Of Good Demons and Bad Angels:
Guaranteeing Safe Control under Finite Precision

1st Samuel Teuber [ID]
Karlsruhe Institute of Technology
Karlsruhe, Germany
teuber@kit.edu

2nd Debasmita Lohar [ID]
Karlsruhe Institute of Technology
Karlsruhe, Germany
debasmita.lohar@kit.edu

3rd Bernhard Beckert [ID]
Karlsruhe Institute of Technology
Karlsruhe, Germany
beckert@kit.edu

Abstract—As neural networks (NNs) become increasingly prevalent in safety-critical neural network-controlled cyber-physical systems (NNCSs), formally guaranteeing their safety becomes crucial. For these systems, safety must be ensured throughout their entire operation, necessitating infinite-time horizon verification. To verify the infinite-time horizon safety of NNCSs, recent approaches leverage Differential Dynamic Logic ($d\mathcal{L}$). However, these $d\mathcal{L}$-based guarantees rely on idealized, real-valued NN semantics and fail to account for roundoff errors introduced by finite-precision implementations.

This paper bridges the gap between theoretical guarantees and real-world implementations by incorporating robustness under finite-precision perturbations—in sensing, actuation, and computation—into the safety verification. We model the problem as a *hybrid game* between a *good Demon*, responsible for control actions, and a *bad Angel*, introducing perturbations. This formulation enables formal proofs of robustness w.r.t. a given (bounded) perturbation. Leveraging this bound, we synthesize sound and efficient implementations, thus providing a complete end-to-end solution. We evaluate our approach on case studies from the automotive and aeronautics domains, producing efficient NN implementations with rigorous infinite-time horizon safety guarantees.

Index Terms—Differential Dynamic Logic, Mixed-Precision Fixed-Point Tuning, Neurally-Controlled Systems

I. INTRODUCTION

Neural networks (NNs) are increasingly integrated into critical systems, such as adaptive cruise control in cars [1]–[3] or collision avoidance in airplanes [4], [5]. Thus, ensuring the safety of NN Control Systems (NNCSs) becomes imperative.

Recent research has made significant progress toward this goal. Some approaches [6]–[19] only analyze the input-output behavior of the NN but ignore the essential feedback-loop dynamics between the NN and its physical environment. Others [20]–[31] consider this feedback loop but only verify safety for a *finite*-time horizon, which may be insufficient. A recent technique, VerSAILLE [32], addresses this limitation by proving safety throughout the entire operation (*infinite-time horizon*) of the NNCS using Differential Dynamic Logic ($d\mathcal{L}$).

However, VerSAILLE's strong guarantees rely on idealized, real-valued NN semantics and do not account for roundoff errors arising from practical finite-precision (floating- or fixed-point) implementations. As recently also demonstrated empirically [33], finite-precision arithmetic introduces errors at

potentially every operation, which can accumulate and lead to incorrect or unsafe decisions [34]. Furthermore, efficiency demands often require implementations in low-precision fixed-point arithmetic [35], [36], introducing larger numerical errors in the computation. Thus, to ensure real-world safety, it is crucial to bridge this implementation gap [37] by verifying that safety guarantees are valid under finite-precision semantics.

While verification of quantized NNs has been explored [38], it is known to be PSPACE-hard and typically focuses on input-output specifications rather than closed-loop dynamics, which must simultaneously account for the continuous, real-valued behavior of physical dynamics. Sound quantization techniques [39] automate precision tuning but leave the burden of safety verification to the user. A recent work [40] has combined safety analysis with bounded roundoff errors, but their guarantees are limited to finite-time horizons.

This leaves a critical gap: the need for an efficient and practical method to guarantee *infinite-time horizon safety* of NNCSs while accounting for finite-precision errors.

In this paper, we propose a general infinite-horizon safety verification methodology, building on VerSAILLE [32], that accounts for finite-precision errors in sensing, actuation, and computation. We observe that some "safe" $d\mathcal{L}$ control strategies (*envelopes*) can become unsafe even under miniscule input or output perturbations, making them unsuitable for verifying realistic (NN) controllers. To address this, we introduce a new notion of *robustness* for control envelopes, which is also of independent interest beyond handling roundoff errors. Finally, we integrate our approach with mixed-precision tuning to generate sound and efficient fixed-point implementations that can be directly compiled using standard high-level synthesis tools for FPGAs, providing a complete end-to-end solution. To summarize, we make the following contributions:

(C1) We formalize robustness for control envelopes under perturbation – a necessary condition for finite-precision $d\mathcal{L}$-based NN verification – using Differential Game Logic ($d\mathcal{GL}$), as a hybrid game between a *good* Demon (control actions) and a *bad* Angel (perturbations). We present a decidable, sufficient criterion for robustness.

(C2) We propose a novel, decidable criterion for guaranteeing the infinite-time horizon safety of NNs under given perturbations leveraging existing *real-valued* NN verifiers.

(C3) We integrate precision tuning (with Daisy [41]) to synthesize efficient mixed-precision fixed-point implementations that guarantee a provided error bound and are compatible with standard hardware synthesis tools [42].

We evaluated our approach on three realistic case studies: an Adaptive Cruise Control (ACC) system [1], [32] with discrete and continuous acceleration control, and a Vertical Airborne Collision Avoidance System (VCAS) [4]. Our analysis shows that the continuous ACC control envelope from [32] is not robust to output perturbations; we propose modifications and verify safety under bounded errors. For VCAS, our results imply (limited) robustness of the original control envelope. Using the computed perturbation bounds, we synthesized mixed-precision fixed-point implementations for all case studies, compiled them with Xilinx Vivado [42], and measured their latency (running time) in machine cycles. An extended version of this paper (including the Appendix) can be found on arXiv [43].

II. APPROACH AT A GLANCE

This section provides a high-level overview using a simplified ground robot from [45]. The technical background and the proposed approach are presented in the subsequent sections.

a) Ground Robot Example:
The robot, at distance $p \in \mathbb{R}$ to an obstacle, moves along a corridor (see Figure 2). It controls velocity v, which can be updated once per control cycle of duration of at most T. To ensure safety, the robot must maintain $p \geq 0$.
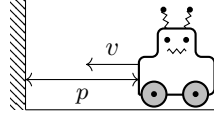
Fig. 2. A robot at distance p controls its velocity v and avoids an obstacle ($p \geq 0$).

Let us consider the case where the robot can only move toward the wall (i.e., $v \geq 0$). Since the robot has full control over its velocity, a safe strategy is to choose any $v \geq 0$ such that $p - Tv \geq 0$, ensuring it does not collide with the obstacle within one control cycle. The aim is to formally prove that the robot remains safe under this strategy.

b) Nomenclature: As we discuss control strategies at different levels, we unify the terminology as follows: First, nondeterministic control strategies modeled and verified in $d\mathcal{L}$ are referred to as *control envelopes*. Second, concrete real-valued NNs that deterministically provide a control action for a given input are called *idealized implementations*, and will be verified against control envelopes. Finally, hardware-level realizations of idealized implementations using fixed-point arithmetic are termed *fixed-point implementations*. We will bound their numerical errors w.r.t. their idealized counterparts.

c) Verifying Control under Finite Precision: A control envelope, where v is chosen arbitrarily such that $p - Tv \geq 0 \wedge v \geq 0$, can be proven safe in real-valued $d\mathcal{L}$.

