model parameters (how robustly does the system fit the model, how robustly does it satisfy the inherited safety properties?);
- coverage evidence about a test suite (which aspects of the formal model are covered by test scenarios?);
- validation evidence about the safety-relevant similarity between a simulation environment and a true environment; may help in understanding the challenges of sim-to-real transfer in machine learning (how do the training scenarios in simulation compare to the encountered situations in true system execution?);
- dynamic evidence about system degradation or environment changes (how does safety robustness compare over a history of system executions?).

Model validation is particularly challenging in cyber-physical systems, because judgments about model accuracy must be made from imperfect sensors or based on partially available information [15]. In the presence of sensor uncertainty and partial observability, model validation, therefore, checks for existence of model executions that explain the values observed in the true system [15].

In order to align a symbolic monitor obtained from a symbolic model with the dynamic realities of the system, parametrization is required prior to model validation. Uniform substitution [14] is a technique to prove parameterized models from the symbolic model proofs without additional effort.

III. Proofs as Explanations

Hybrid systems proofs combine reasoning principles from a multitude of fields. In order to be useful as explanations, the challenge is to present proofs at an appropriate level of abstraction, without jeopardizing the soundness of their conclusions. High-level reasoning principles can be useful to gain an overall understanding of the safety argument, while foundational axioms can provide justification for each of the high-level reasoning steps if needed. In [16], different prover designs are compared for their tradeoffs in terms of achieving such levels of abstraction: traditional prover designs favor raw reasoning speed but each new rule increases the trusted code base, while a small-core design emphasizes a small set of foundational axioms and a universal axiom application algorithm, e.g., uniform substitution [14]. In either design, proofs can be explained using invariants and winning strategies as major proof insights, but only the small-core design with executable derived rules (e.g., in a tactic framework) allows us to explore a proof at varying degrees of abstraction without increasing the trusted code base.

A. System Invariants and Winning Strategies

Hybrid systems and hybrid games proofs, just like other proofs of sufficient complexity, are typically structured into supporting lemmas that identify intermediate proof obligations. In hybrid systems, those intermediate proof obligations identify important safety-related characteristics of the system; their role as proof insight in derived proof rules concisely summarizes high-level reasoning concepts—justified from foundational axioms—and might serve as explanations of safety-related properties of the system behavior as follows.

a) Inductive invariants: are guaranteed to be maintained by the combined discrete and continuous system behavior. The derived rule "loop" [16] below expresses that property P is true after all possible ways of executing the repeated program α^* (conclusion below the horizontal bar) when we show three properties of a loop invariant J (premises above the horizontal bar): *(i)* J must hold under the assumptions (left premise, J or alternatives Δ follow from the assumptions Γ), *(ii)* be maintained by the program (middle premise, assuming J, all runs of the loop body α maintain J, expressed by $[\alpha]J$), and *(iii)* imply safety (right premise, P is true assuming J).

$$\text{(loop)}\ \frac{\Gamma \vdash J, \Delta \quad J \vdash [\alpha]J \quad J \vdash P}{\Gamma \vdash [\alpha^*]P, \Delta}$$

Conducting proofs with parametric loop invariants [17] allows us to explore different options for loop invariants in the same proof. That way we can explore modifications of provided loop invariants when scrutinizing a proof for certification to gain a better understanding of the ways in which a safety argument could be strengthened.

b) Intermediate conditions: identify program contracts, e.g., what a controller must guarantee for the continuous dynamics to stay within a safe region. The derived rule MRp below identifies condition Q to explain how reasoning about a sequential program $\alpha;\beta$ is separated into local program proofs $[\alpha]Q$ from assumptions Γ and $[\beta]P$ from assumptions Q:

$$\text{(MRp)}\ \frac{\Gamma \vdash [\alpha]Q, \Delta \quad Q \vdash [\beta]P}{\Gamma \vdash [\alpha;\beta]P, \Delta}$$

c) Proof rules for continuous dynamics: differential cuts [18] describe continuous dynamics with increasingly fine-grained regions, which explain the safety-relevant boundaries of continuous dynamics (e.g., dynamics never exceeds some threshold). A useful technique to prove and explain safety properties of continuous dynamics are energy conservation arguments, which typically require constructing a "ghost" differential equation [18] that balances energy with the original differential equation. Once constructed, we then characterize the continuous dynamics with differential invariants [14], Barrier certificates [19], or Lyapunov functions that are preserved throughout the continuous dynamics. Proof rules for such reasoning concepts about continuous dynamics can be derived [18] in dL. These derived proof rules not only summarize the more fine-grained reasoning steps of the prover core, but also can check the results of numeric invariant/Lyapunov generation methods in order to symbolically explain and justify their correctness.

d) Stability proofs: explain absence of subtle instability behavior [20] known to occur in switched systems of combined discrete switching behavior and continuous system behavior. In [21] several proof rules for different switching mechanisms are derived within dL. Stability arguments based on these rules explain how a system remains stable under small perturbations (stability) and eventually dissipates the energy of these perturbations (attractivity).

e) Winning strategies: provide witnesses for the choices (existential) of an ego-agent in order to counteract adversarial (universal) environment behavior in hybrid games proofs. For example, [22] identifies climb rate strategies for vertical aircraft collision avoidance (e.g., when intruder aircraft is in a certain region, climb with at least some minimum upwards acceleration; in another region, descend). Together with discrete and continuous invariants, winning strategies summarize and explain the main proof arguments.

In terms of explanations, invariant regions and other conditions are amenable for graphical presentation, while winning strategies represent step-by-step instructions of how to react to certain events and conditions. Their justification using proof rules derived from core axioms opens up the possibility to understand the proof in a hierarchical sense, as discussed next.

B. Proofs at Different Levels of Abstraction

Theorem provers range from fully interactive to fully automatic, with most systems implementing a combination of interaction and automation to tackle complex systems. For example, the hybrid systems prover KeYmaera X [7] provides automated proof heuristics built on top of a tactics framework [23], which steers the soundness-critical prover core to generate proof terms [24]. Proof terms provide the most detailed view of a proof in the form of the soundness-critical operations of the core, such as builtin rules and axioms, while tactics provide an intermediate explanation in terms of the main proof insights, and the automated proof heuristics document existence of a proof. Even though proof terms justify correctness from just a small set of core logical concepts and, hence, reduce trusting the proof to trusting just these core concepts (instead of trusting the much larger prover software), they are difficult to decipher for their sheer size and lack of presentation structure. Proof terms are therefore mainly useful for proof checking. On the other end of the spectrum, fully automated proofs (as produced, e.g., by SAT/SMT solvers) or automated checks produced by reachability analysis tools, require trust in the full codebase of the tool.

A small-core theorem prover design, such as followed by KeYmaera X, provides opportunity to implement high-level reasoning steps as derived rules in a way that does not extend the soundness-critical core of the theorem prover [16]. Such a design allows us to provide proof explanations by unwinding proof details for all or some steps in a proof to produce a hierarchy of proof details, similar to abstraction in models. In this way, we can navigate a proof at varying levels of detail as our understanding of the proof increases. Fig. 1 shows a screenshot of the KeYmaera X UI with the single-step derived rule "loop" for induction proofs, and its expanded justification in terms of others tactics, core axioms, and proof rules.

IV. Discussion and Open Challenges

Early successes in formal methods for certifying cyber-physical systems (e.g., the Clearsy safety platform [25]) focus on providing a pre-certified platform, so that subsequent development using the platform can fast-track certification. Others emphasize the development process for certification, rather than the produced artifacts. In this paper, we argue for a complementary approach that considers aspects of formal models and proofs as explanations for certification. We take a holistic view that emphasizes the importance of providing explanations for certification authorities to scrutinize both the

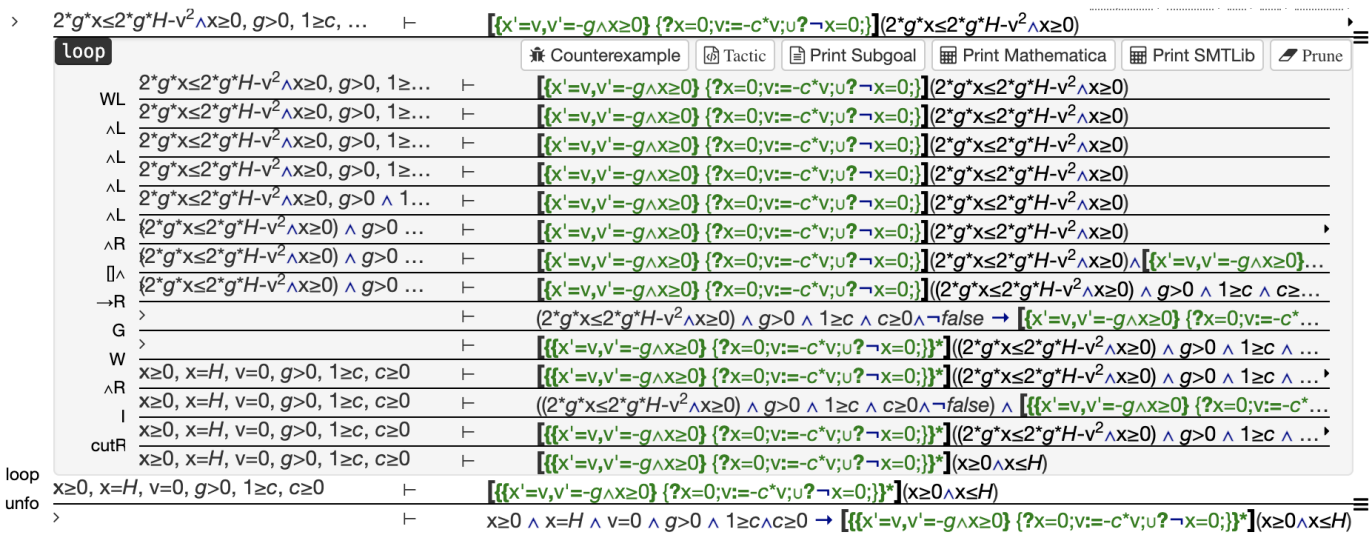


Fig. 1. Screenshot of the KeYmaera X UI: the derived rule "loop" offers a one-step explanation in terms of an induction proof; the icon ☰ to the right of this step is activated and shows a detailed justification of "loop" in terms of core axioms and other proof rules. The internal steps of tactic "unfold" are hidden.

models (how accurately do models reflect reality?) and the proofs (what are the major proof arguments?).

Even though the proof structure in terms of lemmas and derived rules are a possible way of achieving proof explanations at varying degree of detail, proof trustworthiness and presentation is challenging since computer-checked proofs may have millions of proof steps. Another aspect of proofs that is not yet explored here is vacuity: proofs may succeed for the wrong reasons (e.g., because models exclude important behavior), and we need logic tools to demonstrate that our models and proofs are not vacuously true [26].

Acknowledgments

This material is based upon work supported by the US Air Force and DARPA under Contract No. FA8750-18-C-0092.

References

[1] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" RAND Corporation, Tech. Rep. RR-1478-RC, 2016.

[2] S. Mitsch and A. Platzer, "ModelPlex: Verified runtime validation of verified cyber-physical system models," *Form. Methods Syst. Des.*, vol. 49, no. 1-2, pp. 33–74, 2016, selected papers from RV'14.

[3] H. Torfah, S. Junges, D. J. Fremont, and S. A. Seshia, "Formal analysis of ai-based autonomy: From modeling to runtime assurance," in *RV*, 2021, pp. 311–330.

[4] S. Jha, T. Sahai, V. Raman, A. Pinto, and M. Francis, "Explaining AI decisions using efficient methods for learning sparse boolean formulae," *J. Autom. Reason.*, vol. 63, no. 4, pp. 1055–1075, 2019.

[5] D. Bayani and S. Mitsch, "Fanoos: Multi-resolution, multi-strength, interactive explanations for learned systems," in *VMCAI*, 2022, pp. 43–68.

[6] A. Platzer, "Differential game logic," *ACM Trans. Comput. Log.*, vol. 17, no. 1, pp. 1:1–1:51, 2015.

[7] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer, "KeYmaera X: An axiomatic tactical theorem prover for hybrid systems," in *CADE*, 2015, pp. 527–538.

[8] A. Müller, S. Mitsch, W. Retschitzegger, W. Schwinger, and A. Platzer, "Tactical contract composition for hybrid system component verification," *STTT*, vol. 20, no. 6, pp. 615–643, 2018.

[9] S. Lunel, S. Mitsch, B. Boyer, and J. Talpin, "Parallel composition and modular verification of computer controlled systems in differential dynamic logic," in *FM*, 2019, pp. 354–370.

[10] A. Müller, S. Mitsch, W. Schwinger, and A. Platzer, "A component-based hybrid systems verification and implementation tool in KeYmaera X (tool demonstration)," in *CyPhy*, 2018, pp. 91–110, selected papers.

[11] R. Banach, M. J. Butler, S. Qin, N. Verma, and H. Zhu, "Core hybrid Event-B I: single hybrid Event-B machines," *Sci. Comput. Program.*, vol. 105, pp. 92–123, 2015.

[12] S. M. Loos and A. Platzer, "Differential refinement logic," in *LICS*, 2016, pp. 505–514.

[13] S. Mitsch, J.-D. Quesel, and A. Platzer, "Refactoring, refinement, and reasoning: A logical characterization for hybrid systems," in *FM*, 2014, pp. 481–496.

[14] A. Platzer, "A complete uniform substitution calculus for differential dynamic logic," *J. Autom. Reas.*, vol. 59, no. 2, pp. 219–265, 2017.

[15] S. Mitsch and A. Platzer, "Verified runtime validation for partially observable hybrid systems," *CoRR*, vol. abs/1811.06502, 2018.

[16] ——, "A retrospective on developing hybrid systems provers in the KeYmaera family - A tale of three provers," in *Deductive Software Verification: Future Perspectives*, ser. LNCS, W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, and M. Ulbrich, Eds. Springer, 2020, vol. 12345, pp. 21–64.

[17] S. Mitsch, "Implicit and explicit proof management in KeYmaera X," in *Proceedings of the 6th Workshop on Formal Integrated Development Environment, F-IDE@NFM 2021, 24-25th May 2021*, 2021, pp. 53–67.

[18] A. Platzer and Y. K. Tan, "Differential equation invariance axiomatization," *J. ACM*, vol. 67, no. 1, pp. 6:1–6:66, 2020.

[19] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *HSCC*, 2004, pp. 477–492.

[20] D. Liberzon, *Switching in Systems and Control*, ser. Systems & Control: Foundations & Applications. Birkhäuser, 2003.

[21] Y. K. Tan, S. Mitsch, and A. Platzer, "Verifying switched system stability with logic," in *HSCC*. ACM, 2022, pp. 1–11.

[22] R. Cleaveland, S. Mitsch, and A. Platzer, "Formally verified next-generation airborne collision avoidance games in ACAS X," *Transactions on Embedded Computing Systems*, 2022.

[23] N. Fulton, S. Mitsch, R. Bohrer, and A. Platzer, "Bellerophon: Tactical theorem proving for hybrid systems," in *ITP*, 2017, pp. 207–224.

[24] N. Fulton and A. Platzer, "A logic of proofs for differential dynamic logic: Toward independently checkable proof certificates for dynamic logics," in *CPP*, 2016, pp. 110–121.

[25] T. Lecomte, D. Déharbe, P. Fournier, and M. Oliveira, "The CLEARSY safety platform: 5 years of research, development and deployment," *Sci. Comput. Program.*, vol. 199, p. 102524, 2020.

[26] Y. Selvaraj, J. Krook, W. Ahrendt, and M. Fabian, "On how to not prove faulty controllers safe in differential dynamic logic," *CoRR*, vol. abs/2207.05854, 2022.

Towards Explainable Formal Verification

Ruben Martins
Carnegie Mellon University
Pittsburgh, USA
rubenm@andrew.cmu.edu

***Abstract*—Formal verification is essential to prove the correctness of safety-critical systems. However, many of these safety-critical systems require certification. When a program is proven correct by formal verification tools, it is often the case that no evidence is provided by the formal verification tool explaining why the program is correct. The lack of explainability makes it harder to use formal verification tools for certification.**

In this position paper, we advocate that explainability should be a first-class citizen and that formal verification tools should consider explanations when finding bugs or proving the correctness of programs. We show the current explanations of model checking tools such as CBMC and propose research directions for more explainable formal verification tools that can improve software certification, real-time systems, among others, for safety-critical systems.

***Index Terms*—verification, explainability, certification**

I. Introduction

Safety-critical systems [1], such as railway control systems, aviation systems, and electric power grid systems, have a significant software component that must be correct to ensure the proper behavior of the system. The use of formal methods is a way to increase confidence in these systems since the underlying software can be proven to be mathematically correct to some user-defined specification.

The use of formal verification tools increases trust in safety-critical systems. However, when proving a program is correct, formal verification tools do not provide any interpretable explanation of why the program is correct. Some safety-critical systems, such as aviation, require certification by human analysts to ensure that the development of aviation systems complies with software and hardware standards such as DO-178C [2] and DO-254 [3]. A human analyst may pose the question: *"Why should I trust the verification tool when it just outputs that the system is correct without further evidence?"* This is a valid question since the formal verification tool could itself have bugs that could put in doubt the verified results.

