

High Performance High-Assurance Interval Arithmetic Based Safety Monitoring for Robot Arms on Industrial Multi-Core PLCs

José Cestero, John Rogan, Franz Franchetti
{jcestero, jrogan, franz}@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, USA

Tao Cui
tao.cui@siemens.com
Siemens U.S.
755 College Rd E, Princeton, NJ 08540, USA

Abstract—The certified runtime of an industrial Programmable Logic Controller (PLC) excels at IEC 61131-3 control logic but is computationally constrained for floating-point-intensive safety checks. We present a proof of concept that offloads safety computation to the Linux partition of a hypervisor-based PLC, using robot-arm boundary checking as the demonstration case. Using a high performance interval arithmetic library, we perform polytope-based boundary checks on the Linux partition, exchanging data with the PLC runtime partition through a shared-memory inter-partition framework. Our implementation achieves sub-millisecond latency while providing formally sound floating-point guarantees.

Index Terms—Robotic safety systems; interval arithmetic; real-time boundary checking; Programmable Logic Controllers (PLC)

I. INTRODUCTION

Programmable Logic Controllers (PLCs) are pivotal in industrial automation. Their certified IEC 61131-3 runtime excels at Boolean interlocks and integer operations but is computationally constrained for complex floating-point tasks such as geometric boundary checking. Modern hypervisor-based PLCs pair this certified runtime with a general-purpose Linux partition on the same hardware, offering greater compute power — but the Linux partition is not safety-certified. We use robot arm boundary monitoring as a proof-of-concept to bridge this gap: we offload floating-point boundary computation to the Linux partition and apply interval arithmetic to provide formally sound floating-point guarantees.

We make the following contributions:

- A high-assurance interval arithmetic implementation of bounding box safety checking for robot arm position monitoring.
- A novel inter-partition architecture that offloads floating-point safety computation to the Linux partition of a hypervisor-based PLC, exchanging real-time data with the certified PLC runtime partition via shared memory.

II. BACKGROUND

Originally designed to replace complex control circuits of relays and switches, PLCs are specialized computers that execute IEC 61131-3 programs — a family of languages specifically designed for industrial control. Modern industrial PLCs increasingly adopt a hypervisor-based architecture, hosting two isolated partitions on the same hardware: a PLC runtime partition that exclusively executes IEC 61131-3 con-

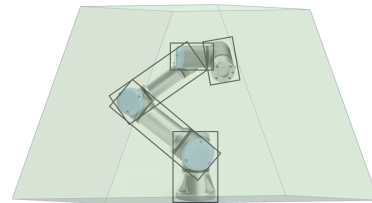


Fig. 1. Visualization of UR3e robot arm operating within a polytope-based safety boundary. The shaded volume defines the allowable workspace used by the safety monitor to detect violations. UR3e image adapted from [6].

trol logic and has direct access to I/O modules and fieldbus, and a generic OS partition that handles cloud connectivity, web services, and computationally demanding tasks [1]. The hypervisor enforces strict isolation between partitions, which is critical for PLC runtime safety.

To enable communication between these two partitions, most multi-core PLCs provide a shared-memory software environment for inter-core data exchange. In the evaluation platform used in this work, this environment is called the Real-time Information Backbone (RIB) [2]; the architecture generalizes to any multi-core PLC with a comparable inter-partition shared-memory interface.

The generic OS partition, which often runs Linux, has much more freedom in the types of applications and computational techniques that it can execute. Our goal is to make the OS core as safe and secure as the PLC runtime core by utilizing modern high-assurance procedures, allowing the PLC to complete more complex industrial tasks. Prior work, such as HA-SPIRAL, has demonstrated how interval arithmetic and formal guarantees can be applied to cyber military systems [3].

III. METHODOLOGY AND IMPLEMENTATION

Our safety system simulates a polytope bounding box around the machine that represents a volume of space where the arm can operate freely (Fig. 1). The simulation of this polytope is achieved by generating a set of planes, with each plane represented by a normal vector perpendicular to the plane and a displacement from the origin.

To ensure all parts of the robotic arm are within the newly generated bounding box, the arm's sensor data is converted to a set of points in space that surround the arm. Our implementation uses Universal Robots' UR3e model, which has six axes, using joint angle sensors as the primary positional data.

Whether each of these points lies within the bounding box is determined using interval arithmetic. The operational code is taken from an open source high performance library [4]. A single interval is calculated using the dot product between each point and the normal vector of each plane. If the upper bound of any interval is positive, a point is outside the plane and the arm is said to be outside the bounding box, and an emergency stop is transmitted.

The main reason for using interval arithmetic, along with the high-performance library that it requires, is to demonstrate how floating-point operations can be offloaded to the Linux partition of the PLC safely. The runtime core, with its ironclad safety requirements and high performance when it comes to generic control logic can be greatly beneficial, yet it comes with the drawback of struggling with floating-point computation latency. Offloading floating-point calculation opens the door to a variety of new applications for multi-core PLCs, including more efficient linear algebra for machine learning and the ability to accommodate more I/O devices with high precision. Interval arithmetic is an especially safety-conscious stand-in, prior work has demonstrated the importance of interval arithmetic as a solution to floating-point roundoff errors as the computational foundation of the HA-SPIRAL software system that we are emulating [3, 4].