In practice, real-valued arithmetic does not exist, and finite precision introduces pitfalls: When p is close to 0, even infinitesimal perturbations to v can cause $p < 0$, leading to a collision. Braking earlier *cannot* resolve this, as similar perturbations could, by induction, still push the robot into an unsafe region. This illustrates a *non-robust* control envelope,

highlighting the need to *rethink* the system structure—or at least the model—by, e.g., also admitting negative velocities.

d) Our Approach: Our work extends $d\mathcal{L}$-based safety guarantees to finite-precision NNs while: 1) still providing guarantees w.r.t. precise real-valued physical dynamics modeled in $d\mathcal{L}$ and 2) leveraging efficient, state-of-the-art real-valued NN verification tools. We build on VerSAILLE [32], which proves real arithmetic refinement relations between NNs and provably safe $d\mathcal{L}$ control envelopes. Just as VerSAILLE, and as indicated in Figure 1, our approach assumes the availability of a safe $d\mathcal{L}$ model of the analyzed control system. In particular, this $d\mathcal{L}$ model comprises a (nondeterministic) *control envelope* which encodes provably safe control actions. To consider perturbations arising from sensing, computation (e.g., fixed-point arithmetic), or actuation, we model such perturbations as real-valued hybrid programs, termed as *angelic perturbations*. For example, in our robot example, we define an angelic perturbation α_{noise} executed after the original control envelope chooses v: It nondeterministically selects a bounded perturbation $|\varepsilon_v| \leq \delta_v$ and then updates v via $v := v + \varepsilon_v$. This captures bounded perturbation of up to δ_v for the robot's actuation, whether due to numerical imprecision or actuator inaccuracies.

As can be seen in Figure 1, these angelic perturbations can be used to multiple ends: First, we can use them to analyze the robustness of a given $d\mathcal{L}$ model/control envelope. To this end, we provide a game-based robustness definition and decidable sufficient criteria for robustness. Second, for a robust control envelope, we extend VerSAILLE's analysis to take account of angelic perturbations in a decidable manner. This allows us to prove infinite-time horizon NNCS safety under perturbation. Finally, for NNs that are safe under perturbation, we can leverage sound quantization techniques, which synthesize a fixed-point implementation of the real-valued NN where the maximal error bound matches our angelic perturbation.

Towards our contribution (C1), we provide a game-theoretic definition for the robustness of control envelopes: There must *exist* control actions that keep the system safe irrespective of the environment's *angelic perturbations*. Classic $d\mathcal{L}$ safety properties ($[\alpha]\,\psi$) interpreted as game show that Demon, who has no influence on execution, has a winning strategy irrespective of Angel's control and environment choices. Robustness hands over the controller choices to a *good* Demon who must choose actions to weather a *bad* Angel adversarially choosing perturbations and environment behavior. Our formulation induces *decidable* real-arithmetic conditions applicable to *any* discrete, loop-free angelic perturbation. We identify *liveness* (a control envelope does not get *stuck*). as a special case of robustness. For our robot, a control envelope with $v \geq 0$ is not robust under α_{noise}, but an envelope with bidirectional movement is (choose any v such that $p - T(v + \delta) \geq 0$).

Concerning (C2), we extend VerSAILLE to account for angelic perturbations: NN verification reduces to checking whether the NN implements a *winning strategy* for the robustness game formulated in (C1) and robustness becomes a *necessary* condition for $d\mathcal{L}$-based NN verification under finite
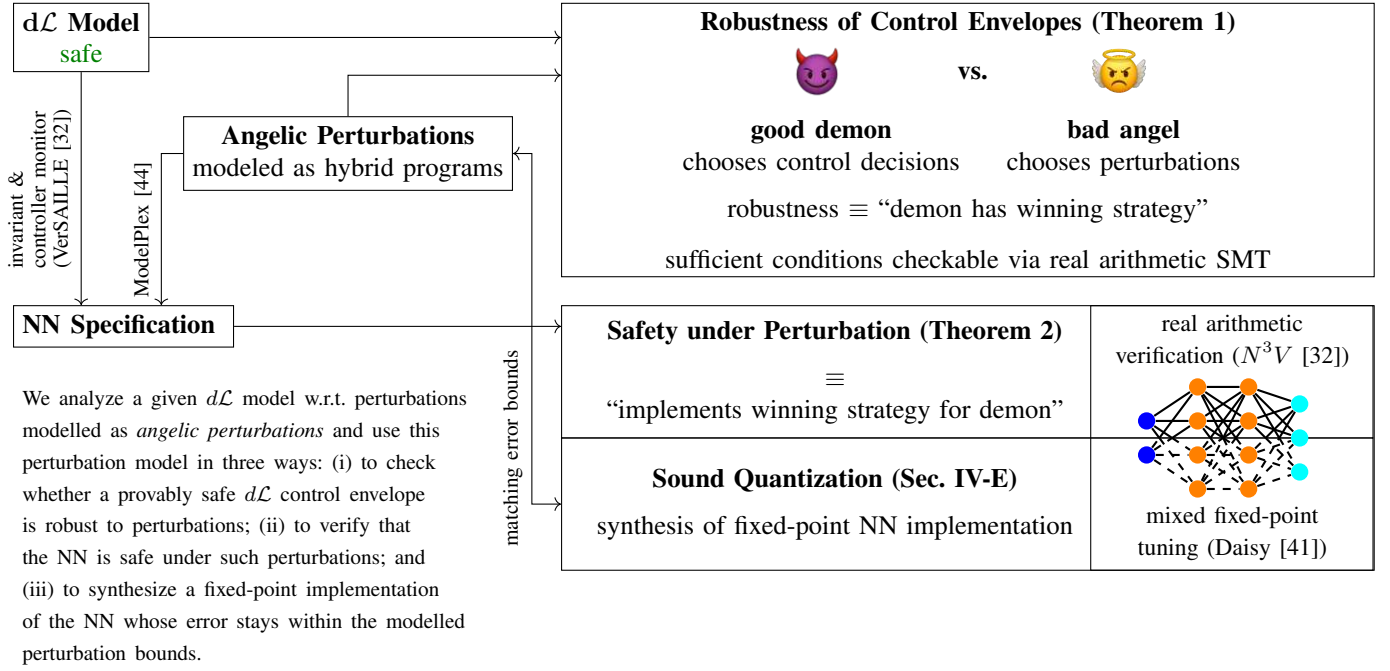
dL Model		Robustness of Control Envelopes (Theorem 1)

dL Model
safe

Robustness of Control Envelopes (Theorem 1)

😈 **vs.** 👿

good demon **bad angel**
chooses control decisions chooses perturbations

robustness ≡ "demon has winning strategy"

sufficient conditions checkable via real arithmetic SMT

Angelic Perturbations
modeled as hybrid programs

invariant &
controller monitor
(VerSAILLE [32])

ModelPlex [44]

NN Specification

We analyze a given dL model w.r.t. perturbations modelled as *angelic perturbations* and use this perturbation model in three ways: (i) to check whether a provably safe dL control envelope is robust to perturbations; (ii) to verify that the NN is safe under such perturbations; and (iii) to synthesize a fixed-point implementation of the NN whose error stays within the modelled perturbation bounds.

matching error bounds

Safety under Perturbation (Theorem 2)
≡
"implements winning strategy for demon"

real arithmetic
verification (N^3V [32])

Sound Quantization (Sec. IV-E)
synthesis of fixed-point NN implementation

mixed fixed-point
tuning (Daisy [41])

Fig. 1. Overview of approach which assumes a safe dL model as foundation (emojis generated by ChatGPT)

precision. For the robot with bidirectional movement, the real-valued condition $p \geq 0 \rightarrow p + Tf(p) \geq 0$, where $f(p)$ is the velocity computed by the NN f can be extended to $-\delta_v \leq \varepsilon_v \leq \delta_v \wedge p \geq 0 \rightarrow p+T(f(p)+\varepsilon_v) \geq 0$, which is also compatible with existing linear and nonlinear NN verification tools. Disproving NN safety amounts to finding a concrete input p and perturbation ε_v violating the specification. Proving the specification yields an *infinite-time horizon safety guarantee* for the NN control system under bounded perturbations. Our approach systematically handles a broad class of perturbations with decidable construction from dL control envelopes.