Recently, there have been reports of bugs in constraint solvers such as SMT solvers [4]–[6], which are used by many formal verification tools as their decision procedure. To increase the trust in constraint solvers, it is common for these tools to output a machine-checkable proof that can be checked by an independent and verified checker [7], [8]. A similar approach is also made for formal verification tools, where a correctness witness [9] can be emitted and checked by a third-party validator [10]. This verification approach does not guarantee that the constraint solver or formal verification tool has no bugs, but at least it guarantees that the verification result for that given program is correct. The machine-checkable proof increases the trust in constraint solvers and formal verification tools but does not provide explainable evidence that human analysts can use for certification.

In this position paper, *we advocate that formal verification tools should not focus only on correctness and scalability but must also consider explaining why the program is buggy or correct*. Programmers and human analysts can use these explanations to either fix the program or certify that the program is correct. We believe that improving explainable formal verification is a critical step toward using more formal verification tools for certification and non-experts to understand better why the program is buggy or correct.

This position paper about making formal verification tools more explainable is organized as follows. First, we show examples of the output returned by formal verification tools such as CBMC and discuss why they are not explainable and how they could be improved. Next, we suggest some ideas to make formal verification tools more explainable for buggy and correct programs. Finally, we show how explainable formal verification is not limited to software and should be applied to other applications such as real-time systems.

II. Motivation

Many formal verification tools can prove the correctness of software to a given specification or find a counterexample that shows the program is buggy [11]–[13]. Bounded model checking is a technique steered towards finding bugs, but it can also prove program correctness if it can unroll all loops completely. In this section, we will use CBMC [14], [15] to show an example of the lack of explainability of formal methods tools when verifying the correctness of software. CBMC is a bounded model checker that can reduce questions about program paths to logical constraints by unrolling loops in the program. This logical formula can then be solved by off-the-shelf constraint solvers such as SAT [16] or SMT solvers [17]. CBMC can verify memory safety, checks for exceptions, checks for various variants of undefined behavior, and user-specified assertions.

Consider the buggy program in Listing 1 that finds the minimum element of an array (`min`) and its corresponding position (`pos`). The program is buggy since, on line 4, the minimum value is initialized to be 0. Due to this incorrect initialization, the program may return the wrong minimum value if the array does not contain elements smaller than 0.

Listing 1. Buggy program that find the minimum element of an array

```
1   int a[10];
2   int i = 0;
3   int pos = 0;
4    int min = 0; // bug
5   while ( i < 10 ) {
6     if ( a[i] < min ) {
7     min = a[i];
8     pos = i;
9     }
10    i = i + 1;
11  }
12
13 // specification
14  assert (a[pos] == min);
15
16  for (i = 0 ; i < 10 ; i++)
17    assert(a[i] >= min);
```

If we give this program to CBMC, *how explainable is the counterexample* returned by this tool? CBMC can find a program trace that falsifies the user assertion on line 14. A programmer can use this trace to fix the program to comply with the specification given by the user.

Listing 2. Excerpt of the output of CBMC for the buggy min program

```
1   ...
2   [main.assertion.1] line 20 assertion a[pos] ==
        min: FAILURE
3   [main.assertion.2] line 23 assertion a[i] >=
        min: SUCCESS
4
5   Trace for main.assertion.1:
6
7   State 17 file min_bug.c function main line 7
        thread 0
8   ----------------------------------------------------
9     a={ 1, 0, 0, 0, 1879048192, 1073741824,
          1376256000, 0, 512, 0 } ({ 00000000
          00000000 00000000 00000001, 00000000
          00000000 00000000 00000000, 00000000
          00000000 00000000 00000000, 00000000
          00000000 00000000 00000000, 01110000
          00000000 00000000 00000000, 01000000
          00000000 00000000 00000000, 01010010
          00001000 00000000 00000000, 00000000
          00000000 00000000 00000000, 00000000
          00000000 00000010 00000000, 00000000
          00000000 00000000 00000000 })
10
11  State 18 file min_bug.c function main line 8
        thread 0
12  ----------------------------------------------------
13    i=0 (00000000 00000000 00000000 00000000)
14
15  State 19 file min_bug.c function main line 8
        thread 0
16  ----------------------------------------------------
17    i=0 (00000000 00000000 00000000 00000000)
18
19  ...
20
21  State 23 file min_bug.c function main line 11
        thread 0
22  ----------------------------------------------------
23    min=0 (00000000 00000000 00000000 00000000)
24
25  ...
26
27  State 58 file min_bug.c function main line 17
        thread 0
28  ----------------------------------------------------
29    i=9 (00000000 00000000 00000000 00001001)
30
31  State 62 file min_bug.c function main line 17
        thread 0
32  ----------------------------------------------------
33    i=10 (00000000 00000000 00000000 00001010)
34
35  Violated property:
36    file min_bug.c function main line 20 thread 0
37    assertion a[pos] == min
38    !((signed long int)!(a[(signed long int)pos]
          == min) != 0l)
```

The program trace contains an initial assignment to the array 'a' and assignments to all of the variables during the program's execution. Listing 2 shows an excerpt of the trace returned by CBMC. A trace provides a reasonable explanation, but even for this simple program, the program trace contains 79 lines with 62 states of variables. Even for this simple program, it may take the programmer some time to understand the counterexample.

To improve the explanation given by CBMC, researchers have automatically created test cases from program traces [18]. Programs can use test cases within their IDE and debugging tools to find the root cause of the program. In this case, a test case could be constructed by the initialization of the array 'a={1, 0, 0, 0, 1879048192, 1073741824, 1376256000, 0, 512, 0}' together with the expected output of 'min = 0' and 'pos = 1'. Note that running the buggy program would output 'min = 0' and 'pos = 0' incorrectly, thus falsifying the assertion on line 15.

However, a more *interpretable explanation could be given using natural language*. For instance, a programmer is more likely to understand the root cause of the bug if a message like "Incorrect initialization of the variable min since the array may not contain values smaller than 0." was returned. Other alternatives could include fixing the user assertion by saying, "The current program is correct only if there exists a value in the array smaller than 0."

If this program was fixed by replacing the initialization of the min variable with 'min = a[0]' then CBMC would return that the program is correct with respect to its specification. When a program is proven to be correct, CBMC [1] can output a certificate of correctness called a *correctness witnesses* [9]. These witnesses are represented semantically as automata and stored syntactically as GraphML [20]. A correctness witness is valid if its predicates are invariants for the program, and its correctness can be checked by independent validators [10], [21]. This checking procedure can increase the trust in formal methods tools since when they output that a program is correct, this can now be checked by a third party. However, this format is not meant to be interpretable, and for this simple program, the GraphML file generated by CBMC already contains 67 edges and 68 nodes.

When the program is correct, it is not trivial to have an interpretable justification. Even if a programmer is asked to

[1]All verification tools that participate in the annual software verification competition [19] must output a correctness witness when the program is correct.

explain why this algorithm is correct, the explanation may not be simple, but it may follow the notion of loop invariants. A more interpretable justification could be that if `min` is initialized with `a[0]`, then at the beginning of the loop, this variable stores the minimum value of the array up to position 0. While the loop is executed, we know that `min` stores the minimum value of the array up to position `i`. Therefore, at the end of the loop execution, `min` will contain the minimum element in the array. Using the program's invariants as part of the justification can be a helpful way to convince a human analyst that the program is correct.

The kind of explanations presented in this section for both buggy and correct programs could significantly improve the explainability of the output of formal verification tools. It would make them more likely to be used by programmers and certification entities.

III. Explainable Formal Verification

In this position paper, we advocate that the output of formal methods tools should become more explainable. In Section II, we have shown some examples of the output of verification tools when the program is buggy (e.g., counterexample as a trace of the program) and when the program is correct (e.g., a correctness witness). However, as discussed in Section II, these outputs are not necessarily interpretable. For formal methods tools to be more used by programmers and to be adopted as the standard in certification, the formal methods community should not focus only on the scalability of formal methods tools but also on explainability. We argue that explainability can be even more critical than scalability and should be treated as a first-class citizen. If programmers and analysts do not understand the outputs of formal methods tools, they are less likely to use them even if they improve their scalability.

Explainability is becoming the target of current research on other areas, such as explainable AI [22]–[24] in which humans can understand the decisions or predictions made by AI engines such as neural networks. Since AI components are used as black-boxes and are becoming part of safety-critical systems [25], it is critical to assure they have a correct behavior. Even though experts well understand the approaches used by formal verification tools, they are often used as black-boxes by programmers and analysts, and similar reasoning to explainable AI could also be used for formal verification tools.

The need for better explainability leads us to our main question in this position paper: *How can formal verification tools become more explainable?* This is an open problem that we believe should be the target of future research. However, we summarize some initial ideas presented in this position paper that can improve the explainability of the output of verification tools on buggy and correct programs.

Buggy programs: These are easier to be understood since formal verification tools frequently output a counterexample that corresponds to a trace of a program. However, interpreting this trace is still challenging and we believe that additional forms of explainability are needed. Some potential explanations that formal verification tools should support:

- *Test cases:* automatically generating test cases from the counterexamples and letting programmers use their usual debugging tools to analyze the issue.
- *Natural language descriptions:* transforming the counterexample into a natural language description of the problem that can help the programmer to find the bug.
- *Fault localization:* list of lines that are the root cause of the bug and that should be modified by the programmer.
- *Preconditions:* description of a precondition that would make the program correct. These preconditions could be helpful information for the programmer to know that if these are met then the program is correct.

Correct programs: Explaining that a program is correct is a more challenging problem. Formal verification tools can prove correctness by reasoning about program invariants. Some of these tools can output a correctness witness, as referred in Section II. However, these witnesses are not interpretable by humans. Some potential explanations for correct programs that formal verification tools could support are:

- *Invariants:* showing invariants satisfied by the program can help programmers and analysts understand why the program is correct.
- *Natural language descriptions:* transforming invariants into natural language and having a description of why the program is correct would also help convince others about the correctness of the program.
- *Range of values:* when proving that a program cannot reach a given state, it can be helpful to show an analyst the range of values each variable can take. Additionally, we could show that the path to reach the goal state is not feasible by showing that the range of variables does not satisfy the corresponding control-flow path.
- *Lifting logic to code:* Verification tools use constraint solvers to prove that the program is correct. The constraint solvers must show that the corresponding logical formula is unsatisfiable to prove that a program is correct. When solving this logical formula, the constraint solver learns lemmas that can potentially be lifted to the code and used as an explanation. The logical proof could also be lifted to code and used as an explanation. However, lifting from logic to code is challenging since code statements do not necessarily have a one-to-one matching with logical constraints.
- *Correct-by-construction programs:* An alternative to explaining that a program is correct could be to automatically construct a program from its specification [26]. Even though this was once seen as impractical except for small programs, the recent advances in program synthesis as shown that this approach can scale to larger programs [27]–[30]. However, some of the recent program synthesis approaches do not have guarantees that the program satisfies a formal specification [28], [30] and use over-approximations instead (e.g., input-output examples or natural language). Another challenge is that if the program is constructed in one step, then the analyst

will also question if this program satisfies the desired specification. However, if the program's construction was done piece-wise, then this construction could explain why the specification is satisfied.

When using formal specifications to verify the correctness of a program, another challenge is *writing correct specification* that meets the user's intention. If the specification is incorrectly written, it is meaningless even if the system is verified for that specification since we did not prove the desired property. Writing specifications is particularly challenging for complex logic languages such as linear temporal logic (LTL) [31]. In the case of linear temporal logic, there has been some work on automatically creating specifications from positive and negative examples [32], [33]. However, since examples are an over-approximation of the specification, it could be the case that the specification that was constructed is still incorrect. Having an interaction loop with the user where a system asks questions until the user is convinced that the synthesized specification is correct is crucial for certification when using formal specifications.

IV. Explainable Real-time Systems

The future of explainable formal verification is not only applicable to software, and it can have many applications in critical systems that require certification, as is the case of real-time systems. For instance, consider *real-time scheduling*, where real-time systems have tasks with strict timing constraints that must be met to guarantee the correctness and safety of the system [34]–[36]. We envision a world where real-time scheduling can have interpretable explanations that can be helpful for analysts to fix potential issues or understand why the scheduling is not feasible.

Worst-case execution time (WCET): WCET analysis has many applications, like automotive motor control systems [37], [38]. However, current WCET analysis tools [39]–[42] do not explain the bounds in a way that can be certified by an independent party. Even when performing WCET analysis with model-checking [43], it may not be straightforward to map the counterexample back to a concrete path. Even if a concrete path could be extracted, this may suffer from similar issues as the one presented in Section II, i.e., the counterexample may contain many irrelevant steps. An ideal explanation could be a concrete input that triggers the WCET since it can be used as an explanation and be checked independently by running these inputs in the real-time system.

Rate monotonic analysis: Another example is the real-time scheduling for different scenarios. For instance, rate monotonic analysis [44], [45] is a mathematical solution that can prove that for all possible schedules, all deadlines are met. However, these theoretical analyses cannot generally return a counterexample when a schedule is impossible.[2] Recent approaches using model checking [46] can produce counterexamples for scenarios where the schedulability is unfeasible. Still, these counterexamples are hard to interpret, and only expert analysts would be able to use them to adjust the design and change the priorities of the tasks or the scheduling algorithm. Instead of trying to understand a long counterexample, these should be lifted to the real-time system domain and only consist of relevant information. For instance, the counterexample could contain two tasks with corresponding priorities that make the schedule not feasible or potentially suggestions to the analyst about changes to the system that would make the schedule feasible.

[2]For some scenarios, it may be possible to find an explanation of failure using schedulability analysis. For instance, when considering single-core Earliest Deadline First (EDF), solving the schedulability test can lead to an explanation of failure.

V. Conclusions

Formal verification tools are more scalable and powerful than before, but the lack of explainable results restricts their usage for certifying critical systems. This position paper advocates that explainability should be a *first-class citizen*. Formal methods tools must be able to explain their results to convince users about their correctness. We have proposed multiple research directions to make verification tools more explainable for buggy and correct programs. Explainable formal verification has many applications in critical systems that require certification, as in real-time systems. We hope that this position paper sparks interest from the formal methods community and the real-time systems community in developing explainable tools that can help analysts build verified systems and certify the results of formal methods tools.

VI. Acknowledgments

The author wishes to acknowledge the helpful and stimulating discussions with Björn Andersson and Dionisio de Niz about explainable formal verification and real-time systems. We are also thankful to the anonymous reviewers for their comments and suggestions that helped to improve this manuscript significantly. This work was partially supported under NSF Grant No. CCF-1762363, and CMU-Portugal project ANI 045917 funded by FEDER and FCT. All statements are those of the author, and do not necessarily reflect the views of any funding agency.

References

[1] J. Bowen and V. Stavridou, "Safety-critical systems, formal methods and standards," *Software engineering journal*, vol. 8, no. 4, pp. 189–209, 1993.

[2] L. Rierson, *Developing safety-critical software: a practical guide for aviation software and DO-178C compliance*. CRC Press, 2017.

[3] V. Hilderman and T. Baghi, *Avionics certification: a complete guide to DO-178 (software), DO-254 (hardware)*. Avionics Comm., 2007.

[4] M. N. Mansur, M. Christakis, V. Wüstholz, and F. Zhang, "Detecting critical bugs in SMT solvers using blackbox mutational fuzzing," in *FSE*. ACM, 2020, pp. 701–712.

[5] D. Winterer, C. Zhang, and Z. Su, "On the unusual effectiveness of type-aware operator mutations for testing SMT solvers," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, pp. 1–25, 2020.

[6] J. Park, D. Winterer, C. Zhang, and Z. Su, "Generative type-aware mutation for testing SMT solvers," *Proceedings of the ACM on Programming Languages*, vol. 5, no. OOPSLA, pp. 1–19, 2021.

[7] N. Wetzler, M. J. Heule, and W. A. Hunt, "DRAT-trim: Efficient checking and trimming using expressive clausal proofs," in *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 2014, pp. 422–429.

[8] L. Cruz-Filipe, M. J. Heule, W. A. Hunt, M. Kaufmann, and P. Schneider-Kamp, "Efficient certified RAT verification," in *ICAD*. Springer, 2017, pp. 220–236.

[9] D. Beyer, M. Dangl, D. Dietsch, and M. Heizmann, "Correctness witnesses: exchanging verification results between verifiers," in *FSE*, T. Zimmermann, J. Cleland-Huang, and Z. Su, Eds. ACM, 2016, pp. 326–337.

[10] D. Beyer and M. Spiessl, "Metaval: Witness validation via verification," in *CAV*. Springer, 2020, pp. 165–177.

[11] J. C. King, "Symbolic execution and program testing," *Communications of the ACM*, vol. 19, no. 7, pp. 385–394, 1976.

[12] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking." *Handbook of satisfiability*, vol. 185, no. 99, pp. 457–481, 2009.

[13] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in *PLDI*, 1977, pp. 238–252.

[14] D. Kroening and M. Tautschnig, "CBMC–C bounded model checker," in *TACAS*. Springer, 2014, pp. 389–391.