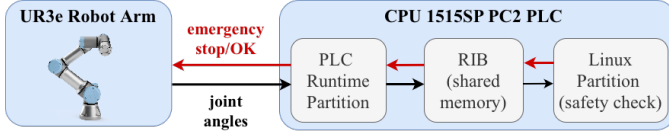


Fig. 2. **System architecture.** The PLC runtime partition receives joint angle data, sends it to the inter-partition shared memory (RIB), where it is consumed by the Linux partition for safety checking. UR3e image adapted from [6].

IV. EVALUATION

To test the implementation on hardware, we used a Siemens CPU 1515SP PC2 PLC, a representative hypervisor-based industrial PLC, running on an Intel Atom E3940 (1.60 GHz). We deployed two programs to the Linux partition: one that feeds simulated joint sensor data into the RIB shared memory (standing in for data that would arrive from the PLC runtime partition when connected to a physical robot arm), and another that reads from shared memory and performs the safety check (Fig. 2).

PROFINET, which is a standard field bus protocol for PLCs, prescribes a cycle time of less than a millisecond for motion control [5]. We measured the latency of just the safety check by taking an average of 360 checks on a continuously moving robotic arm, with the complexity of the boundary polytope as the independent variable. We also measured the latency of a single worst-case interval arithmetic operation to establish an upper bound on operation complexity via the RDTSCP hardware timestamp counter, taking the minimum over 30 trials to account for interruptions by the OS.

We determined that interval multiplication was the most computationally expensive (Table 1). At this rate, the Linux core can execute approximately 85,000 total interval multi-

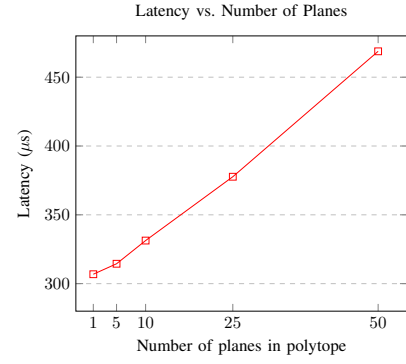


Fig. 3. The latency displayed is the average execution time of the safety check after 360 discrete robotic movements.

TABLE I
INTERVAL ARITHMETIC OPERATION LATENCY (INTEL ATOM E3940)

Category	Operation	Latency (ns/op)
Arithmetic	add	1.271
	sub	1.352
	mul	11.802
Comparison	max	1.354
	geq	2.940
Unary	abs	5.030

plications within the sub-millisecond budget, indicating that interval arithmetic itself is not a limiting factor for real-time performance. We also observed that increasing the complexity of the boundary increased latency linearly (Fig. 3); even the most complex boundary of 50 planes saw a latency increase of about 50% when compared with the single plane latency. This further suggests the safety check's latency is much more dependent on the performance of the joint-to-position conversion, and the interval arithmetic computation takes up a comparatively smaller fraction of the total latency.

V. CONCLUSION AND FUTURE WORK

In this work, we have demonstrated that complex floating-point safety boundary computation can be offloaded to the Linux partition of a hypervisor-based industrial PLC with sub-millisecond latency, using interval arithmetic to provide sound floating-point guarantees. A key limitation of the current architecture is that the Linux partition, as opposed to the PLC runtime partition, is not safety-certified (e.g., per IEC 62061); this work establishes computational feasibility, not a deployable safety function. Closing this gap is the primary direction for future work. One approach that we have taken is to create a more robust communication protocol between cores by using a watchdog that verifies a secure connection over the RIB shared memory interface. We have successfully demonstrated this watchdog using a PLC emulator, but we have yet to test on real hardware. Additional future work would include optimizing the performance of the safety check, lookup-table trigonometric functions for joint-to-position conversion, and using an algorithm optimized for fewer sign comparisons when rounding on the boundary calculation.

REFERENCES

- [1] M. Cinque, D. Cotroneo, L. De Simone, and S. Rosiello, "Virtualizing mixed-criticality systems: A survey on industrial trends and issues," *Future Generation Computer Systems*, vol. 129, pp. 315–330, Apr. 2022, doi: 10.1016/j.future.2021.12.002.
- [2] Siemens AG, "Realtime Information Backbone (RIB)," Siemens Industry Online Support, Document ID 109810853, [Online]. Available: <https://support.industry.siemens.com/cs/document/109810853>
- [3] T.-M. Low and F. Franchetti, "High Assurance Code Generation for Cyber-Physical Systems," presented at the Proc. IEEE International Symposium on High Assurance Systems Engineering (HASE), 2017.
- [4] J. Rivera, F. Franchetti, and M. Püschel, "An Interval Compiler for Sound Floating Point Computations," presented at the Proc. International Symposium on Code Generation and Optimization (CGO), 2021.
- [5] H. Harrington, "Real Time Classes - PROFINET University," PROFINET University, May 13, 2016. <https://profinetuniversity.com/profinet-basics/profinet-real-time-classes/> (accessed July 09, 2026).
- [6] Universal Robots A/S, "UR3e Collaborative Robot," Universal Robots. [Online]. Available: <https://www.universal-robots.com/products/ur3e/>