Addressing (C3), we leverage NN safety guarantees under bounded perturbations and utilize Daisy for mixed-precision fixed-point tuning, with added pre- and post-processing steps to support typical NN structures and operations, enabling direct hardware synthesis using Xilinx Vivado [42].

Thus, our *end-to-end* approach ensures infinite-horizon safety under bounded finite-precision errors and produces sound, efficient, hardware-deployable NN controllers. While in this paper, we focus on NNs due to current tool support, the approach is broadly applicable and can be straightforwardly extended to other types of controllers.

III. BACKGROUND

This section provides background on proving infinite-time horizon safety for NN Control Systems using dL (Sec. III-A) and on efficient code generation for finite-precision (Sec. III-B). We illustrate these concepts using the robot example from Figure 2.

A. Safety Verification

We review Differential Dynamic Logic [46] for abstractly reasoning about the hybrid (continuous and discrete) behavior of Cyber-Physical Systems, Differential Game Logic [47] for handling adversarial dynamics, and describe the dL-based NN verification technology VerSAILLE [32].

1) Differential Dynamic Logic: Differential Dynamic Logic (dL) [46], [48]–[50] is a (first-order) dynamic logic [51], [52], i.e., a modal logic over a state space $\mathcal{S}$ whose modalities are parametrized with programs. The syntax of *hybrid programs* analyzed in dL is given by the following grammar, where x is a variable, Q is a real arithmetic formula, e is either $*$ or a real arithmetic term, and $f(x)$ is a real arithmetic function over x (with α and β representing hybrid programs):

$$\alpha, \beta ::= \underbrace{x' = f(x) \,\&\, Q}_{\text{continuous}} \mid \underbrace{x := e \mid ?(Q) \mid \alpha; \beta \mid \alpha \cup \beta}_{\text{discrete and loop-free}} \mid \underbrace{(\alpha)^*}_{\text{loop}}$$

The semantics of hybrid programs are described as state transitions where a state ν is a map from variables x to (real) values $\nu(x)$. For example, for e a real arithmetic term, the semantics of variable assignment are defined as $[\![x := e]\!] = \{(\nu, \mu) \in \mathcal{S}^2 \mid \mu = \nu_x^e\}$ where ν_x^e is a state that evaluates x to e and all other variables y to $\nu(y)$.

The defining feature of hybrid programs is their ability to model *continuous* dynamics: $x' = f(x) \,\&\, Q$ denotes a program where x nondeterministically evolves along the given differential equation within an evolution constraint Q, allowing x to take *any* reachable value satisfying Q. The primitive $x := *$ represents a nondeterministic assignment. The remaining constructs are: predicate checks (no transition if the check fails), sequential composition, nondeterministic choice, and looping. Safety properties can be expressed as formulas $P \rightarrow [\alpha] Q$,

stating that, if the precondition P holds, then Q holds after all (nondeterministic) executions of the hybrid program α. For example, the formula $x = 2 \rightarrow [y := x \cup y := x + x] \, y > 0$ is valid because for any state with $x = 2$, all possible executions ensure that $y > 0$ (specifically $y \in \{2, 4\}$) after execution.

$d\mathcal{L}$ also provides a dual modality to express liveness: $P \rightarrow \langle\alpha\rangle Q$ guarantees that, starting from any state satisfying P, it is *possible* to reach *some* state where Q holds. Moreover, the equivalence $[\alpha] \, Q \equiv \neg\langle\alpha\rangle\neg Q$ applies.

We also use the subset of *concrete, discrete, and loop-free* programs, corresponding to the discrete and loop-free production rules from the hybrid program grammar above. Vectors of variables are denoted as $\bar{x}$ while x is an individual variable.

Safety guarantees in $d\mathcal{L}$ typically have the form:

$$\mathrm{safety}(\mathrm{pre}, \mathrm{ctl}, \mathrm{env}, \mathrm{post}) \equiv \mathrm{pre} \rightarrow \big[(\mathrm{ctl}; \mathrm{env})^*\big] \, \mathrm{post}, \quad (1)$$

where pre defines initial conditions and post the safety guarantee; ctl and env describe the (nondeterministic) *control envelope* and the environment, respectively.

2) Differential Game Logic: Differential Game Logic ($dG\mathcal{L}$) extends $d\mathcal{L}$ to reason about *hybrid games*, i.e., hybrid, noncooperative zero-sum games between two players, *Angel* and *Demon*. A game is expressed as a program α in a syntax extending the hybrid program grammar as summarized below, however, for a comprehensive introduction, see [47].

For a game α, $[\alpha] \, Q$ states that Demon has a winning strategy to achieve Q, while $\langle\alpha\rangle Q$ indicates that Angel has a winning strategy. In the $d\mathcal{L}$ primitives, we assume Angel controls *nondeterministic decisions* by default, with control explicitly transferred to Demon via additional primitive α^d.

In this interpretation, a $d\mathcal{L}$ safety property $P \rightarrow [\alpha] \, Q$ for a classic hybrid program (i.e., without duality) states that, when starting from P, Demon has a winning strategy to ensure Q regardless of Angel's nondeterministic decisions. For example, the game $\alpha \equiv (v := 1 \cup v := -1)^d \, ; x' = v$ states that Demon can choose $v \in \{-1, 1\}$ (observe the duality operator) and subsequently Angel controls for how long to evolve x along the differential equation. The formula $x \neq 0 \rightarrow [\alpha] \, x \neq 0$ is then valid: Demon can choose v such that, even as Angel controls the evolution of x, x never reaches 0. Specifically, Demon selects $v := 1$ iff $x > 0$, ensuring $x \neq 0$. Hence, Demon has a winning strategy for $x \neq 0$. Although $dG\mathcal{L}$ has a substantially different semantical foundation [47], it alligns with $d\mathcal{L}$ semantics for programs and formulas without duality [53, p. 520ff].

3) Verifying NN with VerSAILLE: As indicated in Sec. III-A1, $d\mathcal{L}$ is used to prove the safety of abstract control envelopes, typically formulated as in Formula (1). For real-world systems, however, we must relate these abstract safety proofs to concrete controller implementations outside $d\mathcal{L}$. This requires showing that the idealized implementation *refines* the safe control envelope [54], [55], i.e., that *all* behaviors of the idealized implementation are also possible in the $d\mathcal{L}$ model [54], [55]. To this end, ModelPlex [44] can derive correct-by-construction *controller monitor formulas*. Given a control envelope ctl, its monitor formula χ_{ctrl} is a first-order

formula over pre- and post-states, satisfied only if a state transition is modeled by ctl. Formally:

Definition 1 (Correct Controller Monitor [44])**.** *A controller monitor formula χ_{ctrl} is* correct *for a hybrid program controller ctl with bound variables $x_1, \ldots, x_n$ iff the following $d\mathcal{L}$ formula is valid:* $\chi_{\mathrm{ctrl}} \rightarrow \langle\mathrm{ctl}\rangle \bigwedge_{i=1}^{n} x_i = x_i^+$.

At least, χ_{ctrl} typically includes the variables $x_1, x_1^+, \ldots, x_n, x_n^+$ describing the bound variables of ctl before and after the execution of ctl.