[15] P. Schrammel, D. Kroening, M. Brain, R. Martins, T. Teige, and T. Bienmüller, "Incremental bounded model checking for embedded software," *Formal Aspects Comput.*, vol. 29, no. 5, pp. 911–931, 2017.

[16] J. Marques-Silva, I. Lynce, and S. Malik, "Conflict-driven clause learning SAT solvers," in *Handbook of satisfiability*. ios Press, 2021, pp. 133–182.

[17] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of model checking*. Springer, 2018, pp. 305–343.

[18] A. Holzer, C. Schallhart, M. Tautschnig, and H. Veith, "FS hell: systematic test case generation for dynamic analysis and measurement," in *CAV*. Springer, 2008, pp. 209–213.

[19] D. Beyer, "Software Verification Competition," https://sv-comp.sosy-lab.org//, 2022, [Online; accessed 30-September-2022].

[20] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall, "GraphML Progress Report," in *GD*, P. Mutzel, M. Jünger, and S. Leipert, Eds., vol. 2265. Springer, 2001, pp. 501–512.

[21] D. Beyer, M. Dangl, D. Dietsch, M. Heizmann, and A. Stahlbauer, "Witness validation and stepwise testification across software verifiers," in *FSE*. ACM, 2015, pp. 721–733.

[22] H. Hagras, "Toward human-understandable, explainable AI," *Computer*, vol. 51, no. 9, pp. 28–36, 2018.

[23] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Metrics for explainable AI: Challenges and prospects," *arXiv preprint arXiv:1812.04608*, 2018.

[24] A. Holzinger, "From machine learning to explainable AI," in *DISA*. IEEE, 2018, pp. 55–66.

[25] G. Falco, B. Shneiderman, J. Badger, R. Carrier, A. Dahbura, D. Danks, M. Eling, A. Goodloe, J. Gupta, C. Hart *et al.*, "Governing AI safety through independent audits," *Nature Machine Intelligence*, vol. 3, no. 7, pp. 566–571, 2021.

[26] Z. Manna and R. J. Waldinger, "A Deductive Approach to Program Synthesis," *ACM Trans. Program. Lang. Syst.*, vol. 2, no. 1, pp. 90–121, 1980.

[27] Y. Feng, R. Martins, Y. Wang, I. Dillig, and T. W. Reps, "Component-based synthesis for complex apis," in *POPL*, G. Castagna and A. D. Gordon, Eds. ACM, 2017, pp. 599–612.

[28] Y. Feng, R. Martins, O. Bastani, and I. Dillig, "Program synthesis using conflict-driven learning," in *PLDI*, J. S. Foster and D. Grossman, Eds. ACM, 2018, pp. 420–435.

[29] S. Gulwani, O. Polozov, R. Singh *et al.*, "Program synthesis," *Foundations and Trends® in Programming Languages*, vol. 4, no. 1-2, pp. 1–119, 2017.

[30] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," *CoRR*, vol. abs/2107.03374, 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[31] D. Neider and R. Roy, "Expanding the horizon of linear temporal logic inference for explainability," in *REW*. IEEE, 2022, pp. 103–107.

[32] D. Neider and I. Gavran, "Learning linear temporal properties," in *FMCAD*, N. S. Bjørner and A. Gurfinkel, Eds. IEEE, 2018, pp. 1–10.

[33] J. Gaglione, D. Neider, R. Roy, U. Topcu, and Z. Xu, "Learning Linear Temporal Properties from Noisy Data: A MaxSAT-Based Approach," in *ATVA*, ser. Lecture Notes in Computer Science, Z. Hou and V. Ganesh, Eds., vol. 12971. Springer, 2021, pp. 74–90.

[34] S. K. Dhall and C. L. Liu, "On a real-time scheduling problem," *Operations research*, vol. 26, no. 1, pp. 127–140, 1978.

[35] L. Sha, T. Abdelzaher, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, A. K. Mok *et al.*, "Real time scheduling theory: A historical perspective," *Real-time systems*, vol. 28, no. 2, pp. 101–155, 2004.

[36] R. I. Davis and A. Burns, "A survey of hard real-time scheduling for multiprocessor systems," *ACM computing surveys (CSUR)*, vol. 43, no. 4, pp. 1–44, 2011.

[37] S. Byhlin, A. Ermedahl, J. Gustafsson, and B. Lisper, "Applying static WCET analysis to automotive communication software," in *ECRTS*. IEEE, 2005, pp. 249–258.

[38] A. Ermedahl, J. Gustafsson, and B. Lisper, "Experiences from industrial WCET analysis case studies," in *WCET*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2007.

[39] S. Hahn, M. Jacobs, N. Hölscher, K.-H. Chen, J.-J. Chen, and J. Reineke, "LLVMTA: An LLVM-Based WCET Analysis Tool," in *WCET*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[40] C. Ballabriga, H. Cassé, C. Rochange, and P. Sainrat, "OTAWA: An Open Toolbox for Adaptive WCET Analysis," in *Software Technologies for Embedded and Ubiquitous Systems*, ser. Lecture Notes in Computer Science, vol. 6399. Springer, 2010, pp. 35–46.

[41] L. Yun, A. Roychoudhury, and T. Mitra, "Timing Analysis of Body Area Network Applications," in *WCET*, ser. OASIcs, C. Rochange, Ed., vol. 6. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.

[42] D. Hardy, B. Rouxel, and I. Puaut, "The Heptane Static Worst-Case Execution Time Estimation Tool," in *WCET*, ser. OASIcs, J. Reineke, Ed., vol. 57. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 8:1–8:12.

[43] R. Metta, M. Becker, P. Bokil, S. Chakraborty, and R. Venkatesh, "TIC: a scalable model checking based approach to WCET estimation," in *LCTES*, T. Kuo and D. B. Whalley, Eds. ACM, 2016, pp. 72–81.

[44] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[45] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *RTSS*, vol. 89, 1989, pp. 166–171.

[46] J. Liu, M. Zhou, X. Song, M. Gu, and J. Sun, "Formal modeling and verification of a rate-monotonic scheduling implementation with real-time maude," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 4, pp. 3239–3249, 2016.

Towards Explainability in Modular Autonomous Vehicle Software

Hongrui Zheng*, Zirui Zang*, Shuo Yang*, Rahul Mangharam

***Abstract*—Safety-critical Autonomous Systems require trustworthy and transparent decision-making process to be deployable in the real world. The advancement of Machine Learning introduces high performance but largely through black-box algorithms. We focus the discussion of explainability specifically with Autonomous Vehicles (AVs). As a safety-critical system, AVs provide the unique opportunity to utilize cutting-edge Machine Learning techniques while requiring transparency in decision making. Interpretability in every action the AV takes becomes crucial in post-hoc analysis where blame assignment might be necessary. In this paper, we provide positioning on how researchers could consider incorporating explainability and interpretability into design and optimization of separate Autonomous Vehicle modules including *Perception, Planning,* and *Control*.**

I. Introduction

According to the Morning Consult and Politico poll [1], only 16% of respondents are "very likely" to ride as a passenger in an autonomous vehicle, while 28% of respondents state that they "not likely at all". Moreover, only 22% of respondents believe self-driving cars are safer than the average human driver, while 35% of them believing self-driving cars are less safe than the average human driver. The public's distrust in Autonomous Vehicles (AV) shows that improving explainability in AV software is a necessity.

There exist many surveys on explainable AI (XAI) and robotics [2]–[5]. Specifically, [6]–[8] surveys explainability in Autonomous Driving. Atakishiyev et al. [7] believes that AVs need to provide regulatory compliant operational safety and explainability in real-time decisions. It focuses on providing discussion through the cause-effect-solution perspective. Zablocki et al. [6] provides an in-depth overview of XAI methods in deep vision-based methods, but is limited to the scope of perception only. Omeiza et al. [8] also provides an overview of explanations in AVs in the full self-driving pipeline. Gilpin et al. [9] proposes explainability as the trade-off between *interpretability* and *completeness*. As described in [9], to be interpretable is to describe the internals of a system in such a way that is understandable to humans; to be complete is to describe the operation of a system in an accurate way.

We position ourselves to provide insight in augmenting explanability in Autonomous Vehicle's sense-plan-act software modules as a task of balancing interpretability and completeness. In this paper, we look at the explainability in existing works and in our recent contributions in localization, planning, and control. In each case, we want to be able to quantify the uncertainty at each step of the decision making and interpret the provenance of the outcome of the algorithm.

*Authors contributed equally. All authors are with University of Pennsylvania, Department of Electrical and Systems Engineering, 19104, Philadelphia, PA, USA. Emails: {hongruiz, zzang, yangs1, rahulm}@seas.upenn.edu

II. Explainability in Localization

Robot localization is a problem of finding a robot's pose using a map and sensor measurements, such as LiDAR scans. The map is pre-built and the environment is assumed to not change significantly after the map is captured. It is crucial for any moving robot to interact with the physical world correctly. However, the problem of finding the mappings between measurements and poses can be ambiguous, because sensor measurements from multiple distant poses can be similar. Therefore, to tightly integrate the localization module with other parts of the software stack and for the engineers implementing and tuning the algorithm, the explainability of localization algorithms using neural networks becomes important. We need to estimate the uncertainty of the localization results, and in the worst case, to know when and why the robot fails to localize on a certain map.

Monte Carlo Localization (MCL) [10], the widely adopted method, uses random hypothesis sampling and sensor measurement updates to infer the pose. In MCL, the proposed particles are explicit poses on the map and we can interpret the distribution of the particles as the uncertainties. The random generation of the particles can be tuned with parameters that have physical meaning, providing an interface for humans to adjust the behavior of the algorithm. Many developments in localization seek to improve within the framework of MCL. [11]–[13] Although particle filter has been a popular localization framework for its robustness and reliable performance, it introduces random jitter into the localization results.

Other common approaches are to use Bayesian filtering [14] or to find more distinguishable global descriptors on the map [15], [16]. In Bayesian filtering, the explainability lies in the conditional probability attached with the motion model and each measurement. The estimation of such probability is challenging. For the global descriptor approach, oftentimes manual selection of map features are needed, which increases the explainability of the system, but also increases the human workload and reduces robustness. Developments in localization research usually propose better measurement models or feature extractors within these frameworks. [17], [18].

Recent research in localization has also focused on the use of learning-based methods outside of the above frameworks [19]. Although learning-based methods may provide better localization precision with lower latency, the interpretability of the method decreases. While the traditional localization methods can be manually tuned according to the specific user scenarios, learning-based localization methods are usually not tunable once the network is trained. Uncertainty estimations of the neural networks also become a challenge for learning-based methods. There are efforts to approximate the uncer-

tainty [20]–[22], but it hasn't been widely applied.

Our contribution: In our recent paper, Local_INN, we proposed a new approach to frame the localization problem as an ambiguous inverse problem and solve it with an invertible neural network (INN) [23]. It stores the map data implicitly inside the neural network. With the assumption that the environment doesn't not change significantly from the map, by evaluating the reverse path of the neural network, we can get robot poses from LiDAR scans. It also provides uncertainty estimation from the neural network and is capable of learning and providing localization for complex environments.

Localization is an inverse problem of finding a robot's pose using a map and sensor measurements. This reverse process of inferring the pose from sensor measurements is ambiguous. Invertible neural networks such as normalizing flows [24] have been used to solve ambiguous inverse problems in various fields [25]–[28]. The version of normalizing flows we used is called RealNVP [29], which uses a mathematical structure called coupling layers to ensure the invertibility while performing transformations with arbitrary neural network layers, such as MLPs. This framework of solving inverse problems with normalizing flows was introduced by Ardizonne et al. [25] and was later extended by [26], [30] to include a conditional input that is concatenated to the vectors inside the coupling layers. They proposed to use normalizing flows to learn a bijective mapping between two distributions and use a normal-distributed latent variable to encode the lost information in training due to the ambiguity of the problem. The network can be evaluated in both forward and reverse paths. During the evaluation, repeatedly sampling the latent variable can give the full posterior distribution given the input.

In Local_INN, we use pose-scan data pairs to train such a bijective mapping. As shown in Fig. 1, The forward path is from pose to scan and the reverse path is from scan to pose. We use a conditional input calculated from the previous pose of the robot to reduce the ambiguity of the problem. Because INNs require the same input and output dimensions, we use a Variational Autoencoder [31] to reduce the dimension of the LiDAR scans and use Positional Encoding [32] to augment that of the poses. The network is trained with supervised loss functions on both sides. The data used for training the Local_INN can be simulated or real data recorded from LiDARs. In our experiments, we tested on both real and simulated data with 2D and 3D LiDARs. To collect training data, we uniformly sample x, y position and heading θ on the drivable surface of each map, and use a LiDAR simulator to find the corresponding LiDAR ranges. This means the trained network will be able to localize everywhere on the map. For each different map, we need to train a separate network. Map files are compressed inside the neural network and are no longer needed during evaluation.

We claim that INN is naturally suitable for the localization problem with improved explainability compared to other learning-based methods. In particular, uncertainty estimation and map representation are the two advantages that Local_INN provides in the context of explainability.

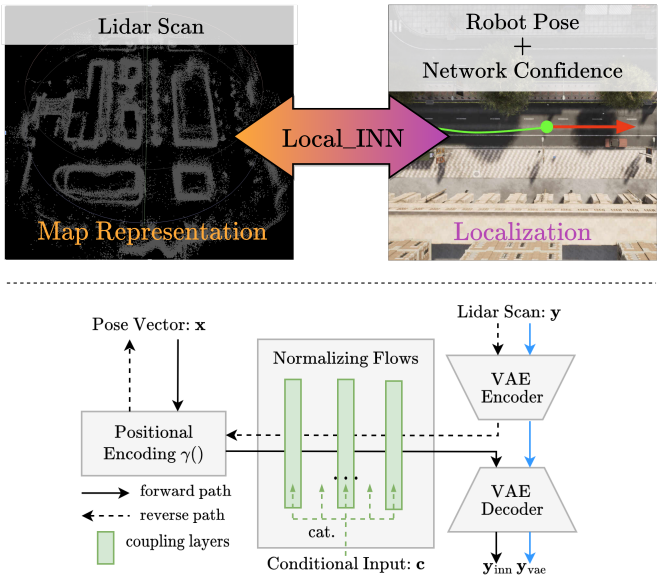
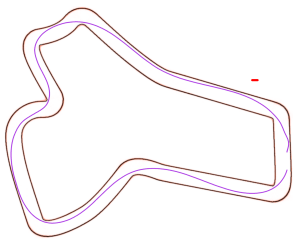
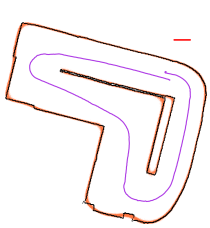


Fig. 1: **Top**: Evaluation of Local_INN in forward direction gives compressed map information, and in the reverse direction gives accurate localization with fast runtime and uncertainty estimation. **Bottom**: Network structure of Local_INN. Solid arrows are from pose to lidar scan. Dashed arrows are from lidar scan to pose. Conditional input is calculated from the previous pose of the robot.

A. Explainability from Uncertainty Estimation

When we use Local_INN to localize, the input to the reverse path of the INN consists of the LiDAR scans concatenated with a latent vector that is sampled from normal distribution. With this sampling of latent vector, the network can output not just a pose but a distribution of inferred poses. The covariance of this distribution can be used as the confidence of the neural network when fusing with other sensors. Uncertainty estimation improves explainability by providing information on the measurement quality of the prediction. Compared to learning methods that do not provide uncertainty estimates, it is much easier to determine whether the prediction of the neural network is lower in accuracy due to higher uncertainty, and improve the prediction results by augmentation. In our experiments, we used an EKF to fuse the localization result with the odometry information. The results show that this fusion significantly improved localization accuracy where the

TABLE I: Local_INN Experiments: Map Reconstruction and RMS Localization Errors with 2D LiDAR (xy[m], θ[°])

	Race Track (Simulation)	Outdoor (Real)
Original Map Reconstruction Test Trajectory		
Online PF	0.168, 2.107	0.047, 1.371
Local_INN+EKF	**0.056**, **0.284**	**0.046**, **1.130**

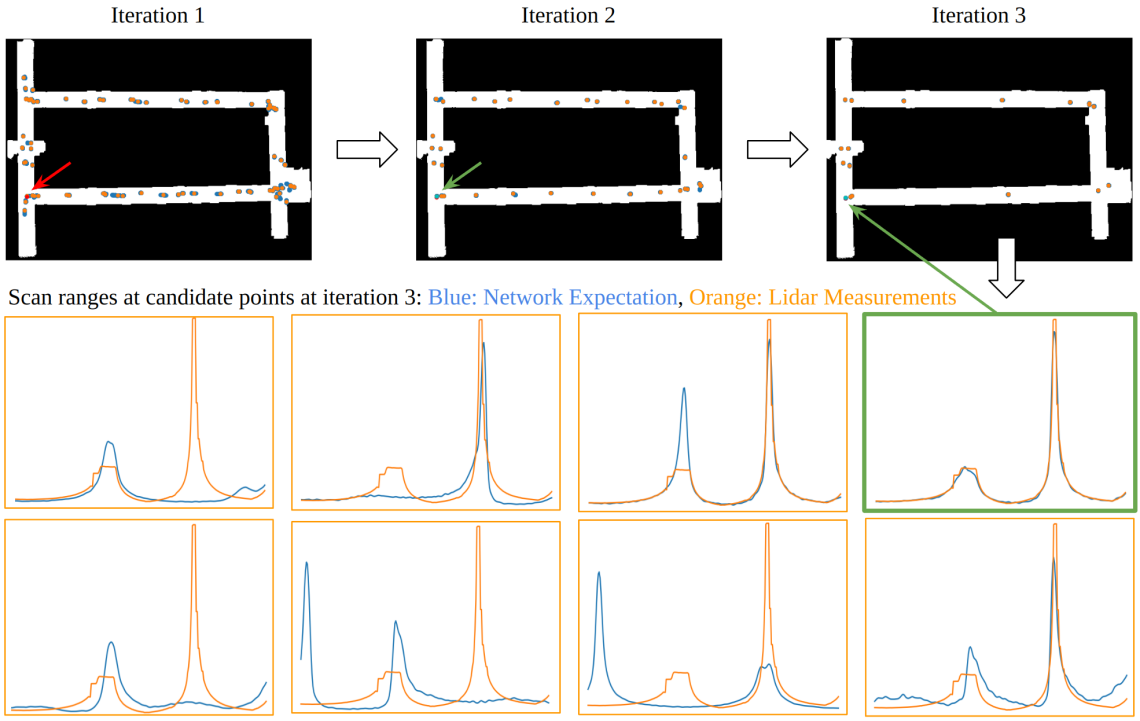


Fig. 2: Example of global localization finding the correct pose at the 2nd iterations (green arrow). **Top**: Narrowing down of the candidate poses in the first 3 iterations. We can see candidate poses on the map (orange dots), correct pose (red dot), selected pose (green dot). **Bottom**: Examples of LiDAR scan ranges at candidate poses at iteration 3 (orange boxes), and at the selected pose (green box). In the range plots, the horizontal axis is the angle of LiDAR scans and vertical axis is the measured distance. We can see a comparison of network expected ranges (blue curve) at various poses, and the actual LiDAR measurement (orange curve). The correct pose is selected where the measurement best matches the expected shape. The network is trained on simulated data and tested on real LiDAR data.

map geometry is ambiguous, which means this covariance is very effective in revealing the confidence of the network.

As shown in Table I, The accuracy of Local_INN is at par with the current localization efforts. See [23] for a comparative analysis of Local_INN localization accuracy in 2D and 3D maps.

B. Explainability from Map Representation

Local_INN provides an implicit map representation and a localization method within one neural network. The guaranteed invertibility of the neural network provides the use a direct way to check the neural network's 'understanding' of the map by reproducing part of the map with poses. That is, we can compare the reconstructed map to the original map to see how much detail is used by the neural network in localization. Again, this feature improves explainability in failure scenarios. When the localization fails, this comparison can help us explain the failure and guide us in improving the methods. For example, we can train with more data from a particular location on the map that was difficult to localize in.

As an example of how the stored map information in the forward path of the neural network can help us explain the localization results, let us consider the algorithm for global localization. Global localization is needed when a robot starts with an unknown pose or when the robot encounters the 'kidnapping problem'. In this case, it is challenging to find the correct position on the map due to the ambiguity of problem. MCL algorithms usually do global localization by spreading the covariance all around the map and using iterations of control inputs and measurements to decrease the covariance, which gives an explicit visualization to see the progress of the global localization processes. For other learning-based method, this process is usually hard to explain as we rely on the neural network to output poses as a black box.

With Local_INN, we can randomly initialize a set of random poses on the map as conditional inputs and use new lidar scans to narrow down the assumptions. In other words, we initially have multiple random assumptions of the robot's location on the map and use them as the conditional inputs for the nerual network. As shown in figure 2 iteration 1, when we input a LiDAR scan to the network along with these assumptions, it will output multiple possible pose distributions. In our algorithm, for each possible pose distribution, we compare the sensor measurement with what the neural network expects at this location, and use the reciprocal of the error term to weight the assumptions differently. The weights for the assumptions is used to determine the amount of latent variable we use. This process repeats with each new LiDAR scan we get from the sensor. In our experiments, the convergence of candidate

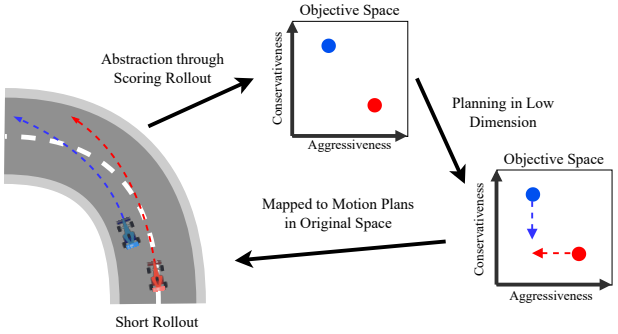


Fig. 3: Overview of Game-theoretic Objective Space Planning

poses is fast and accurate. As shown in iteration 2 and 3 in figure 2, even if we still track multiple poses, in this example, the correct pose is determined at the 2nd iteration in a highly symmetrical map. The low part of figure 2 shows plots of the expected lidar scan from the neural network and the current LiDAR measurement. This reveals the black-box process of the neural network so that we can see why multiple poses are possible and how we should decide which one to pick.

To summarize, the explainability of localization methods generally lies in the uncertainty estimation and the ability to explain and tune the methods when localization fails. Traditional localization methods usually offer higher interpretability than learning-based methods, whereas the learning-based methods can provide better empirical performance. The new method, Local_INN, we recently proposed uses an invertible network network architecture to solve the localization problem. It offers interpretability by giving uncertainty estimation through the covariance of the inferred pose distributions, and by ensuring the invertibility of the network so that we can reveal the what information the neural network is using during the localization. At the same time, it does sacrifice completeness by using a Variational Autoencoder (VAE) to model a latent space of the LiDAR scans.

III. Explainability in Planning

Planning is the task of finding a viable motion plan for the robot to reach some predetermined goal given the current observation of the environment through various sensors. The planning step is usually the next step after perception, or localization. Traditionally, sampling-based and model-predictive methods are the most popular choices in Autonomous Vehicle motion planning. Planning algorithms provide explanability through human-designed objectives: e.g. maximizing the distance of a planned trajectory to obstacles and undrivable areas, maximizing velocity on the trajectories, minimizing the lateral acceleration the vehicle experiences on a planned trajectory. We propose a unique position in adding interpretability and completeness to planning algorithms: explainability through abstraction. Next, we show our approach based on our recent paper on Game-Theoretic objective space planning [33].

In this case study, our primary context is two-player racing games in close proximity. An overall depiction of the pipeline is shown in Figure 3. We choose a game theoretic approach that models racing as a *zero-sum extensive game* with *imperfect information* and *perfect recall*. Extensive games model sequential decision-making of players and naturally form a game tree where each node is a decision point for a player. However, the planning problem presented in autonomous racing is continuous, and the state space of the agents, in turn, the game tree in the extensive game, will also be infinitely large if we model the game in the vehicle's planning space. Since the decision made by a game-theoretic algorithm in the planning space cannot be explained in a way that a human can understand, we use a lower dimensional space for planning. We define the notion of *Objective Space* $\mathcal{O}$. For each short rollout in an adversarial environment, we can compute multiple metrics regarding this agent's performance, such as safety and aggressiveness. These metrics also add to the interpretability of our planning algorithm while not losing the completeness. $\mathcal{O}$ models the average outcome of each agent against competent opponents. Using $\mathcal{O}$, our planner maps complex agent behaviors to a lower dimension where only the episodic outcome is recorded instead of the entire decision-making process in the planning space. We define an action in our game as movements in a dimension of $\mathcal{O}$. This action space is analogous to the planning space in a grid world with actions that move the agent to a neighboring cell. This means the planning problem is much simpler than the original problem.

In our case study, we choose *aggressiveness* and *restraint* as the two dimensions of $\mathcal{O}$. Aggressiveness is scored on an agent's lead over the other at the end of the rollout, and restraint is scored on an agent's time to collision to the environment and the other agent. Two movements are available for each dimension: increasing or decreasing for a fixed distance along the axis. For example, four discrete actions are available at each turn when $\mathcal{O} \in \mathbb{R}^2$. Even with the formulation of agent action space, the possible objective values of an opponent agent or possible position in $\mathcal{O}$ is still infinite. We propose a predictive model for regret values within Counterfactual Regret Minimization (CFR) [34] to make the problem tractable. Finally with head-to-head racing experiments, we demonstrate that using the proposed planning pipeline above significantly improves the win rate that generalizes to unseen opponents in an unseen environment.

A. Explainability in Agent Actions

In this section, we examine specific cases of agents moving in the lower dimension and investigate whether we achieve the task of instilling explainability in our algorithm. We choose a 2-D space that encodes aggressiveness and restraint. In Figure 4, we show four examples of races between two agents. The Ego (orange) uses our proposed game-theoretic approach in the Objective Space, and the Opponent (green) is static in the objective space. In the first two cases, the opponent is in the lower right quadrant, meaning that they're more conservative than aggressive. Hence our planner chooses to increase in aggressiveness continuously to win the races. In the last two

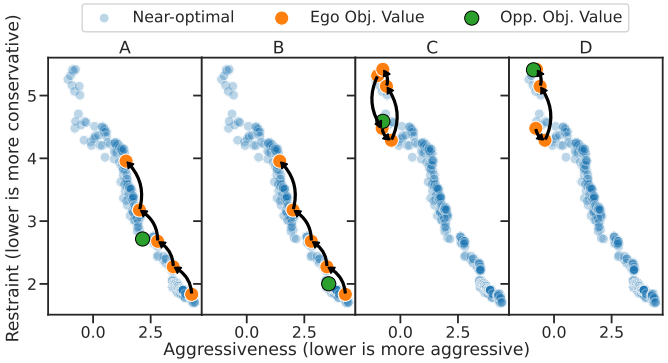


Fig. 4: Trajectories of ego moves in $\mathcal{O}$.

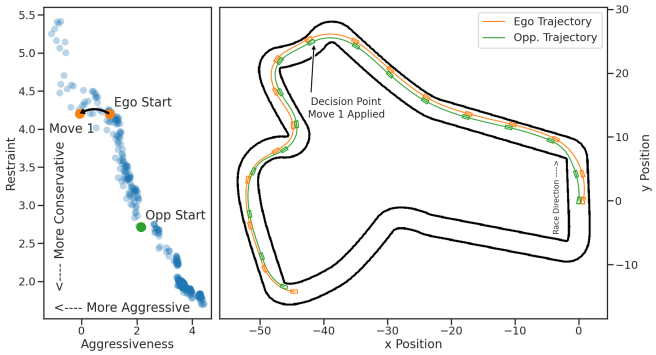


Fig. 5: Effect of making a move in $\mathcal{O}$ in motion planning space.

cases, the opponent is in the upper left quadrant, meaning that they're more aggressive than conservative. Hence our planner chooses to first become more conservative, and once a chance presents itself, increase in aggressiveness and win the races. In Figure 5, we inspect a specific move to show the effect in the motion planning space. In the beginning of the rollout, both agents are side by side. At a decision point, our planner locates the opponent in the lower right quadrant as more conservative than aggressive. Then the ego decides to increase in aggressiveness to finally overtake the opponent. From these examples, it is clear that moving the planning problem in a lower dimension that encodes interpretable metrics for humans doesn't decrease the capability of the planner, or the completeness in the algorithm.

IV. Explainability in Control

With robust localization, the autonomous vehicle can plan its trajectory but requires a safe an interpretable controller to execute the plan. In this section, we show our recent progress on learning-based safety-critical control through control barrier functions (CBFs) and provide our positioning on explainable safe control. Specifically, we show that our proposed differentiable safety filter has more completeness than non-differentiable safety filter without sacrificing interpretability.

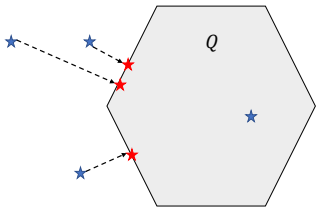


Fig. 6: Illustration of QP: If the NN output is not in the safe control set Q, it will be projected in minimally invasive way to Q; otherwise, the control keeps the same.

A. Safety-critical control

Learning-based control could provide high empirical performance thus it has become popular for controlling complex dynamical systems. However, learning-based controllers, such as neural network (NN), generally lack formal safety guarantees because of their black-box nature. This limits their deployments with complex safety-critical systems. To address safety risk, multiple methods have been proposed, such as model predictive control (MPC) [35], Hamilton-Jacobi reachability analysis [36], contraction theory [37], and control barrier functions (CBF) [38].

Among the many safety-critical control techniques, CBF is becoming a popular choice since it explicitly specifies a safe control set by using a Lyapunov-like condition and guards the system inside a safe invariant set. When a continuous-time control-affine system is considered, such projection reduces to a convex quadratic program (QP) which is referred to as CBF-QP. Due to its simplicity, flexibility, and formal safety guarantees, CBFs have been applied in safe learning control with many successful applications [39]–[43].

Compared with MPC, which needs to handle a possibly nonconvex optimization problem in the face of nonlinear dynamical systems, CBF-QP is computationally efficient to solve online. However, unlike MPC, the QP-based safety filter only operates in a minimally invasive manner, i.e., it generates the safe control input closest to the reference control input (in the Euclidean norm), as shown in Fig. 6, unaware of the long-term effects of its action. This indicates that the effects of the safety filter on the performance of the closed-loop system are hard to predict. Therefore, the application of the safety filter may give rise to myopic controllers [44] that induce subpar performance in the long term.

To address the issue of myopic CBF-based safety filters, in our recent work [45], we propose to utilize CBF to construct safe-by-construction NN controllers that allow end-to-end learning. Incorporating safety layers in the NN controller allows the learning agent to take the effects of safety filters into account during training in order to maximize long-term performance.

We design a differentiable safety layer using the gauge map [46] (as shown in Fig. 7) which establishes a bijection mapping between the polytopic set of a NN output (e.g., an ℓ_∞ norm ball) and the CBF-based safe control set. The

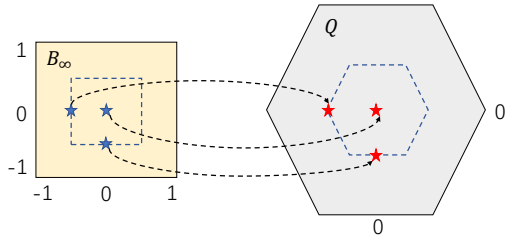


Fig. 7: Illustration of the gauge map from the ℓ_∞ ball B_∞ to a polytopic set Q. The original point in B_∞ and mapped point in Q are in the same level set and with the same direction.

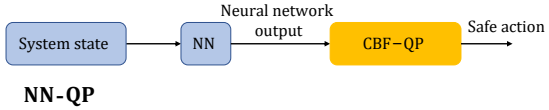
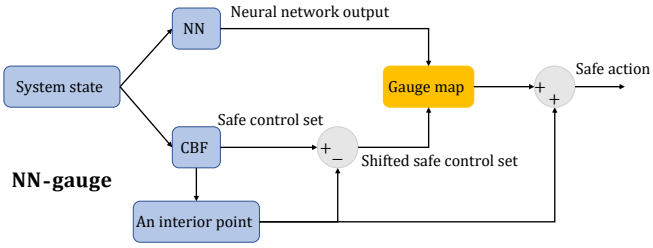
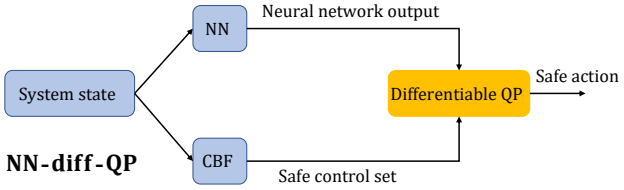


Fig. 8: NN controller with CBF-QP safety filter.



(a) Gauge map-based safe NN controller architecture.



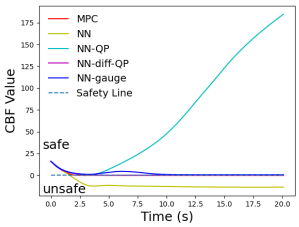
(b) CBF-QP-based safe NN controller architecture.

Fig. 9: Safe-by-construction NN controllers that utilize CBFs to construct differentiable safety layers (yellow blocks).

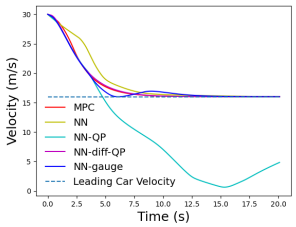
proposed architecture is denoted as NN-gauge (Fig. 9a). We compare NN-gauge with an alternative differentiable safe-by-construction NN controller called NN-diff-QP, which consists of a NN followed by a differentiable CBF-QP layer (Fig. 9b). Specifically, NN-diff-QP (Fig. 9b) concatenates a differentiable projection layer which can be implemented in a NN using toolboxes such as cvxpylayers [47], qpth [48]. NN-gauge involves finding an interior point of the safe control set since gauge map requires that Q set in Fig. 7 is convex and cover the origin. It can be reduced to implicitly finding the Chebyshev center [49] of the safe control set.