Originally used for constructing monitors and shields [1], [45], ModelPlex was recently shown to support a priori safety verification through VerSAILLE [32]. Given a safe $d\mathcal{L}$ model (Formula (1)) and a loop invariant χ_{inv}, ModelPlex derives the controller monitor formula χ_{ctrl}. We model the NN's behavior as a before-after predicate $\mathrm{impl}(\bar{x}, \bar{x}^+)$. Verifying infinite-horizon safety of the NNCS then reduces to proving the following real arithmetic formula [32, Thm. 1]:

$$\chi_{\mathrm{inv}} \wedge \mathrm{impl}\big(\bar{x}, \bar{x}^+\big) \rightarrow \chi_{\mathrm{inv}} \quad (2)$$

The obtained guarantee has a rigorous foundation in $d\mathcal{L}$: Based on $\mathrm{impl}(x, x^+)$, we derive a hybrid program (called *nondeterministic mirror*, denoted α_{refl} (impl)) mirroring the behavior of the before-after predicate *within $d\mathcal{L}$*. Proving Formula (2) then implies safety when replacing ctl with α_{refl} (impl).

B. Efficient Code Generation

We review the background on fixed-point arithmetic and mixed-precision tuning here.

1) Fixed-Point Arithmetic: Fixed-point numbers represent values as integers with an (implicit) format $\langle s, Q, \pi \rangle$, where $s \in 0, 1$ indicates the presence of a sign bit, $Q \in \mathbb{N}$ is the total word length, and $\pi \in \mathbb{N}$ specifies the binary point position (counted from the least significant bit). Bit allocation defines an integer part, I, determining the range $[-2^I, 2^I]$, and a fractional part, π, controlling the *precision*: more fractional bits yield higher precision. Assuming truncation as the rounding mode (default for synthesis tools like Xilinx Vivado [42]), the maximum roundoff error is $2^{-\pi}$. Fixed-point operations use efficient integer arithmetic and bit-shifting [56], making them ideal for resource-constrained systems.

2) Mixed-Precision Tuning: Mixed-precision tuning assigns different bitwidths to variables and constants to optimize resource usage while satisfying error bounds. Several techniques [41], [57]–[59] have been proposed to automate this process. In this work, we focus on the tool Daisy [41] and briefly summarize its approach below.

Daisy [41] performs mixed-precision tuning by heuristically navigating the search space using *delta debugging* [60]. To soundly compute roundoff errors during the search, Daisy applies either interval or affine arithmetic, depending on the configuration. The process starts from a minimal bit length, uniformly increasing the precision to meet the error target, and then recursively partitioning variables to selectively lower bitwidths. Among multiple feasible assignments, Daisy selects the one minimizing an *area-based* cost function. The resulting

mixed-precision fixed-point code can be compiled by synthesis tools like the Xilinx HLS compiler. Daisy is most effective for straight-line numerical code without complex control structures such as loops, conditionals, or arrays.

IV. SAFETY UNDER PERTURBATION

As discussed in Sec. III-A, VerSAILLE's safety guarantees rely on idealized NN implementations in real arithmetic. This has the advantage of enabling powerful real-arithmetic solvers [61]–[64] and real-valued NN verification tools (as reported in recent surveys and competitions [65]–[67]). However, it raises the question of whether these safety results hold when the NN is implemented in fixed-point arithmetic. In practice, real-world systems deviate from their idealized real-valued models along three dimensions:

(I1) *Sensor Readouts:* Limited bandwidth or finite measurement accuracy affects input precision.
(I2) *Roundoff Errors:* Errors can arise during computations.
(I3) *Actuator Limitations:* Physical constraints may impact the precision of chosen control actions.

In principle, such factors can be mitigated by designing control envelopes for environments that explicitly model sensor and actuator noises, and computation errors. For instance, the hybrid program $\alpha_{\text{noise}} \equiv (\varepsilon_v := *; ? (|\varepsilon_v| \le \delta_v); v := v + \varepsilon_v)$ models bounded perturbations in computing v in the robot example from Sec. IV-A. However, many case studies, e.g., [32], [68]–[70], only assume ideal sensing, actuation, and computation for simplicity. This is unsurprising as hybrid system analysis is already complex (even undecidable [71]) even without accounting for finite-precision effects.

This work proposes a technique to check control envelope robustness to sensing, actuation, and roundoff errors, given a safe control envelope oblivious to such perturbations. We further show how these control envelopes can be used to verify fixed-point implementations.

We begin by recalling how VerSAILLE guarantees safety for an idealized implementation for the robot example (Sec. IV-A). Building on this, our contribution starts with the formalization of perturbations (Sec. IV-B), a robustness property for control envelopes and decidable conditions for proving it (Sec. IV-C). We then extend VerSAILLE to support verification under perturbation (Sec. IV-D). While our method is illustrated using the running example — with some formal notions deferred to the extended version — Sec. V shows its applicability across realistic case studies.

A. Setup: Idealized Implementation Safety with VerSAILLE

We recall how $d\mathcal{L}$ and VerSAILLE can derive safety guarantees for an idealized implementation of our robot example from Figure 2. The system state is given by position p and velocity v along with a clock variable t tracking elapsed time. We formalize the robot's physical behavior ($p' = v$), or *plant model*, as a hybrid program env ensuring control actions occur at least every T seconds (see details in extended version).

Different control scenarios and corresponding pre/postconditions and control envelopes are shown in

$$
\begin{aligned}
\text{ctl}_1^1 &\equiv (v := *; ? (0 \le v \le V_{\max} \land Tv \le p)) \\
\text{ctl}_1^2 &\equiv (v := *; ? (-V_{\max} \le v \le V_{\max} \land 0 \le p - Tv)) \\
\text{ctl}_2^2 &\equiv (v := *; ? (-V_{\max} \le v \le V_{\max} \land 0 \le p - Tv \le W)) \\
\text{pre}_1 &\equiv p, V_{\max} \ge 0 \ \land \ T > 0 \quad \text{post}_1 \equiv 0 \le p \\
\text{pre}_2 &\equiv \text{pre}_1 \ \land \ p \le W > 0 \quad \text{post}_2 \equiv 0 \le p \le W
\end{aligned}
$$

Fig. 3. Possible control envelopes, pre- and postconditions for robot (Sec. II)

Figure 3. In the single-wall case, the safety property post_1 is guaranteed by the envelopes ctl_1^1 and ctl_1^2, which differ in whether bidirectional motion is allowed. For the two-wall case, the robot must stay within bounds defined by post_2, requiring a different control envelope ctl_2^2.

With suitable loop invariants, we can prove safety for ctl_1^1 and ctl_1^2 w.r.t. pre_1 and post_1, as well as safety for ctl_2^2 w.r.t. pre_2 and post_2 in the theorem prover KeYmaera X [72] (proof in extended version). The envelope ctl_1^1 slightly deviates from the controller in [45], which checks for $TV \le p$ (instead of $Tv \le p$). This change makes the envelope more *permissive* by allowing intermediate velocities $0 < v < V_{\max}$ when choosing $v = V_{\max}$ would no longer be safe; however, ctl_1^1's behavior directly depends on the relation between the (changing) variables p, v and the computed Tv.

We can now use the $d\mathcal{L}$ safety results to verify an idealized NN implementations. For demonstration, we assume the idealized implementation is described by the predicate $\text{impl}_{\mathbb{R}}(p, v^+) \equiv \left(v^+ = \frac{-1}{0.01(p+10)} + M\right)$. Infinite-time horizon safety for one-wall, unidirectional case then be verified by instantiating Formula (2) and proving the following:

$$
\underbrace{p \ge 0}_{\chi_{\text{inv}}^{1,1}} \land \text{impl}_{\mathbb{R}}(p, v^+) \to \underbrace{(0 \le v^+ \le V_{\max} \land 0 \le p - Tv^+)}_{\chi_{\text{ctrl}}^{1,1}}
$$

where $\chi_{\text{inv}}^{1,1}$ is the loop invariant and $\chi_{\text{ctrl}}^{1,1}$ is the controller monitor derived from ctl_1^1 via ModelPlex. In practice, this validity is not proven directly by Formula (2), but via highly tuned NN verification tools that implicitly reason about impl.