Remark 1: In the online execution, NN-gauge requires closed-form evaluation or solving a linear program (LP) while NN-diff-QP solves a quadratic program. Both methods are significantly cheaper to run than MPC.

As an example, let's consider the adaptive cruise control (ACC) in which the ego car is expected to achieve the desired



(a) CBF values.



(b) Velocity of the ego car.

Fig. 10: Results of adaptive cruise control. CBF values (left) and velocity of the ego car (right) under different controllers are evaluated in closed-loop for 20s. A CBF value below zero indicates unsafety, and the optimal behavior of the ego car is expected to have a steady state velocity of $16m/s$, same as the leading car.

cruising speed while maintaining a safe distance from the leading car. As shown in Fig. 10b, applying the CBF-QP safety filter directly to enforce safe control deteriorates the long-term closed-loop performance of the NN controller. However, both NN-gauge and NN-diff-QP achieve similar closed-loop performance, and they are comparable to MPC.

B. Explainability of safety-critical control

1) Non-differentiable QP filter: Quadratic program (QP) is the classic minimally invasive safety filter to modify the nominal unsafe controller. Due to its simplicity and the nature of minimal modification of the original controller, it has always been equipped to CBF as the safety filter. As shown in Fig. 6, it is very understandable to humans, since one only needs to project the unsafe control input to the "nearest" point in the safe control set, which is specified by CBF. Thus, QP safety filter enjoys high interpretability.

However, on the other hand, the QP operation is not necessarily the best choice from the perspective of optimizing a long-term system objective. For example, in the long run, sometimes it might be better to choose a "farther" point in the safe control set than the "nearest" point on the boundary when the nominal control is not safe. These modified actions are unaware of the long-term effects and may give rise to myopic controllers which deteriorate their performance. From the ACC case study, we can also observe in Fig. 10b that the modified control (i.e., NN-QP) is myopic with unpredictable long-term effects. Thus, vanilla QP filter lacks the completeness in the sense of controlling the system in the optimal way with the safety guarantee.

2) Differentiable safety filters: Our proposed two differentiable safety filters, i.e., differentiable QP and gauge map, enjoy both a high level of interpretability and completeness.

- **Differentiable QP**: the QP structure is the same with vanilla CBF-QP, so there is no interpretability loss in differentiable QP. On the other hand, since the NN controller considers the gradient of QP structure while training to minimize the long-term loss, so the trained

NN adapts to QP much better than non-differentiable QP. Thus, even if the QP is applied to ensure safety when evaluating online, the modified controller could still have good performance. Therefore, from the explainability perspective, completeness is improved with no loss on interpretability.

- **Gauge map**: unlike QP, gauge map seeks to map the original control "proportianally" to the safe control set with the same direction. This mapping is also understandable to humans. Furthermore, since gauge map is sub-differentiable itself, so the end-to-end training holds naturally, which allows it to perform as good as differentiable QP after training. Thus, explanation could find the high completeness in this method as well.

In the ACC example, it also has been shown that both NN-gauge and NN-diff-QP have comparable performance with MPC, which implies the high completeness from demonstration. Looking ahead, we are interested in exploring a more general parameterized gauge map method, which even not necessarily maps to the same level set. In that case, it will perhaps lose some interpretability but have better completeness as it is not limited to the same level set mapping. This way, there is more flexibility while choosing the safe control input.

V. Discussion and Conclusion

In any autonomous vehicle which operates a sense-plan-act processing loop, the essential components are localization, planning and control. For humans to trust the system to make decisions and actions on their behalf in safety-critical situations, it is essential to have explainability across the stack. We view explainability as a trade-off between *interpretability* of the machine's actions and the *completeness* in terms of describing the operation of the system in an accurate way.

The explainability of localization methods generally lies in the uncertainty estimation and the ability to explain and tune the methods when localization fails. We introduce Local_INN, which utilizes an invertable neural network architecture that provides explainability for uncertainty estimation and explainability from map representation. Uncertainty estimation, through the covariance of the inferred pose distributions, improves explainability by providing information on the measurement quality of the prediction. Furthermore, the guaranteed invertibility of the neural network provides the use a direct way to check the neural network's understanding of the map by analyzing the reconstructed map. This feedback from the internals of the localization engine allows the human to know where in the map the network has lower confidence and that they should augment it with more data.

The expainability in planning is necessary to ensure the autonomous vehicle maintains a safe trajectory at all times in terms of maximizing the distance of a planned trajectory to obstacles and undrivable areas, maximizing velocity on the trajectories, minimizing the lateral acceleration the vehicle experiences on a planned trajectory, and so on. To achieve this, we introduced a new approach on Game-Theoretic Objective Space Planning where we map these complex planning objectives to a lower dimensional space of balancing aggressiveness and restraint. In a racing context, we show how our planner increases aggressiveness continuously to win the races. Similarly, it chooses to be more conservative to maintain safety. By moving the planning problem in a lower dimension that encodes interpretable metrics for humans we demonstrate how it doesn't decrease the capability of the planner, or reduce the completeness in the algorithm.

Finally, to ensure the plan is executed in a safe manner, we describe our efforts in explainable safe neural network controller design with Differentiable CBF-QP filters. The Control Barrier Function structure of the filter ensures the control output is always safe in an interpretable manner as there is a direct projection of the NN controller output into the safe control set. The performance of the proposed safe controllers is comparable to model-predictive controllers, which implies the high completeness from demonstrations.

Through these three learning-based architectures for localization, planning and safe control, we have demonstrated the initial findings on explainability of decisions in the sense-plan-act process of autonomous systems. Further developments in explainability of autonomous systems will lead to more public trust and safer systems.

References

[1] [Online]. Available: https://morningconsult.com/wp-content/uploads/2017/09/National-Tracking-Poll-170904.pdf

[2] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 2018, pp. 0210–0215.

[3] S. R. Islam, W. Eberle, S. K. Ghafoor, and M. Ahmed, "Explainable artificial intelligence approaches: A survey," *arXiv preprint arXiv:2101.09429*, 2021.

[4] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 11, pp. 4793–4813, 2020.

[5] T. Sakai and T. Nagai, "Explainable autonomous robots: a survey and perspective," *Advanced Robotics*, vol. 36, no. 5-6, pp. 219–238, 2022.

[6] É. Zablocki, H. Ben-Younes, P. Pérez, and M. Cord, "Explainability of deep vision-based autonomous driving systems: Review and challenges," *International Journal of Computer Vision*, pp. 1–28, 2022.

[7] S. Atakishiyev, M. Salameh, H. Yao, and R. Goebel, "Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for future research directions," *arXiv preprint arXiv:2112.11561*, 2021.

[8] D. Omeiza, H. Webb, M. Jirotka, and L. Kunze, "Explanations in autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[9] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 2018, pp. 80–89.

[10] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)*, vol. 2. IEEE, 1999, pp. 1322–1328.

[11] C. Zhang, M. H. Ang, and D. Rus, "Robust lidar localization for autonomous driving in rain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3409–3415.

[12] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss, "Range Image-based LiDAR Localization for Autonomous Vehicles," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[13] X. Chen, T. Läbe, L. Nardi, J. Behley, and C. Stachniss, "Learning an overlap-based observation model for 3d lidar localization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4602–4608.

[14] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," *IEEE pervasive computing*, vol. 2, no. 3, pp. 24–33, 2003.

[15] R. Dube, A. Cramariuc, D. Dugas, H. Sommer, M. Dymczyk, J. Nieto, R. Siegwart, and C. Cadena, "Segmap: Segment-based mapping and localization using data-driven descriptors," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 339–355, 2020.

[16] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 716–12 725.

[17] M. A. Uy and G. H. Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4470–4479.

[18] L. Sun, D. Adolfsson, M. Magnusson, H. Andreasson, I. Posner, and T. Duckett, "Localising faster: Efficient and precise lidar-based robot localisation in large-scale environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4386–4392.

[19] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-net: Towards learning based lidar localization for autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6389–6398.

[20] M. Cai, C. Shen, and I. Reid, "A hybrid probabilistic model for camera relocalization," 2019.

[21] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *2016 IEEE international conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4762–4769.

[22] H. Deng, M. Bui, N. Navab, L. Guibas, S. Ilic, and T. Birdal, "Deep bingham networks: Dealing with uncertainty and ambiguity in pose estimation," *International Journal of Computer Vision*, pp. 1–28, 2022.

[23] Z. Zang, H. Zheng, J. Betz, and R. Mangharam, "Local_inn: Implicit map representation and localization with invertible neural networks," *arXiv preprint arXiv:2209.11925*, 2022.

[24] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference." *J. Mach. Learn. Res.*, vol. 22, no. 57, pp. 1–64, 2021.

[25] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe, "Analyzing inverse problems with invertible neural networks," *arXiv preprint arXiv:1808.04730*, 2018.

[26] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe, "Guided image generation with conditional invertible neural networks," *arXiv preprint arXiv:1907.02392*, 2019.

[27] T. J. Adler, L. Ardizzone, A. Vemuri, L. Ayala, J. Gröhl, T. Kirchner, S. Wirkert, J. Kruse, C. Rother, U. Köthe *et al.*, "Uncertainty-aware performance assessment of optical imaging modalities with invertible neural networks," *International journal of computer assisted radiology and surgery*, vol. 14, no. 6, pp. 997–1007, 2019.

[28] T. Wehrbein, M. Rudolph, B. Rosenhahn, and B. Wandt, "Probabilistic monocular 3d human pose estimation with normalizing flows," in *International Conference on Computer Vision (ICCV)*, Oct. 2021.

[29] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[30] C. Winkler, D. Worrall, E. Hoogeboom, and M. Welling, "Learning likelihoods with conditional normalizing flows," *arXiv preprint arXiv:1912.00042*, 2019.

[31] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[33] H. Zheng, Z. Zhuang, J. Betz, and R. Mangharam, "Game-theoretic objective space planning," *arXiv preprint arXiv:2209.07758*, 2022.

[34] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret Minimization in Games with Incomplete Information," in *Advances in Neural Information Processing Systems*, vol. 20. Curran Associates, Inc., 2007.

[35] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[36] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.

[37] G. Chou, N. Ozay, and D. Berenson, "Safe output feedback motion planning from images via learned perception modules and contraction theory," *arXiv preprint arXiv:2206.06553*, 2022.

[38] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[39] A. Anand, K. Seel, V. Gjærum, A. Håkansson, H. Robinson, and A. Saad, "Safe learning for control using control lyapunov functions and control barrier functions: A review," *Procedia Computer Science*, vol. 192, pp. 3987–3997, 2021.

[40] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[41] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 708–717.

[42] S. Dean, A. J. Taylor, R. K. Cosner, B. Recht, and A. D. Ames, "Guaranteeing safety of learned perception modules via measurement-robust control barrier functions," *arXiv preprint arXiv:2010.16001*, 2020.

[43] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3387–3395.

[44] M. H. Cohen and C. Belta, "Approximate optimal control for safety-critical systems with control barrier functions," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 2062–2067.

[45] S. Yang, S. Chen, V. M. Preciado, and R. Mangharam, "Differentiable safe controller design through control barrier functions," *arXiv preprint arXiv:2209.10034*, 2022.

[46] D. Tabas and B. Zhang, "Computationally efficient safe reinforcement learning for power systems," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 3303–3310.

[47] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in neural information processing systems*, vol. 32, 2019.

[48] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

[49] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

MARS: a toolset for the safe and secure deployment of heterogeneous distributed systems

Giann Spilere Nandi[1], David Pereira[1], José Proença[1], José Santos[2], Lourenço A. Rodrigues[2], André Lourenço[2] and Eduardo Tovar[1]

***Abstract*—This work discusses the ongoing development of a toolset named MARS aimed to ease the process of safely deploying runtime verification monitors into distributed micro-ROS and ROS2 nodes. The work is motivated by a use case in the health and automotive domains and covers safety/security concerns around the manipulation of sensitive biometric data.**

***Index Terms*—Safety, Security, Runtime Verification, ROS2, micro-ROS**

I. Introduction

Artificial Intelligence (AI) has been one of the main drivers of innovation in recent years. Its contributions impacted, and continue to impact, several research fields and application domains, including automotive, robotics, and health [16].

Although efforts towards porting AI to extremely resource-constrained devices were made in recent years [1], a large portion of these devices cannot yet process complex AI algorithms in a timely manner locally. A common approach to overcome this limitation is to offload the more resource-intensive tasks to more robust distributed nodes. However, dependable distributed applications may still demand that all network participants comply with safety, security, and real-time requirements.

Developers implementing distributed systems that must comply with such requirements will inevitably face challenges such as how to configure nodes to communicate with other heterogeneous nodes; how to make sure that messages are not disclosed nor manipulated by unauthorized participants (especially in the context of resource-constrained devices); and how to guarantee that data collection and processing is correct and executed in a timely manner.

This work discusses the development of the ***M**onitoring **A**rchitecture **S**pecification **L**anguage* (MARS), a domain-specific language with an associated toolset, introduced on [12], capable of: (i) specifying ROS2-compatible distributed system architectures, (ii) specifying and generating runtime monitors that evaluate a system's correctness while also guaranteeing that their associated overhead will not disrupt the target system's safety requirements, and (iii) generating components that communicate through secure channels. By achieving these goals, we facilitate the process of safely deploying runtime verification monitors into real-time systems, providing a way for teams of engineers with no background in formal methods to deploy runtime monitors in their systems while also providing evidence that these monitors will not disrupt the system's safety with respect to its real-time constraints.

We contextualize and motivate the use of MARS with a use case in the health and automotive domains developed in the context of the VALU3S European R&D project in collaboration with CardioID, a Portuguese company specialized in the analysis and integration of human hearts' biometric data into innovative solutions. The use case builds up on top of a generic and adaptable architecture with a newly developed and publicly available TLS 1.3-based transport that encrypts and authenticates the communication between micro-ROS [17] and ROS2 [18] devices.

II. CardioWheel Use Case

We start by describing our motivating use case provided by CardioID, called *CardioWheel*, used to guide and validate the development efforts of the MARS language and toolset.

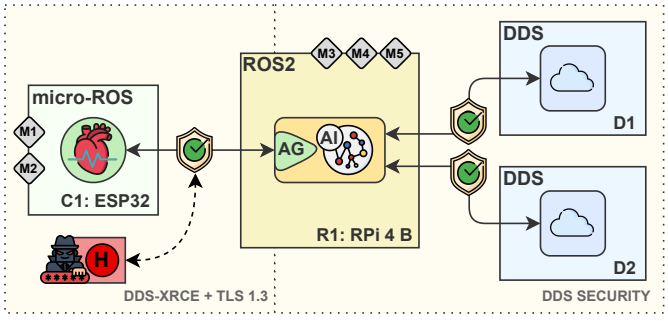


Fig. 1. Node `C1` collects, pre-processes, and transmits heart beat signals to `R1` over a wireless channel. `R1` is analyzes it and forwards results to a set of trusted nodes in the cloud, represented by `D1` and `D2`. The channel between `C1` and `R1` is secured by our custom transport based on TLS 1.3 and the DDS-XRCE standard.

In *CardioWheel*, the heart-beat signals of a vehicle's driver are collected, transmitted wirelessly, and analyzed by AI

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UID/CEC/04234); also by FCT within project ECSEL/0016/2019 and from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey. We would also like to thank Antonio Rodriguez and Pablo Garrido from eProsima for their collaboration in development of the secure custom transport layer of MARS.

[1]CISTER Research Centre in Real-Time and Embedded Computing Systems, Polytechnic Institute of Porto, Rua Alfredo Allen, 535, 4200-135, Porto, Portugal. {giann,drp,pro,emt}@isep.ipp.pt

[2]CardioID Technologies Lda, Instituto Superior de Engenharia de Lisboa, Rua Conselheiro Emídio Navarro 1, Room E.06 1959-007 Lisboa, Portugal {jfs, lar, arl}@cardio-id.com

algorithms. Figure 1 illustrates the system's underlying distributed architecture: a sensor, powered by an Espressif ESP32 microcontroller (C1), is placed in the steering wheel of the vehicle to collect and send aggregated data to a Raspberry PI 4 Model B (R1), which receives it and analyses it using a trained AI model while interacting with remote nodes D1/D2.

While the communication between R1 and D1/D2 complies with the underlying middleware [5] of the full fledged ROS2, C1 uses a stripped-down version called micro-ROS, specially developed to comply with the restrictions of (extremely) resource-constrained devices that have as little memory as 32kB of RAM and 256kB of flash and can run on top of real-time operating systems like FreeRTOS.[1] The information collected by C1 to allows R1 to provide:

- **Biometric Authentication:** by analyzing the user's electrocardiogram (ECG) signals, R1 can authenticate and validate a user's identity analogously to what is done with fingerprints [10].
- **Heart Monitoring:** although unique to every person, ECG signals share common traits that could indicate various body conditions, including heart anomalies, levels of fatigue, and emotional distress [10].