For $M = V_{\max} = 10$, we prove validity of the formula using an SMT solver. Let $\alpha_{\text{refl}}(\text{impl}_{\mathbb{R}})$ be the non-deterministic mirror of $\text{impl}_{\mathbb{R}}$, i.e., the hybrid program capturing its behavior. This implies the safety predicate $\text{safety}(\text{pre}_1, \alpha_{\text{refl}}(\text{impl}_{\mathbb{R}}), \text{env}, \text{post}_1)$. Using the same approach for ctl_2^2, we also prove safety for $\text{impl}_{\mathbb{R}}(p, v^+)$ for $M = V_{\max} = 10, W = 100$. However, VerSAILLE's analysis assumes real arithmetic. Hence, it ignores roundoff errors introduced by realistic, fixed-point implementations of $\text{impl}_{\mathbb{R}}$.

B. Formalizing Perturbations

We formalize perturbations as concrete, discrete, loop-free hybrid programs. As seen earlier, α_{noise} captures *post-controller* perturbations to v, which may represent roundoff errors, actuation errors, or both (with δ_v as maximal roundoff, maximal deviation in actuation, or the combined bound). However, sensor perturbations require modifying the state *before* controller execution. To cover all perturbation profiles (I1-I3), we formalize general perturbation as a tuple $(\alpha_{\text{pre}}, \alpha_{\text{post}})$,

where α_{pre} applies before the controller and α_{post} after. We call such pairs *angelic perturbations*:

Definition 2 (Angelic Perturbation). *An* angelic perturbation *consists of two discrete, concrete, loop-free hybrid programs* $\alpha_{pre}, \alpha_{post}$. *We denote an angelic perturbation as a tuple (i.e.* $(\alpha_{pre}, \alpha_{post})$).

In the robot example, we consider two such perturbation profiles:

$$\text{angel}^1_{\text{pre}} \equiv ?(\top) \quad \text{angel}^1_{\text{post}} \equiv \alpha_{\text{noise}}$$
$$\text{angel}^2_{\text{pre}} \equiv p^- := p; \varepsilon_p := *; ?(|\varepsilon_p| \le \delta_p); p := p + \varepsilon_p$$
$$\text{angel}^2_{\text{post}} \equiv p := p^-; \varepsilon_v := *; ?(|\varepsilon_v| \le \delta_v); v := v + \varepsilon_v$$

Here, $(\text{angel}^1_{\text{pre}}, \text{angel}^1_{\text{post}})$ perturbs only v by at max δ_v; whereas, $(\text{angel}^2_{\text{pre}}, \text{angel}^2_{\text{post}})$ perturbs both the sensor input p without affecting the system's dynamics (the original value is restored after control execution) and the control output v.

C. Envelope Robustness

Before verifying implementations, we must check if the envelope itself is robust to angelic perturbations —a necessary condition for subsequent verification as we use the envelope to check if our implementations represent a winning strategy.

We formulate the robustness of control envelopes as a game: rather than requiring that *all* control actions remain safe under perturbation, we require that for any state, *there exists* a control action that 1) is permitted according to the envelope, and 2) guarantees safety even when perturbed by the given angelic perturbation. The *good* Demon, choosing an action from within the envelope, hence must take into account the perturbations of the *bad* Angel.

To check this property, we formulate a $dG\mathcal{L}$ game where Demon chooses from ctl and Angel controls perturbations and the environment. Robustness (in $dG\mathcal{L}$) is then defined as Demon having a winning strategy to guarantee post:

Definition 3 (Robustness). *A control envelope* ctl *with a safety guarantee* safety(pre, ctl, env, post) *is robust to angelic perturbations* $(\text{angel}_{pre}, \text{angel}_{post})$ *iff the following is valid:*

$$\text{pre} \to \left[\left(\text{angel}_{pre}; (\text{ctl})^d; \text{angel}_{post}; \text{env}\right)^*\right] \text{post} \quad (3)$$

Definition 3 assumes that ctl already has a real-valued safety guarantee (e.g. from [32], [68]–[70]). We *reuse* this knowledge (and the safety proof) to check the robustness of ctl utilizing insights on the decidability of differential refinements [55]. We propose a practical, *sufficient* criterion for controller robustness that can be checked in decidable real arithmetic (see extended version for proofs and additional simplifications).

Theorem 1 (Robustness of Control Envelopes). *Let* ctl *be as in Definition 3 and* χ_{inv} *be the inductive invariant used to prove* safety$(\dots)$. *For angelic perturbation* $\text{angel}_{pre}, \text{angel}_{post}$ *and monitor formulas* $\chi_{\text{ctrl}}, \chi_{\text{pre}}, \chi_{\text{post}}$ *for*

ctl, angel_{pre}, angel_{post}, *the validity of the following real arithmetic formula implies the validity of Equation* (3).

$$\forall \bar{x}_0 \, \forall x_1 \, \exists \bar{x}_2 \, \forall \bar{x}_3 \, \left(\chi_{\text{inv}}(\bar{x}_0) \wedge \chi_{\text{pre}}(\bar{x}_0, \bar{x}_1) \to \right.$$
$$\left. \left(\chi_{\text{ctrl}}(\bar{x}_1, \bar{x}_2) \wedge \left(\chi_{\text{post}}(\bar{x}_2, \bar{x}_3) \to \chi_{\text{ctrl}}(\bar{x}_0, \bar{x}_3)\right)\right)\right) \quad (4)$$

Liveness: When both $\text{angel}_{\text{pre}}$ and $\text{angel}_{\text{post}}$ are set to the skip program $?(\top)$, the robustness property reduces to a standard *liveness* property, i.e., whether the control envelope (without perturbation) contains a safe action in *all* reachable states. In this sense, robustness can be seen as liveness under input/output perturbations.

Robot Example: As noted in Sec. II, the robot restricted to moving toward the wall may crash if velocity computation is even slightly perturbed. Using the formalization from Sec. IV-A and our robustness notion, we can state that the ctl^1_1 is not robust w.r.t. the angelic perturbation $(\text{angel}^1_{\text{pre}}, \text{angel}^1_{\text{post}})$, as confirmed by a counterexample to the corresponding $dG\mathcal{L}$ formula in Definition 3 in KeYmaera X (see also extended version's appendix). In contrast, for $\delta_v \le V_{\text{max}}/2$, we prove via Theorem 1 that ctl^2_1 is robust w.r.t. angelic perturbation $(\text{angel}^1_{\text{pre}}, \text{angel}^1_{\text{post}})$. Similarly, ctl^2_2 is robust under the additional assumption that $\delta_v * T \le W/2$, ensuring sufficient distance between the *two* walls for safe operation under worst-case velocity perturbations.

D. Implementation Safety under Perturbation

We have shown how to check the robustness of control envelopes by proving that each state admits *one* safe action. However, this is insufficient for concrete implementations, whether idealized or fixed-point, where *every* chosen action must ensure safety.

In $d\mathcal{L}$ terms, given a before-after predicate for an implementation impl and its $d\mathcal{L}$ representation $\alpha_{\text{refl}}(\text{impl})$, safety under perturbation requires proving:

$$\text{pre} \to \left[\left(\text{angel}_{\text{pre}}; \alpha_{\text{refl}}(\text{impl}); \text{angel}_{\text{post}}; \text{env}\right)^*\right] \text{post} \quad (5)$$

By translating angelic perturbations into real arithmetic [44], [55], we derive a decidable, real arithmetic condition for safety under perturbation (see proof in extended version).