After authenticating the driver, the CardioWheel's inferred levels of fatigue could provide warning signals and potentially adapt parameters during assisted driving. It is important to mention that the quality of the analysis performed by R1 is heavily dependent on the quality of the signals fed to it, requiring C1 to provide low signal-noise readings at high frequency (around 1kHZ). Given the automotive context and the safety hazards that an incorrect system behavior could cause, CardioWheel deploys runtime monitors (M1-M5 in Fig. 1) to check for incorrect system behavior.

Due to the sensitive content of the transmitted data, security and privacy measures are required to protect against cyberattacks. While the communication of R1 with D1 and D2 can be secured by employing the DDS Security protocol [6], no similar option is available to the communication between C1 and R1, leaving the communication channel between C1 and R1 susceptible to attacks coming from a malicious user H, like **(i)** *tampering* with the data transmitted to R1, possibly leading genuine heart problems to be misclassified as a healthy heart behaviour; and **(ii)** *eavesdropping* on sensitive health information, disclosing a user's health condition. We address this lack of security with a newly developed transport, which is discussed in the next section.

III. MARS and Its Associated Toolset

Our work on MARS includes developing a *specification* language and a *back-end*. The specification language compactly and precisely captures both the distributed architecture (including communication channels) and a set of runtime monitors designed to verify safety properties that were too complex to verify statically. The back-end configures the communication channels with the required security levels, generates correct-by-construction runtime monitors, and safely instruments these monitors into a target system

[1]We point the reader to the current list of supported hardware by micro-ROS for further details: https://micro.ros.org/docs/overview/hardware/

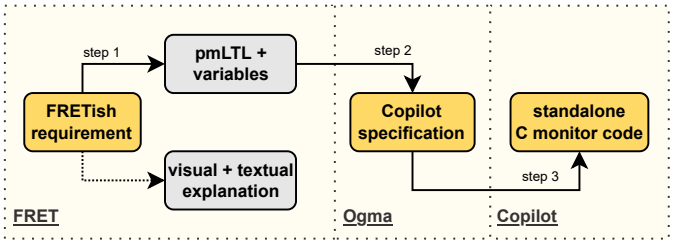


Fig. 2. Workflow adopted by the Ogma tool to synthesize runtime verification monitors from requirements written in the near-natural language of FRETish.

The design of the specification language is ongoing work, partially documented in previous work [12]. This paper focuses on the back-end, i.e., on how to: **(i)** generate runtime monitors from formal specifications, **(ii)** check if the computational overhead caused by these monitors does not disrupt the system's real-time schedulability, and **(iii)** introduce secure communication channels for resource-constrained devices within a micro-ROS/ROS2 distributed environment. Below we address each of these points and clarify the type of systems we support.

Supported Systems Although in the CardioWheel use case we focused on a single application compatible with MARS, we envision our work to easily suit multiple applications as long as they fit into a generic system model. More specifically, we support distributed systems whose nodes comply with the DDS-XRCE [7], the communication middleware for micro-ROS, and DDS [5], the communication middleware for ROS2. Supporting these standards allows excellent flexibility in designing distributed heterogeneous applications while adopting the widespread publish-subscribe message exchange pattern.

With micro-ROS, developers can have (extremely) resource-constrained devices actively interacting with other ROS2 and DDS-compatible nodes in the cloud by publishing and subscribing to topics of interest. On top of that, developers can enforce deterministic and timely behavior on micro-ROS nodes by employing scheduling algorithms to manage the execution of each node's task sets. Finally, MARS focuses on supporting the deployment of software monitors that, at runtime, verify if the system/application complies with a set of formal specifications provided by the user.

Correct-by-Construction Monitor Generation Manually specifying a monitor using formal semantics is error-prone and requires a formal background, usually not present in typical engineering teams. We facilitate the process of specifying and implementing runtime monitors by integrating into MARS tools like Ogma [13] and rmtld3synth [4] that synthesize C/C++ code from specifications written in near-natural languages.

To shed some light on the process behind it, we illustrate in Figure 2 the step-by-step workflow of Ogma to transform a

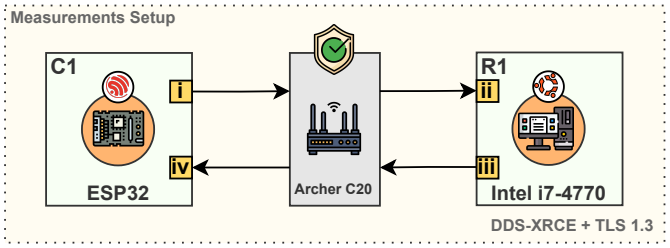


Fig. 3. Setup used to measure the round-trip delay of a message sent by C1. The figure illustrates the hardware used and points to the four additional steps performed by the system when compared to the default transport used by micro-ROS.

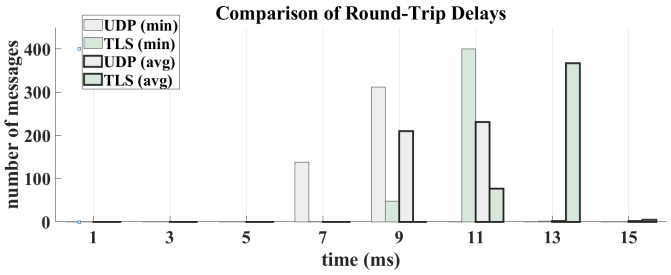


Fig. 4. Round-trip delays measured from 4500 messages sent over 10 identical experiments using the standard micro-ROS transport and 4500 messages sent over 10 identical experiments using our custom TLS 1.3-based transport. The presented histogram shows the minimum and the average round trip delay for each message among all the 10 runs of the experiment for each transport.

specification into a monitor. The workflow starts by receiving as input a requirement written in FRETish [3] describing what needs to be verified in the system and the variables that support such verification. FRET [3] then takes the requirement and variables and provides visual and textual explanations of how the requirement translates to Past-time Metric Linear Temporal Logic (pmLTL) [9]. Ogma then takes the pmLTL formula and associated variables and generates a Copilot [15] monitor specification. Finally, Copilot takes the specification as input and generates the standalone C monitor that MARS will use for its formal analyses and subsequent instrumentation and compilation.

Safe Instrumentation of Runtime Monitors Coupling runtime monitors into a system inevitably incur some computational overhead. In the case of real-time systems, adding a monitor means another task for the real-time operating system to schedule. Depending on its computational impact, adding a monitor could disrupt the system's schedulability and result in unsafe behavior. MARS addresses this concern by employing schedulability analysis algorithms that check the feasibility of scheduling the original task set in addition to the instrumented set of monitors.

Our initial efforts focus on the classical schedulability analysis of simple and static task sets. However, we are also investigating scenarios with systems supporting unbounded mode changes where tasks can be added/removed/have their scheduling parameters modified [2], [8]. For instance, we are currently experimenting with how to verify if a system is schedulable after an unbounded number of mode changes considering the residual accumulative impact that each mode transition can have on the overall schedulability analysis.

Secure Communication Channels MARS addresses the lack of security in the communication between micro-ROS and ROS2 nodes by including in its toolset a newly developed and open-source[2] transport based on TLS 1.3 developed in collaboration with eProsima.[3] The transport allows micro-ROS and ROS2 nodes to authenticate and encrypt their message exchanges with little to no modifications to the system's original behavior. The transport combines the open-source wolfSSL[4] library and the micro-ROS API for custom transports developed by eProsima.

Initial benchmarks, illustrated in Figure 4, show an approximate 2-millisecond difference (minimum and average) between the round-trip delay measured of a message sent using the default UDP-based transport of micro-ROS and our transport, which offers the reliability of TCP and the encryption and authentication of TLS 1.3. The hardware setup for the experiments consists of an ESP32, a TPLink router Archer 20, and an Ubuntu Desktop 20.04 equipped with an Intel 4770 processor.

The round-trip delay is the time difference between the timestamp of when C1 receives a message m_i and the timestamp when C1 sends the same m_i message. In total, we ran ten experiments using the UDP transport and ten experiments using our custom transport. In each experiment, we measured the round-trip delay of 450 messages. Figure 4 shows the minimum and the average of each m_i message sent in the network among the ten experiments of each transport layer such that $1 \geq i \leq 450$.

The approximate 2-milliseconds of difference between the two transport layers reflects the overhead caused by the additional reliability guarantees from TCP and the four extra steps, illustrated in Figure 3, that the system needs to perform after the handshake between C1 and R1: i) encryption of m_i by C1; ii) decryption of m_i by R1; iii) encryption of m_i by R1; iv) decryption of m_i by C1.

IV. Related Work and Conclusion

Several works in the literature address the automatic generation of runtime verification monitors from formal specifications. For instance, the work by Perez et al. [14] uses Ogma to generate monitors for ROS2 nodes, and the work of Meredith et al. [11] generates monitors for Java and hardware description languages. However, these approaches do not address the impact of generated software monitors in the context of real-time systems and are incompatible with commercial off-the-shelf microcontrollers and microprocessors, which is precisely the novelty of our work.

[2] https://bitbucket.org/mars-language/

[3] Developer and maintainer of micro-ROS.

[4] https://www.wolfssl.com/docs/tls13/

By having the harsh resource constraints of these devices in mind, this work presents the current status of the MARS toolset, which is meant to ease the safe instrumentation and deployment of runtime verification monitors in micro-ROS/ROS2/DDS-based distributed systems. More specifically, we discussed the back-end improvements that enable the generation of formally-specified monitors, the employment of formal verification algorithms on the integration of these monitors into real-time systems, and the newly developed secure transport for micro-ROS. The results were motivated and validated by an industrial use case representing the generic system model supported by MARS: a set of micro-ROS nodes with real-time and safety requirements that communicate via publish-subscribe message exchanges that comply with the underlying communication middleware ROS2 and other DDS-based nodes.

References

[1] Francesco Alongi, Nicolo Ghielmetti, Danilo Pau, Federico Terraneo, and William Fornaciari. Tiny neural networks for environmental predictions: An integrated approach with miosix. In *SMARTCOMP 2020*. IEEE.

[2] Hyeongboo Baek, Kang G. Shin, and Jinkyu Lee. Response-time analysis for multi-mode tasks in real-time multiprocessor systems. *IEEE Access*, 8:86111–86129, 2020.

[3] Esther Conrad, Laura Titolo, Dimitra Giannakopoulou, Thomas Pressburger, and Aaron Dutle. A compositional proof framework for fretish requirements. In *Proceedings of the 11th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2022, page 68–81, New York, NY, USA, 2022. Association for Computing Machinery.

[4] André de Matos Pedro, Jorge Sousa Pinto, David Pereira, and Luís Miguel Pinho. Runtime verification of autopilot systems using a fragment of MTL-$\int$. *International Journal on Software Tools for Technology Transfer*, 20(4):379–395, August 2017.

[5] OBJECT MANAGEMENT GROUP. Data distribution service (dds) specification version 1.4. *http://www.omg.org/spec/DDS/1.4*, 2015.

[6] OBJECT MANAGEMENT GROUP. Dds security specification version 1.1. *https://www.omg.org/spec/DDS-SECURITY/1.1/*, 2018.

[7] OBJECT MANAGEMENT GROUP. Dds for extremely resource constrained environments 1.0. *https://www.omg.org/spec/DDS-XRCE/1.0*, 2020.

[8] Wen-Hung Huang and Jian-Jia Chen. Techniques for schedulability analysis in mode change systems under fixed-priority scheduling. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, August 2015.

[9] Martin Leucker and César Sánchez. Regular linear temporal logic. In *Theoretical Aspects of Computing – ICTAC 2007*, pages 291–305. Springer Berlin Heidelberg, 2007.

[10] André Lourenço, Ana Priscila Alves, Carlos Carreiras, Rui Policarpo Duarte, and Ana Fred. CardioWheel: ECG biometrics on the steering wheel. In *Machine Learning and Knowledge Discovery in Databases*, pages 267–270. Springer International Publishing, 2015.

[11] Patrick O'Neil Meredith, Dongyun Jin, Dennis Griffith, Feng Chen, and Grigore Roşu. An overview of the MOP runtime verification framework. *International Journal on Software Tools for Technology Transfer*, 14(3):249–289, April 2011.

[12] Giann Spilere Nandi, David Pereira, José Proença, and Eduardo Tovar. Work-in-progress: a DSL for the safe deployment of runtime monitors in cyber-physical systems. In *RTSS 2020, USA 2020*. IEEE, 2020.

[13] Ivan Perez, Anastasia Mavridou, Tom Pressburger, Alwyn Goodloe, and Dimitra Giannakopoulou. Automated translation of natural language requirements to runtime monitors. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 387–395. Springer International Publishing, 2022.

[14] Ivan Perez, Anastasia Mavridou, Tom Pressburger, Alexander Will, and Patrick J. Martin. Monitoring ros2: from requirements to autonomous robots. 2022.

[15] Lee Pike, Sebastian Niller, and Nis Wegmann. Runtime verification for ultra-critical systems. In *International Conference on Runtime Verification*, pages 310–324. Springer, 2011.

[16] Pramila P. Shinde and Seema Shah. A review of machine learning and deep learning applications. In *ICCUBEA*. IEEE, August 2018.

[17] Jan Staschulat, Ralph Lange, and Dakshina Narahari Dasari. Budget-based real-time executor for micro-ros, 2021.

[18] George Stavrinos. ROS2 for ROS1 users. In *Studies in Computational Intelligence*, pages 31–42. Springer International Publishing, August 2020.

Towards a Systematic Analysis of Timing Aspect of Safety-Security Interactions

Elena Troubitsyna
Dept. of Computer Science
KTH – Royal Institute of Technology
Stockholm, Sweden
elenatro@kth.se

***Abstract*—Modern safety-critical control systems are increasingly relying on networking to provide not only advanced intelligent services but also safety-critical functions. Since they often utilize standard network protocols, the systems inherently become susceptible to cyberattacks that can potentially jeopardize safety. In this paper, we discuss how to systematically analyze safety-security interactions using Data Flow Diagrams and HAZOP and define the corresponding patterns of safety cases. Our approach allows us not only to identify the impact of cyberattacks on safety but also analyze whether security control mechanism introduce acceptable from safety point of view time overhead.**

***Index terms* – safety; cyberattacks; safety case; security control; real-time requirements**

I. INTRODUCTION

Modern safety-critical systems become increasingly networked and interconnected. Connectivity offers a number of significant business and technological benefits, such as remote diagnostics and predictive maintenance, improved resource utilisation as well as process optimisations. However, since the systems utilise the standard networked protocols, they also become susceptible to cyber threats. Cyberattacks might lead to loss of control and situation awareness and jeopardise system safety per se. Therefore, there is a clear need for the techniques facilitating a systematic analysis of safety and security interactions.

Currently safety cases are usually used to argue about system safety. A safety case is "a structured argument, supported by a body of evidence that provides a convincing and valid case that a system is safe for a given application in a given operating environment" [1]. Usually a main strategy for constructing a safety case is to demonstrate that all hazards associated with the system functions are mitigated.

While safety case is inherently functionality-oriented, security analysis is typically data-centric, i.e., it aims at determining the impact of security attacks on the system data flow. Hence, by using security analysis alone, it becomes cumbersome to uncover the interactions between safety and security, e.g., assess whether the introduced security control mechanisms support or contradict functional and timing safety requirements.

In this work, we aim at supporting a systematic integrated analysis of safety-security interactions. We rely on DFDs -- Data Flow Diagrams [2] and HAZOP -- Hazard and Operability Study [3] to analyze the impact of not only of cyberattacks and accidental failures on safety but also the impact of introducing security control mechanism on safety-related timing requirements. We use such an integrated analysis to define patterns for fragments of safety cases that can be used to argue that safety-security interactions do not jeopardize safety.

II. SAFETY-SECURITY INTERACTIONS IN NETWORKED CONTROL SYSTEMS

The top-level safety goal of many safety-critical systems is to keep some physical parameter p within the predefined boundaries while providing the intended system functionality.

Let p correspond to the physical value of such a parameter. Then the safety invariant can be formulated as

$$p\_low\_threshold \leq p \leq p\_high\_threshold$$

To achieve this goal, the system builds a model of the controlled environment and computes the corresponding actions. Hence, safety can be achieved if the systems' model of the controlled environment is sufficiently close to reality, the control algorithm is correct and the actuation is performed according to the controller's commands.

A safety case is often described in the Goal Structuring Notation [1] – a graphical notation explicitly representing how goals (the claims about system safety) are decomposed into subgoals until the claims can be supported by the direct evidences. The fragment of top-level safety case in Goal-Structuring Notation [1] formalizing these requirements is shown in Fig.1. Here

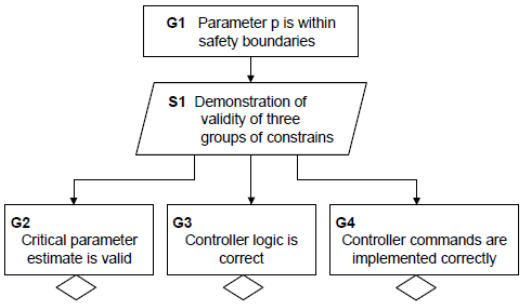


Figure 1. Fragment of top-level safety case

rectangles depict goals, while parallelograms contain decomposition strategies.

By creating a formal specification of the controlling algorithm and verifying its correctness, we can demonstrate achieving the goal **G3**. To justify **G2**, we need to explicitly define a procedure for computing *p* estimates in the presence of non-malicious and malicious faults. To address **G4**, we should investigate how faults might cause deviations in implementing of the controller actions. This requires an analysis of system architecture.

A networked control system usually consists of several components, where each component is a real-time control system that follows the generic architectural pattern shown in Fig.2.

The system behavior is cyclic. The intended functionality is achieved by changing the state of the actuator that influences the value of controlled parameter based on the readings obtained by sensor. Since we consider a networked control system, sensing and actuation can be remote, i.e., the sensor transmits its readings to the controller via the channel **s_c_chan** and the actuator receives the commands from the controller via the **c_a_chan.**

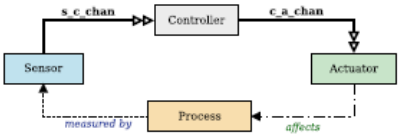


Figure 2. Generic architecture of control system

At each cycle, the system goes through the sequence of steps represented by a data flow diagram (DFD) shown in Fig.3. DFDs give a graphical representation of the flow of information for a given system [3]. At each cycle, the system alternates between the control actions and physical process reaction and hence, it is convenient to represent the physical process as the source and sink of the dataflow. Rectangles depict the system components and the communication channels that consume and produce data attributes shown by the incoming/outgoing arrows.

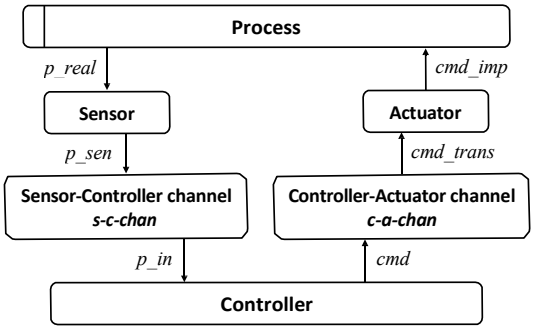


Fig. 3. A data flow diagram

We tailor the well-known safety analysis technique HAZOP – HAZard and OPerability Analysis [3] and integrate it with the data flow analysis to identify the impact of not only cyberattacks and failures on data flow, but also the impact of the introduced security control mechanism of safety-related timing requirements.

The analysis is performed using a set of guidewords and attributes. The original set of guidewords includes *no, more, less, as well as, part of, reverse, other than, early, late, before* and *after*. The guidewords are applied to the system attributes, which allows a team of experts to identify the deviations that might affect safety.

The majority of the guidewords are self-explanatory. The guidewords *REVERSE/INSTEAD* represent the deviations over the discrete data types. If they are applied to booleans then *REVERSE* would represent replacements of *TRUE* by *FALSE* and vise versa; if they are applied to sets then *INSTEAD* would represent replacing one constant of the set with another (e.g., *ON* by *OFF*).

DFD shows the relationships between the system components and the produced and consumed data flow. We propose an interpretation of standard HAZOP guidewords as presented in Fig. 4.

NO	Data is missing
MORE/LESS	Value of data increased/decreased
REVERSE/INSTEAD	Substitution of intended data value to logical opposite
EARLY/LATE	Data available before/after intended

Figure 4. Interpretation of guide words

We start by selecting the data attributes and identifying components responsible for producing or modifying the selected attribute. Then we apply relevant HAZOP guidewords and identify possible deviations as well as identify the causes. Finally, we analyze the impact on safety by performing cause-consequence analysis and construct the corresponding fragment of safety case. An example of application of the proposed approach is shown in Fig. 5.

Attribute	Guide-word	Consequence	Causes
Sensor reading p_sen	MORE/ LESS	The value *p_sen* is greater or smaller than the physical value *p_real*	Sensor stuck at (high/low) Loss of sensor precision Tampering attack
	LATE	Delay in sensing process state	Network congestion Authentication causes delay

Figure 5. An example of analysis applied to sensor reading

Attribute	Guidewor	Consequence	Causes
cmd_trans	NO	No command received by the actuator	DOS attack on controller-actuator channel Omission channel failure
	INSTEAD	Actuator receives opposite command	Spoofing controller identity Tampering with channel data
	LATE	Delay in transmission	Network congestion Authentication causes delay

Figure 6. An example of analysis applied to controller commands

For the deviation "The value *p_sen* is greater (or smaller) than the real value of *p* ", we can define the requirements addressing the causes of this deviations as follows: "Controlling software should check reasonableness of received input data and use indirect data or calculated predicted value if the received data is outside of reasonable range." For the deviation "Late sensing", we can define the requirements related to the monitoring of the network conditions. Moreover, we should perform WCET analysis of the authentication procedure and evaluate whether the sum of the worst possible delay in sensing and WCET of authentication procedure fulfil the real-time requirements of the overall control system.

For the deviations identified for the controller-actuator channel, several different types of requirements could be proposed. In case the actuator receives no command from a controller, either it should execute a safe shutdown or, if additional non-programmable measures are implemented to enforce safety, maintain its previous state. Timing analysis is required to identify the time interval during which such state is safe to maintain. For the deviation caused by the tampering or spoofing, secure gateway should be implemented. However, timing analysis should be performed to establish that the time delay does not invalidate safety requirements.

The proposed methodology allows us to derive safety and security requirements for controlling software and overall system architecture. However, often such requirements are mutually interdependent. For instance, while introducing a secure gateway for the controller-actuator channel, we also introduce a delay in transmitting the control commands. Such a delay should be taken into account while calculating the state of the actuator.

III. RELATED WORK

The problem of safety-security interactions is gaining an attention of research community. Gran et.al [4] proposed to adapt HAZOP to security analysis. In our work, we not only integrate safety and security analysis but also provide a support for identifying whether security control mechanisms contradict timing safety requirements by introducing safety-jeopardising delays. Young and Leveson [5] extend STAMP to address safety concerns. However, they approach does not support a construction of safety cases that are widely used in current development practice.

IV. CONCLUSIONS

In this work, we addressed the problem of an integrated systematic analysis of safety-security interactions. We have proposed an adaptation of HAZOP guide words to analyse the impact of cyberattacks, faults and security control mechanisms, such as authentication or secure gateways on functional and timing safety requirements. The analysis provides an input for the construction of system safety case in a security-aware way.

References

[1] T. Kelly, R.Weaver. The goal structuring notation–a safety argument notation. In Proceedings of the dependable systems and networks 2004 – workshop on assurance cases.2004.

[2] P.D. Bruza and Th. P. van der Weide. The Semantics of Data Flow Diagrams. Technical Report 89-16, University of Nijmegen, The Netherlands, 1989.

[3] Ministry of Defence. Interim Defence Standard 00-58/1: Hazop Studies on Systems Containing Programmable Electronics. In *Directorate of Standardization*, 1994.

[4] Gran B.A. Winther R., Johnsen OA. Security Assessments of Safety Critical Systems Using HAZOPs. In *Computer Safety, Reliability and Security. SAFECOMP 2001*, volume 2187 of *LNCS*, pages 24–39. Springer, 2001.

[5] W. Young and N. G. Leveson. An Integrated Approach to Safety and Security Based on Systems Theory. *Commun. ACM*, 57(2):31–35, 2014.

[6] A. Iliasov, A. Romanovsky, L. Laibinis, E. Troubitsyna, T. Latvala. Augmenting Event-B modelling with real-time verification. In Proc. of FormSERA: 2012 First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches, pp. 51-57, IEEE.

[7] L. Laibinis, B. Byholm, I. Pereverzeva, E. Troubitsyna, K. E. Tan, I. Porres. Integrating Event-B Modelling and Discrete-Event Simulation to Analyse Resilience of Data Stores in the Cloud. In Proc. of IFM 2014, LNCS, 8739, pp. 103-119, Springer, 2014.

[8] A.Iliasov, E.Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, D. Ilic, T. Latvala: Supporting reuse in event B development: modularisation approach. In: M.Frappier, U. Gläasser, S.Khurshid, R.Laleau, S.Reeves. (eds.) ABZ 2010. LNCS, vol. 5977, pp. 174–188. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11811-1_14

[9] A,Iliasov, E.Troubitsyna, L. Laibinis, A. Romanovsky, K. Varpaaniemi, P. Väisänen, D. Ilic, T. Latvala: Developing mode-rich

satellite software by refinement in Event-B. Sci. Comput. Program. 78(7), 884–905 (2013)
[10] LLaibinis, E.Troubitsyna: A contract-based approach to ensuring component interoperability in Event-B. In: Petre, L., Sekerinski, E. (eds.) From Action Systems to Distributed Systems - The Refinement Approach, pp. 81–96. Chapman and Hall/CRC (2016)
[11] I.Rauf, E. Troubitsyna: Generating cloud monitors from models to secure clouds. In: DSN 2018. IEEE Computer Society (2018)
[12] E. Troubitsyna, L.Laibinis, I.Pereverzeva, T. Kuismin, D.Ilic, T.Latvala: Towards security-explicit formal modelling of safety-critical systems. In: A.Skavhaug, J.Guiochet, F.Bitsch (eds.) SAFECOMP 2016. LNCS, vol. 9922, pp. 213–225. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45477-1 17
[13] E.Troubitsyna, I.Vistbakka: Deriving and formalising safety and security requirements for control systems. In: Gallina, B., Skavhaug, A., Bitsch, F. (eds.) SAFECOMP 2018. LNCS, vol. 11093, pp. 107–122. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99130-6 8
[14] I.Vistbakka, E.Troubitsyna: Towards a formal approach to analysing security of safety-critical systems. In: EDCC 2018, pp. 182–189. Computer Society (2018)
[15] I.Vistbakka, E.Troubitsyna, E.: Pattern-based formal approach to analyse security and safety of control systems. In: Papadopoulos, Y., Aslansefat, K., Katsaros, P.,Bozzano, M. (eds.) IMBSA 2019. LNCS, vol. 11842, pp. 363–378. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32872-6 24

Real-Time Byzantine Resilient Power Grid Infrastructure: Evaluation and Trade-offs

Sahiti Bommareddy, Maher Khan, David J Sebastian Cardenas, Carl Miller,
Christopher Bonebrake, Yair Amir, and Amy Babay
Department of Computer Science, Johns Hopkins University — {sahiti, yairamir}@cs.jhu.edu
School of Computing and Information, University of Pittsburgh — {maherkhan, babay}@pitt.edu
Pacific Northwest National Labs (PNNL) — {d.sebastiancardenas, carl.miller, christopher.bonebrake}@pnnl.gov

***Abstract*—Increasing threats to power grid infrastructure are driving the need to build Byzantine-resilient systems that can continue to operate correctly despite failures and attacks. However, the real-time requirements of power grid infrastructure call for a more rigorous evaluation of Byzantine resilient systems than the traditional evaluations performed in the context of standard IT applications. We discuss these requirements, and the potential of commercial-off-the-shelf and open source solutions to support real-time resilient systems.**

I. Introduction

The rising number of cyberattacks against critical infrastructure reinforces the need to build Byzantine resilient power grid infrastructure [1], [2]. While Byzantine resilient techniques have been developed in the context of IT applications [3], applying them in the power grid domain brings the need to consider strict real-time requirements. Supervisory Control and Data Acquisition (SCADA) operations in power grid control centers typically require latency of 100-200ms [4], [5], and the requirements become much more stringent as we move from control centers to substations that carry out critical protection functions, due to the physical properties of the system.

During a fault condition, currents flowing into the faulted zone experience a rapid surge (many orders of magnitude greater than during normal conditions). This sudden increase in current is capable of damaging grid equipment in just a few electrical cycles (in the US, a cycle is 16.667ms). Therefore, according to industry and standardization bodies (IEEE, IEC), fault identification must occur within a quarter power cycle, i.e. 4.167ms [4].

Figure 1 shows a fault condition, where the current swelling behavior can be observed. In a well-designed system, a quick relay response enables the circuit breaker to have more time to safely dissipate the energy present within the system. Due to the amount of energy being released, circuit breakers can take 1-3 cycles to fully interrupt the fault/arcing current, drastically reducing the time to fault identification. If this fault is not identified quickly, damage to physical equipment, and cascading escalations to otherwise healthy systems may be unavoidable. Damaged grid equipment like the 345kV transformer can cost millions of dollars and have lengthy replacement process in the orders of years. Therefore, meeting the 4.167ms requirement is critical.

Most existing Byzantine resilient techniques are not designed for or evaluated under these strict real-time requirements. Hence, the first goal of our work is to investigate the type of evaluation needed to provide confidence that Byzantine-resilient systems can meet the real-time requirements of power grid infrastructure in practice.

The second goal of our work is to use the rigorous evaluation strategy we propose to explore new deployment trade-offs. In particular, we investigate the potential of commercial-off-the-shelf (COTS) and open-source solutions to support real-time Byzantine-resilient infrastructure. Compared to specialized real-time systems, COTS and open-source systems are less expensive, easier to deploy and manage, and easier to audit [6], making them an attractive option.

II. Landscape of BFT SMR Evaluation

Byzantine Fault Tolerant State Machine Replication (BFT SMR) is a classical approach to tolerate intrusions in networked systems. Initial works on BFT SMR focused on proving that the techniques are correct and practical for IT applications like file systems and storage [3], [7]. As the research matured, performance became a key focus [8]–[11], but most works focused on normal-case throughput and scaling evaluations. Some of these works expanded evaluation to include malicious behavior [8], [11]–[14], and even designed protocols to maintain performance under attack [12], [15], [16], but their empirical evaluations are still based on IT domain requirements. Even in more recent BFT SMR protocols designed for permissioned blockchains, the focus of evaluations remains on throughput scalability and blockchain requirements [17]–[19]. Frameworks have been proposed to evaluate and compare the performance of BFT algorithms, but they generally focus on quantifying performance analytically, based on the number of cryptography operations or messages exchanged, rather than on empirical evaluation [20], [21].

III. Evaluating Byzantine-Resilient Power Grid Infrastructure

To gain confidence that a system can meet real-time requirements in practice, it is necessary to (1) consider the worst-case number of constraint violations, not only average latency, (2) evaluate the most demanding operating conditions that the system is expected to work under, including all failure/attack

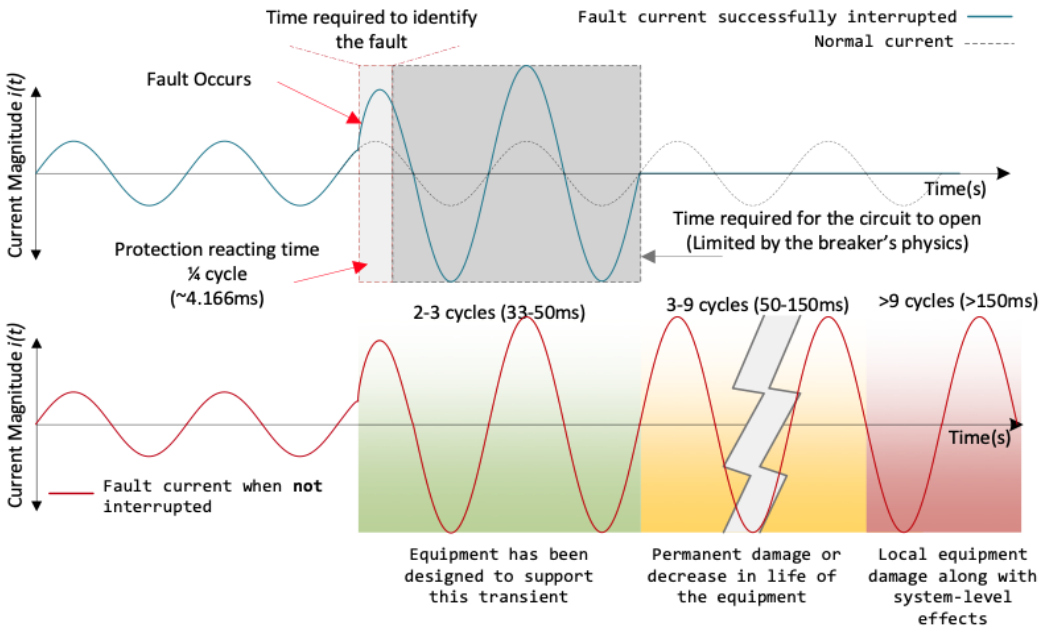


Fig. 1. Fault detection timing requirements in substation

TABLE I
PERFORMANCE IN DIFFERENT OPERATING CONDITIONS WITH FOUR RELAY NODES ($f = 1, k = 1$)

Operating Condition	Normal Kernel (microseconds)			Real-Time Kernel (microseconds)		
	Minimum	Average	Maximum	Minimum	Average	Maximum
Fault-Free (Normal)	**1723**	**2187**	**3323**	**1637**	**1950**	**3596**
Fail-Stop Fault or Proactive Recovery	**1871**	**2260**	**3326**	**1608**	**1976**	**3726**
Fail-Stop Fault and Proactive Recovery	**1912**	**2328**	**7617** (8*)	**1750**	**2015**	**3996**
Byzantine Fault	**1737**	**2227**	**3785**	**1665**	**1984**	**4002**
Byzantine Fault and Proactive Recovery	**1867**	**2313**	**7699** (6*)	**1767**	**2019**	**4101**

* The count of actions that crossed 4.167 milliseconds (out of 1 million total actions)

cases, (3) evaluate each operating condition over a sufficiently long period of time, and (4) use a realistic testbed.