Theorem 2 (Safety under Perturbation). *Let* ctl *be like in Definition 3 and* χ_{inv} *be the inductive invariant used to prove* safety$(\dots)$. *For angelic perturbation* $\text{angel}_{pre}, \text{angel}_{post}$ *and monitor formulas* $\chi_{\text{ctrl}}, \chi_{\text{pre}}, \chi_{\text{post}}$ *for* ctl, angel_{pre}, angel_{post} *the validity of the formula below implies Formula* (5).

$$\forall \bar{x}_0 \, \forall x_1 \, \forall \bar{x}_2 \, \forall \bar{x}_3 \, \left(\chi_{\text{inv}}(\bar{x}_0) \wedge \chi_{\text{pre}}(\bar{x}_0, \bar{x}_1) \wedge \right.$$
$$\left. \text{impl}(\bar{x}_1, \bar{x}_2) \wedge \chi_{\text{post}}(\bar{x}_2, \bar{x}_3)\right) \to \chi_{\text{ctrl}}(\bar{x}_0, \bar{x}_3) \quad (6)$$

Phrased differently, proving Formula (6) shows that impl encodes a *winning strategy* for demon in the game of Equation (3) (with ctl substituted by $\alpha_{\text{refl}}(\text{impl})$). For NNs, verification is typically done using NN verification tools, not general real arithmetic solvers. In this setting, Formula (6) (with impl omitted) serves directly as the NN's specification and can be simplified further (see extended version).

Robot Example: We previously showed that ctl_1^1 and ctl_1^2 are robust under angelic perturbation $(\text{angel}_{\text{pre}}^1, \text{angel}_{\text{post}}^1)$. We now examine whether the idealized implementation $\text{impl}_{\mathbb{R}}(p, v^+)$ (Sec. IV-A) is safe under perturbation.

To this end, we instantiate Formula (6) for all combinations of $\left\{\text{ctl}_1^2, \text{ctl}_1^2\right\} \times \left\{(\text{angel}_{\text{pre}}^1, \text{angel}_{\text{post}}^1), (\text{angel}_{\text{pre}}^2, \text{angel}_{\text{post}}^2)\right\}$ w.r.t. $\text{impl}_{\mathbb{R}}$, yielding four real arithmetic formulas. Each formula's validity implies the safety of implementation $\text{impl}_{\mathbb{R}}$ w.r.t. its environment (one/two walls) and chosen perturbation.

For $T = 1.0, V_{\max} = 10, W = 100, M = 9.5$ and $\delta_p = \delta_v = 0.25$ all 4 formulas are proven using KeYmaera X. Hence, any fixed-point implementation differing from $\text{impl}_{\mathbb{R}}$ by at most 0.25 and with sensor perturbations of ≤ 0.25 remains safe within the fixed domain $p \in [0, 100]$. We now use the safety result in Sec. IV-E to generate safe, efficient fixed-point code.

Importantly, we previously disproved the robustness of ctl_1^1 w.r.t. $(\text{angel}_{\text{pre}}^1, \text{angel}_{\text{post}}^1)$. Hence, as robustness is a necessary condition, no matter the parameter choice, we cannot prove safety of $\text{impl}_{\mathbb{R}}$ w.r.t. ctl_1^1 under the given angelic pertrubations. Without checking robustness, a failed safety proof under perturbation leaves unclear if this is due to a limitation of the fixed-point implementation or of the chosen $d\mathcal{L}$ model.

E. Implementation Synthesis

With $\text{impl}_{\mathbb{R}}(p, v^+)$ proven safe under bounded perturbations, we use state-of-the-art mixed-precision tuning to synthesize a fixed-point implementation that stays within the chosen bound δ_v. For this, we use Daisy [41] (also supports uniform precision) that performs tuning using interval or affine arithmetic. It generates C++ code with mixed-precision fixed-point arithmetic (up to 64-bit width), which can be directly compiled by Xilinx [42] to measure runtime in machine cycles.

However, Daisy does not natively support vectors, matrices, or loops, common in NNs. It requires unrolled structures and loops and also generates fully unrolled code, which may overwhelm Xilinx [39]. To address this, we add a pre-processing step to automatically unroll standard NN structures for Daisy and a post-processing step to reintroduce the structures and loops after tuning. This enables Daisy to handle our neural networks effectively. Note that we also considered the tool Aster [39], which supports NN-specific code. But it is limited to 32-bit precision due to solver constraints and becomes infeasible for deeper networks due to imprecise range overapproximations, making it unsuitable for our purpose.

For our robot example, we first used extended Daisy to generate a 32-bit uniform fixed-point implementation of $\text{impl}_{\mathbb{R}}(p, v^+)$, yielding a worst-case roundoff error of 5.37×10^{-6} and a latency of 68 cycles. Since we proved safety under perturbations up to $\delta_v = 0.25$, Daisy can synthesize a more efficient mixed-precision implementation using at most 17 bits, reducing latency to 30 cycles while still ensuring safety.

F. Limitations

As stated in Theorem 1, Formula (4) implies controller robustness. However, if Formula (4) is not valid, this does not necessarily mean the controller is non-robust. A counterexample $\bar{x}_0$ satisfying pre is a concrete violation, but one outside of pre may be spurious; robustness might still hold under a *stronger* loop invariant that excludes $\bar{x}_0$.

Thus, our control envelope approach is, in general, incomplete. However, many $d\mathcal{L}$ proofs provide guarantees for the weakest assumptions and thus set the invariant region as the initial condition proving safety results for very permissive control envelopes: If pre $\leftrightarrow \chi_{\text{inv}}$ (and not only pre $\rightarrow \chi_{\text{inv}}$), then any counterexample is a concrete violation w.r.t. actions deemed safe by ctl.

Finally, our method relies on proven, safe $d\mathcal{L}$ control envelopes. Fortunately, a wide range of envelopes exists across applications [68]–[70], [73]–[76], and recent work [77] shows that their synthesis can be automated.

V. EVALUATION

We evaluate our methodology on three realistic case studies to demonstrate its practical applicability. We begin with Adaptive Cruise Control (ACC), verifying safety for both discrete and continuous action spaces. We then apply our approach to the more complex and safety-critical Vertical Airborne Collision Avoidance System (VCAS), specifically focusing on two neural networks-Do-Not-Climb (DNC) and Do-Not-Descend (DND)—which issue collision avoidance advisories in case the current advisory is DNC/DND.

While the NNs for continuous ACC and VCAS are taken from prior work [4], [5], [32], the discrete ACC networks were trained specifically for this paper. The continuous ACC network has 256 ReLU nodes and over 12k parameters; the discrete variant has the same architecture but has 3 output nodes instead of 1. Each of the two networks used in the VCAS case study contains 6x45 ReLU nodes. Table I summarizes our experimental results.

A. Experimental Setup

All experiments were conducted on an Ubuntu 22.04 system with an AMD Ryzen 7 CPU and 32 GB RAM. Safety proofs in differential dynamic logic and real arithmetic were performed interactively using KeYmaeraX [72], which internally relies on Mathematica [64] for quantifier elimination. NN verification was done using N^3V [32], [78], which supports polynomial specifications with arbitrary propositional structure. Internally, N^3V is based on Z3 [61], [62], PicoSAT [79] and the linear specification NN verifier nnenum [16]. For mixed-precision code generation and FPGA synthesis, we used Daisy's March 2, 2021 version (with no major updates since then) and Xilinx's Vitis HLS [42] (version 2023.1), downloaded on September 24, 2024.

B. Case Study 1: Continuous Adaptive Cruise Control

For continuous ACC, prior work [32] provides a $d\mathcal{L}$ safety guarantee for NN verification. In this setting, an *ego car* (following a *front car*) chooses a relative acceleration a_{rel} based on current relative position p_{rel} and velocity v_{rel}. The safety goal is to prevent collisions while ensuring the ego car

Case Study	δ	NN Verification		NN Synthesis	
		Time ($\mathbb{R}$)	Time (δ)	Time	# Cycles
ACC (cont.)	1.0	2.02m	9.3m	1.53h	567
ACC (disc.)	0.01	59s	1m	2.01h	579
VCAS DNC	25^{-3}	18.92m	27.45m	1.73h	655
DND	25^{-3}	15.75m	19.1m	1.82h	656

does not fall behind. Previous work had also established a real-valued safety guarantee for a validated NN [32, Sec. 5], raising the question whether the same safety guarantee holds in a finite precision setting. To this end, we formalize an angelic perturbation as $\big(? (\top), \text{angel}_{\text{post}}^{\text{acc}}\big)$ with:

$$\text{angel}_{\text{post}}^{\text{acc}} \equiv \big(\varepsilon_{a_{\text{rel}}} := *; ? \left(|\varepsilon_{a_{\text{rel}}}| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\big)$$

Envelope Robustness: Initial analysis showed that the control envelope derived before is not robust under $\big(? (\top), \text{angel}_{\text{post}}^{\text{acc}}\big)$. It assumes actions in the range $a_{\text{rel}} \in [-B, A]$, where $-B$ is the maximum braking acceleration, and guarantees safety as long as braking with $-B$ avoids a crash. However, this relies on the *precise* execution of $-B$, which is unrealistic under finite precision. We therefore revised the control envelope to reflect this limitation and formally proved its safety and robustness in KeYmaera X.

NN Verification: Using the revised control envelope, we derived the specification for the NN as described in Sec. IV-E and verified the NN using N^3V. Table I reports the verification times for both the original (Time $\mathbb{R}$) and perturbation-aware (Time δ) specifications. While verification under perturbation takes longer, it remains feasible. The longer time is likely due to the increased propositional complexity of the specification, leading to more queries to the underlying verifier nnenum.

Code Generation: Finally, we used Daisy [41] to generate an efficient mixed-precision fixed-point C++ implementation of the NN that meets the error bound $\delta = 1.0$. To compute latency, we compiled the generated code with Xilinx Vivado HLS [42] for a standard FPGA architecture. Note that this latency is exact, i.e., we do not consider noisy runtime measurements from actual hardware.

With our pre- and post-processing pipeline, Daisy successfully generated code with an error of 1.0. While uniform precision required 32 bits, mixed-precision tuning produced an implementation using a mix of 31- and 32-bit variables, reducing total bit usage by 3.66% compared to the uniform version. The latency remained at 567 cycles, as the limited reduction in bit widths was not sufficient to impact the overall execution time. We note that retaining the fully unrolled version could potentially reduce latency, but the size may overwhelm Xilinx.

Discussion: The continuous ACC case study illustrates that the proposed approach presented in this paper applies effectively to realistic regression-based NN control systems, handling roundoff errors.

While NN verification typically scales with the number of ReLU nodes, mixed precision tuning and code generation scale with the number of parameters of the NN, *including* all weights and biases. Since both steps are performed only once, our method remains practical even for networks with 256 ReLU nodes and over 12K parameters.

Most importantly, the proposed approach enhances *warranted* trust in NN controllers by ensuring *infinite-time horizon* safety and bridging the gap between theoretical guarantees in real arithmetic and their efficient, deployable code on FPGAs.

C. Case Study 2: Discrete Adaptive Cruise Control

In this case study, the NNs are classifiers selecting one of three actions ($a_{\text{rel}} \in -B, 0, A$) via an argmax over three outputs. Finite-precision analysis is more challenging here, as *any* error can flip decisions for *some* inputs near classification boundaries. For instance, if $a_{\text{rel}} = -B$ is selected for inputs x such that $\langle w, y \rangle \leq b$, then for input values with $\langle w, y \rangle = b$, even infinitesimal perturbations can flip the action.

Hence, in this case study, the key question is not *if* the NN's behavior changes under perturbations but *where* it changes. For the chosen control envelope and idealized implementation we verified safety under perturbation for a trained NN for $\delta = 0.01$ which simultaneously shows the envelope's robustness.

With a perturbation bound of 0.01, Daisy estimated that 32-bit uniform precision was sufficient to meet the error requirements. We then used Daisy to synthesize a mixed-precision implementation, assigning 29 to 32 bits across variables. The mixed-precision tuning took 2.01 hours and produced code with a latency of 579 cycles, achieving a 0. 69% reduction in latency and a 2.14% improvement in total bit usage compared to the 32-bit uniform precision implementation.

D. Case Study 3: Vertical Airborne Collision Avoidance

Our third case study focuses on the Vertical Airborne Collision Avoidance System (VCAS), developed through multiple Federal Aviation Administration (FAA)-supervised research efforts [80], [81]. It aims to prevent Near Mid-Air Collisions (NMACs), defined as aircraft coming within 500 feet horizontally or 100 feet vertically, by identifying dangerous trajectories and issuing timely advisories to adjust flight paths.

Given that hazardous situations often develop rapidly, the advisory system must provide *correct* decisions. Previous work has formalized this CPS in $d\mathcal{L}$ [68] and proposed NN controllers for use [4], [5]. Although the full combination of these NNs is unsafe, two specific NNs (DNC and DND) were proven safe under idealized real-valued semantics [32]. Like in the discrete ACC, these controllers work in a discrete action space with similar challenges under finite precision.

VCAS represents a critical application where providing formal safety guarantees for fixed-point implementations—not just idealized models—is crucial. To this end, we applied our methodology to the two verified NNs, constructing an angelic perturbation model analogous to $\text{angel}_{\text{post}}^{\text{acc}}$ to reflect realistic numerical and actuation uncertainties. We then derived the corresponding NN safety specifications.

Using N^3V, we verified both NNs under this new specification in comparable time to the original setting. Daisy's analysis showed that 42-bit uniform precision is required to meet the perturbation bound of 25^{-3}. We then synthesized mixed-precision implementations in 1.73 hours (DNC) and 1.82 hours (DND). While the latencies remained the same at 655 and 656 cycles w.r.t. the uniform baselines, mixed-precision versions have bit savings of 6.18% and 5.2%, respectively.

E. Discussion

Our three case studies demonstrate that our approach scales to safety-critical, real-world systems like VCAS and applies to both regression and classification-based NN Control Systems. Importantly, our methodology provides *global* safety guarantees for the NN, which are generally more challenging to obtain and scale less than local robustness guarantees. However, many control applications of interest rely on NNs of comparable size [3], making our approach practically relevant.

VI. RELATED WORK

Safety Analysis: While many techniques guarantee safety for Cyber-Physical Systems (CPS) control, most of them omit finite-precision or sensor/actuation errors. In principle, $d\mathcal{L}$ [46], [48], [50] supports reasoning about such errors; however, case studies often avoid it due to complexity [68]–[70]. Our approach addresses this gap by enabling robustness analysis of control envelopes from such studies.

VerSAILLE [32] is the only method using $d\mathcal{L}$ guarantees for NNCSs. This enables *infinite-time* horizon guarantees. In contrast, closed-loop reachability [20]–[25], [27]–[31], [82] ensures safety only over a finite horizon, limiting inductive reasoning [32, Appendix E.2]. Similarly, barrier certificates [83]–[87] lack the exact reasoning supported by VerSAILLE, and typically cover smaller regions. Unlike our work, all mentioned NNCS verification works *ignore* the effects of fixed-point arithmetic which are essential for *realistic* safety guarantees. Unlike [40], which integrates closed-loop reachability with bounding fixed-point errors, we extend VerSAILLE and can thus provide stronger, infinite-time horizon guarantees.

Recent work [88] analyzes divergence between closed-loop NN controllers using ε-equivalence [89], [90] and Lipschitz analysis, but focuses on finite-horizon settings and does not address sound quantization. For a broader comparison between closed-loop verification, barrier certificates, and VerSAILLE, we refer to [32, Sec. 5, 6, Appendix E.2].