Our Spire intrusion-tolerant SCADA system took a step towards evaluating real-time requirements at the control-center level [22]. It used an extensive 30-hour evaluation on a real wide-area network for the normal-case, and shorter under-attack evaluations in a cluster environment with emulated wide-area latencies, analyzing how many updates failed to meet the target latency of 100-200ms [22].

However, at the substation level, even an evaluation like the one in [22] is not sufficient, as shown by our work on Spire for the Substation, a real-time Byzantine-resilient system for substation protection [23]. In Spire for the Substation, we developed two Byzantine-resilient protocols: Arbiter Protocol and Peer Protocol. Both use four relay nodes to simultaneously tolerate one Byzantine relay and one proactive recovery but have different deployment tradeoffs [23]. In that work, we rigorously evaluated both protocols, running each failure/attack case for 24 hours, processing 1 million actions. That evaluation showed that while the Peer Protocol has a smaller attack surface and can seamlessly integrate into the substation, its tradeoff is a slight risk of not meeting the latency requirement under the most demanding attack scenarios.

Figure 6 illustrates the importance of this extensive evaluation. Running the Peer Protocol for 1 million actions with a simultaneous Byzantine fault and proactive recovery, we see that 6 actions cross the 4.167ms threshold. While an evaluation with 100,000 under-attack actions (taking hours) would be significantly more extensive than those of many BFT systems (see Section II), it would not necessarily show these constraint violations. For example, in Figure 6, in the slices from $[320000, 520000]$, $[620000, 720000]$, $[720000, 820000]$, $[820000, 920000]$, we see intervals of 100,000 actions or more with none crossing 4.167ms. The 1-million-action evaluation is needed for an accurate view of the protocol's performance.

Given the Peer Protocol's tradeoff of better deployment properties vs the risk of not meeting real-time requirements, one possibility is to explore specialized real-time hardware and/or software. However, maintaining a COTS hardware and open-source software environment simplifies system management and maintenance [24], [25]. Therefore, we explore a different trade-off, evaluating whether the Linux real-time kernel option can achieve the needed performance.

Real-time Kernel Evaluation. We deployed Spire for the

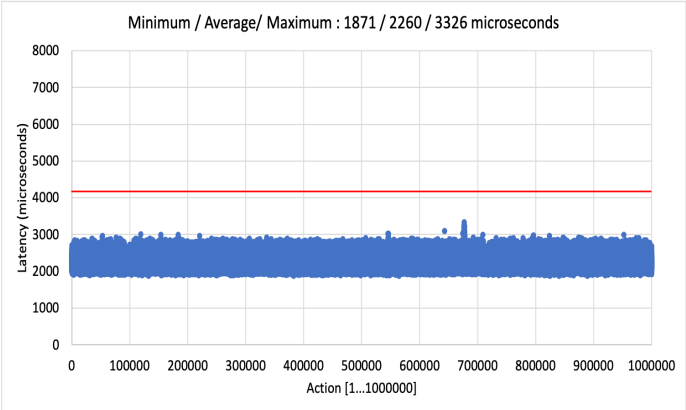


Fig. 2. Normal Kernel: Fail-Stop Fault

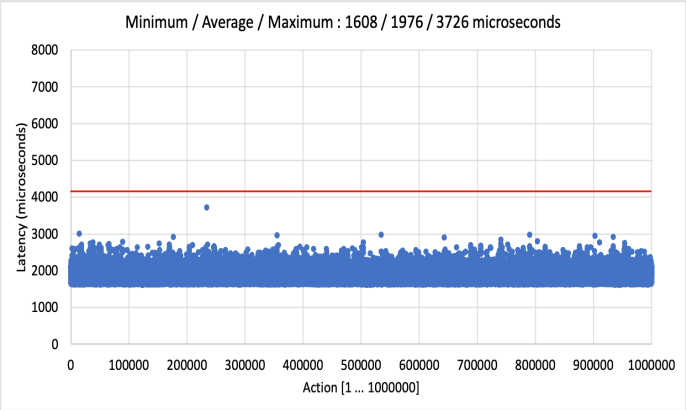


Fig. 3. Real-Time Kernel: Fail-Stop Fault

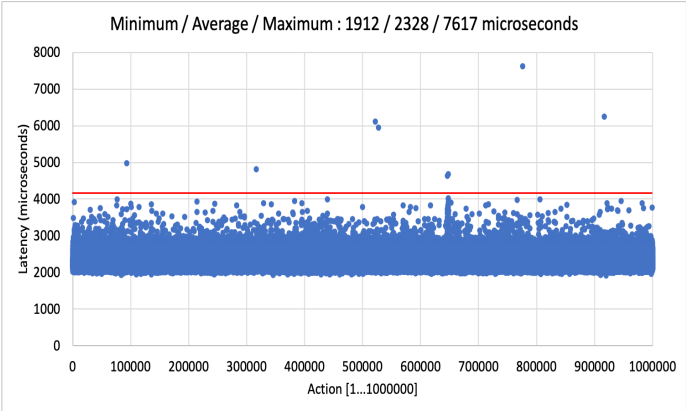


Fig. 4. Normal Kernel: Fail-Stop Fault with Proactive Recovery

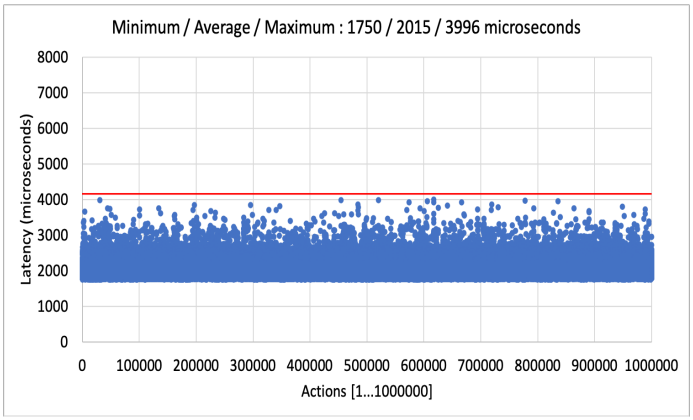


Fig. 5. Real-Time Kernel: Fail-Stop Fault with Proactive Recovery

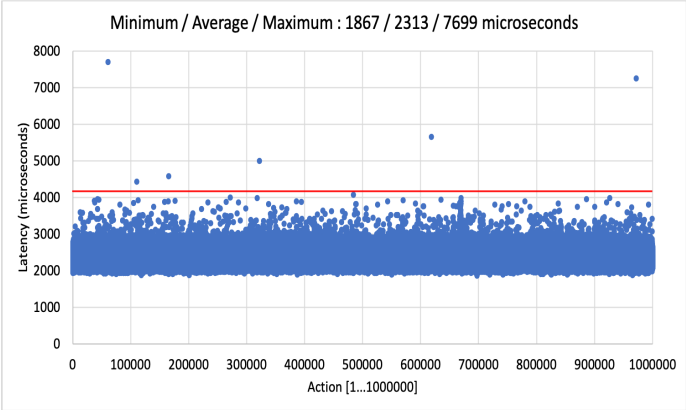


Fig. 6. Normal Kernel: Byzantine Fault with Proactive Recovery

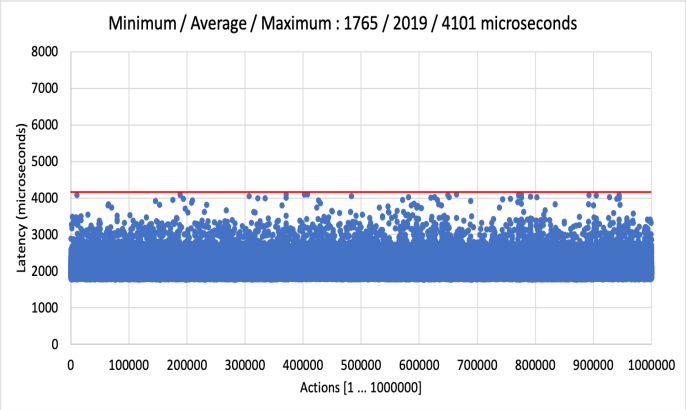


Fig. 7. Real-Time Kernel: Byzantine Fault with Proactive Recovery

Substation (Peer Protocol) in our testbed with four Intel Xenon E3 servers with 16GB RAM running CentOS 8 and connected by a 1Gbps network[1]. The deployed system consists of four relay nodes and therefore can simultaneously tolerate up to one faulty relay node (fail-stop or Byzantine) and up to one additional relay node undergoing proactive recovery. Using the same 24-hour, 1-million-action evaluation strategy described above, we compared the system using the normal Linux kernel, and with the real-time kernel patch and tuning on the same machines. Table I reports the minimum, average and maximum latency in microseconds for all five operating

[1]This is a different testbed than the one in [23]. Although the results show minor quantitative differences between the two testbeds, their qualitative profiles are the same.

conditions supported by the threat model for both the normal and real-time kernels.

While Table I summarizes the results, Figures 4, 5, 6 and 7 focus on the most demanding scenarios: Fail-Stop fault with simultaneous proactive recovery and Byzantine fault with simultaneous proactive recovery. In the former, one of the relay nodes is unavailable due to a fail-stop fault while, simultaneously, an additional relay node is undergoing proactive recovery. In the latter, the Byzantine relay node performs two simultaneous attacks for each action. First, the Byzantine relay node sends a corrupt message that will not be useful. Second, the Byzantine relay node performs a short intermittent denial of service attack on the other relay nodes to consume their network and computational resources further. In both of these operating conditions, the proactive recovery node is not rejuvenated for the entire test duration, so there are only two correct relay nodes available throughout the test. We refer to this condition with only two correct relay nodes available as non-optionality condition.

Under non-optionality, meeting the latency requirement is particularly tough. With only two nodes available, a random delay on either node (e.g., from network delays, kernel scheduling, or even effects of a Byzantine node's actions) would be reflected in the end-to-end latency. Compare this situation to Fail-Stop fault condition in which three nodes are available. In such a case, delays would have to occur on two of the three nodes independently and at the same time in order to be reflected in the final latency. For both the normal and real-time kernel, comparison of their Fail-Stop fault condition performance (Figures 2 & 3) to that of Fail-Stop fault with simultaneous proactive recovery condition (Figures 4 & 5) exemplify this effect i.e., Figure 2 vs Figure 4 and Figure 3 vs Figure 5.

The normal linux kernel is optimized for throughput and fair scheduling of tasks, while the real-time kernel is optimized to maintain low latency, consistent response time and determinism. These characteristics are particularly important in the conditions with non-optionality where any random delay impacts the system performance. Due to its features, in real-time kernel benchmarks, when we use high priority and FIFO scheduling policy, all actions meet the 4.167ms requirement, and average latency is reduced by about 300 microseconds across all operating conditions (Table I). These observations can also be immediately noted by comparing benchmark plots of normal kernel to those of real-time kernel in the three operating conditions shown: Fail-Stop fault (Figure 2 vs Figure 3), Fail-Stop fault with simultaneous proactive recovery (Figure 4 vs Figure 5), and Byzantine fault with simultaneous proactive recovery (Figure 6 vs Figure 7) .

The determinism and latency stability of real-time kernel offers a new deployment option for real-time Byzantine resilient critical infrastructure, enabling us to get the benefits of the Peer Protocol while meeting real-time requirements.

COTS and Open Source Discussion. The real-time kernel option enables us to remain in the more flexible open-source realm while still meeting real-time requirements, which is of increasing interest in the power industry. The protective relay is a relatively expensive device (tens of thousands of dollars). Each substation has multiple relays employed for protection schemes. There are hundreds to thousands of substations across a country (e.g., U.S has over 55,000 substations). Hence, to reduce cost, there is a high incentive to limit the additional relays needed in a Byzantine resilient scheme.

In a deployment, each relay node discussed above consists of a protective relay device directly connected to the Spire for the substation harness. Protective relays are typically specialized hardware relays with proprietary software implementing the protection functions. As we need four relays in the four relay nodes to tolerate an intrusion, the system will be costly. To reduce the cost, we experimented by substituting a hardware protective relay in one of the relay nodes with a software relay. The software relay is implemented on an Intel NUC (CentOS) with real-time kernel to perform the same protection function as hardware protective relays. In fact, the system successfully underwent red team experiments with three hardware relays (from GE, Siemens and Hitachi Energy) and a single software relay (on a different test bed at Pacific Northwest National Laboratory). However, the implications of using a real-time kernel on other processes running on the nodes, overall throughput and the deployment environment needs further investigation.

IV. Conclusion

We have presented the need for rigorous evaluation of Byzantine-resilient systems with respect to the real-time requirements of critical infrastructure. We also investigated the potential of COTS and open source systems to support real-time Byzantine resilient power grid infrastructure.

Acknowledgement

This work was supported in part by the Department of Energy (DOE) Offices of Cybersecurity, Energy Security, and Emergency Response (CESER); Electricity (OE); and Nuclear Energy (NE) under the Grid Modernization Laboratory Consortium (GMLC) Topic 5.1.4 – Cyber-Physical Security. Its contents are solely the responsibility of the authors and do not represent the official view of DOE.

This work was also supported in part by the DoD Strategic Environmental Research and Development Program (SERDP) grant RC20-1138.

Yair Amir is a co-founder and member of and holds equity in Spread Concepts LLC. The results discussed in this paper could affect the value of Spread Concepts LLC. This arrangement has been reviewed and approved by the Johns Hopkins University in accordance with its conflict of interest policies.

References

[1] "Cost of a data breach 2022." [Online]. Available: https://www.ibm.com/reports/data-breach

[2] T. Group, "2022 thales data threat report." [Online]. Available: https://cpl.thalesgroup.com/critical-infrastructure-data-threat-report

[3] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[4] IEEE, "Ieee standard communication delivery time performance requirements for electric power substation automation," *IEEE Std 1646-2004*, pp. 1–24, 2005.

[5] J. Deshpande, A. Locke, and M. Madden, "Smart choices for the smart grid," *Alcatel-Lucent Technolgy White Paper*, 2011.

[6] R. Hat, "The state of enterprise open source," 2021.

[7] L. Lamport, "Paxos made simple," *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58, 2001.

[8] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-scalable byzantine fault-tolerant services," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 5, pp. 59–74, 2005.

[9] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shrira, "Hq replication: A hybrid quorum protocol for byzantine fault tolerance," in *Proceedings of the 7th symposium on Operating systems design and implementation*, 2006, pp. 177–190.

[10] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 bft protocols," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 4, pp. 1–45, 2015.

[11] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, 2007, pp. 45–58.

[12] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Prime: Byzantine replication under attack," *IEEE transactions on dependable and secure computing*, vol. 8, no. 4, pp. 564–577, 2010.

[13] P.-L. Aublin, S. B. Mokhtar, and V. Quéma, "Rbft: Redundant byzantine fault tolerance," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 2013, pp. 297–306.

[14] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 355–362.

[15] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making byzantine fault tolerant systems tolerate byzantine faults." in *NSDI*, vol. 9, 2009, pp. 153–168.

[16] G. S. Veronese, M. Correia, A. N. Bessani, and L. C. Lung, "Spin one's wheels? byzantine fault tolerance with a spinning primary," in *2009 28th IEEE International Symposium on Reliable Distributed Systems*. IEEE, 2009, pp. 135–144.

[17] A. Bessani, J. Sousa, and M. Vukolić, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *Proceedings of the 1st workshop on scalable and resilient infrastructures for distributed ledgers*, 2017, pp. 1–2.

[18] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 31–42.

[19] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, pp. 347–356.

[20] J. R. Clavin, Y. Huang, X. Wang, P. M. Prakash, S. Duan, J. Wang, and S. Peisert, "A framework for evaluating bft," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2021, pp. 193–200.

[21] D. Gupta, L. Perronne, and S. Bouchenak, "Bft-bench: Towards a practical evaluation of robustness and effectiveness of bft protocols," in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2016, pp. 115–128.

[22] A. Babay, T. Tantillo, T. Aron, M. Platania, and Y. Amir, "Network-attack-resilient intrusion-tolerant scada for the power grid," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 255–266.

[23] S. Bommareddy, D. Qian, C. Bonebrake, P. Skare, and Y. Amir, "Real-time byzantine resilience for power grid substations," in *41st International Symposium on Reliable Distributed Systems*, 2022, pp. 213–224.

[24] B. Fitzgerald, "The transformation of open source software," *MIS quarterly*, pp. 587–598, 2006.

[25] "S.4913 - 117th congress (2021-2022): Securing open source software act of 2022," accessed: 2022-9-30. [Online]. Available: https://www.govinfo.gov/content/pkg/BILLS-117s4913is/pdf/BILLS-117s4913is.pdf