Efforts to verify quantized NNs [38], [91], [92] use bit-vector constraints. These tools are incompatible with the real-valued polynomial constraints required by VerSAILLE, limiting their use for infinite-horizon NNCS verification.

Mixed Precision Tuning: For sound mixed-precision tuning, Daisy [41] (used in this work) and POPiX [59] support fixed-point arithmetic, while FPTuner [57], Salsa [58], and POP [93] focus on floating-point programs. Unlike Daisy, POPiX frames tuning as an ILP problem but relies on dynamic range analysis, which compromises soundness. Daisy also integrates expression rewriting to improve mixed-precision

tuning, though this added step increases computational cost, making it harder to scale for larger benchmarks.

Neural Network Quantization: State-of-the-art NN quantization [94]–[97] typically target NN classifiers, optimizing accuracy without formal guarantees. Recent efforts, such as Aster [39], automate mixed-precision tuning for NN controllers under error bounds but suffer from imprecise integer range estimation, limiting their usability for *deeper* networks. Popinns [98] also synthesizes bounded fixed-point implementations but relies on dynamic analysis and assumes access to a floating-point model, unlike our approach, where the goal is sound efficient code generation.

Combined Approaches: VeriPhy [45] synthesizes sandboxes based on $d\mathcal{L}$ contracts and interval-based fallback controllers. However, it provides guarantees around, not *of*, the controller itself. Extending VeriPhy to realistic NNs would require substantial changes, as NNs are not amenable to interactive proofs. We view VeriPhy as orthogonal to our approach, applicable when direct NN verification is impossible.

Finally, for code generation, we use Xilinx Vivado [42] for traditional and NN controllers. For neural controllers, tools like FloPoCo [99] can generate optimized FPGA-based dot products, which we leave as future work.

VII. CONCLUSION

This paper addresses the critical gap between theoretical infinite-horizon safety guarantees for NN-controlled systems and their practical finite-precision implementations. Since real-valued guarantees may fail under finite precision, we formulated control envelope robustness as a game between a *good* Demon and a *bad* Angel. Verifying infinite-horizon safety thus reduces to showing that the NN implements a winning strategy while keeping fixed-point implementation errors within allowable perturbations. Our approach relies on decidable real-arithmetic conditions, avoiding the need for undecidable $dG\mathcal{L}$ or $d\mathcal{L}$ validity proofs.

Our experiments on realistic NN control systems, from automotive to aeronautics, covering both regression- and classification-based NN architectures, demonstrate that accounting for perturbations introduces no significant overhead compared to perturbation-free verification, though costs may increase for more complex systems. While this paper focuses on NN controllers, the proposed technique naturally extends to other controller types. Overall, our work represents a significant step toward fostering *warranted* trust in real-world NN control systems.

REFERENCES

[1] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018* (S. A. McIlraith and K. Q. Weinberger, eds.), pp. 6485–6492, AAAI Press, 2018.

[2] M. Brosowsky, F. Keck, J. Ketterer, S. Isele, D. Slieter, and M. Zöllner, "Safe deep reinforcement learning for adaptive cruise control by imposing state-specific safe sets," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 488–495, 2021.

[3] T. T. Johnson, D. M. Lopez, L. Benet, M. Forets, S. Guadalupe, C. Schilling, R. Ivanov, T. J. Carpenter, J. Weimer, and I. Lee, "ARCH-COMP21 category report: Artificial intelligence and neural network control systems (AINNCS) for continuous and hybrid systems plants," in *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21), Brussels, Belgium, July 9, 2021* (G. Frehse and M. Althoff, eds.), vol. 80 of *EPiC Series in Computing*, pp. 90–119, EasyChair, 2021.

[4] K. D. Julian and M. J. Kochenderfer, "Guaranteeing safety for neural network-based aircraft collision avoidance systems," *CoRR*, vol. abs/1912.07084, 2019.

[5] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–10, 2016.

[6] H. Zhang, T. Weng, P. Chen, C. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 4944–4953, 2018.

[7] K. Xu, Z. Shi, H. Zhang, Y. Wang, K. Chang, M. Huang, B. Kailkhura, X. Lin, and C. Hsieh, "Automatic perturbation analysis for scalable certified robustness and beyond," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[8] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C. Hsieh, "Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[9] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C. Hsieh, and J. Z. Kolter, "Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual* (M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, eds.), pp. 29909–29921, 2021.

[10] P. Henriksen and A. R. Lomuscio, "Efficient neural network verification via adaptive refinement and adversarial search," in *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020* (G. D. Giacomo, A. Catalá, B. Dilkina, M. Milano, S. Barro, A. Bugarín, and J. Lang, eds.), vol. 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 2513–2520, IOS Press, 2020.

[11] P. Henriksen and A. Lomuscio, "DEEPSPLIT: an efficient splitting method for neural network verification via indirect effect analysis," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021* (Z. Zhou, ed.), pp. 2549–2555, International Joint Conferences on Artificial Intelligence Organization, 2021.

[12] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I* (R. Majumdar and V. Kuncak, eds.), vol. 10426 of *LNCS*, (Cham), pp. 97–117, Springer, 2017.

[13] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljic, D. L. Dill, M. J. Kochenderfer, and C. W. Barrett, "The Marabou framework for verification and analysis of deep neural networks," in *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I* (I. Dillig and S. Tasiran, eds.), vol. 11561 of *LNCS*, (Cham), pp. 443–452, Springer, 2019.

[14] R. Bunel, J. Lu, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar, "Branch and bound for piecewise linear neural network verification," *J. Mach. Learn. Res.*, vol. 21, pp. 42:1–42:39, 2020.

[15] S. Bak, H. Tran, K. Hobbs, and T. T. Johnson, "Improved geometric path enumeration for verifying ReLU neural networks," in *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I* (S. K. Lahiri and C. Wang, eds.), vol. 12224 of *LNCS*, (Cham), pp. 66–96, Springer, 2020.

[16] S. Bak, "nnenum: Verification of ReLU neural networks with optimized abstraction refinement," in *NASA Formal Methods - 13th International Symposium, NFM 2021, Virtual Event, May 24-28, 2021, Proceedings* (A. Dutle, M. M. Moscato, L. Titolo, C. A. Muñoz, and I. Perez, eds.), vol. 12673 of *LNCS*, (Cham), pp. 19–36, Springer, 2021.

[17] H. Zhang, S. Wang, K. Xu, L. Li, B. Li, S. Jana, C. Hsieh, and J. Z. Kolter, "General cutting planes for bound-propagation-based neural network verification," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), 2022.

[18] Z. Shi, Q. Jin, Z. Kolter, S. Jana, C. Hsieh, and H. Zhang, "Neural network verification with branch-and-bound for general nonlinearities," *CoRR*, vol. abs/2405.21063, 2024.

[19] H. Wu, O. Isac, A. Zeljic, T. Tagomori, M. L. Daggitt, W. Kokke, I. Refaeli, G. Amir, K. Julian, S. Bassan, P. Huang, O. Lahav, M. Wu, M. Zhang, E. Komendantskaya, G. Katz, and C. W. Barrett, "Marabou 2.0: A versatile formal analyzer of neural networks," in *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part II* (A. Gurfinkel and V. Ganesh, eds.), vol. 14682 of *LNCS*, (Cham), pp. 249–264, Springer, 2024.

[20] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling, "JuliaReach: a toolbox for set-based reachability," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019* (N. Ozay and P. Prabhakar, eds.), (New York, NY, USA), pp. 39–44, ACM, 2019.

[21] C. Schilling, M. Forets, and S. Guadalupe, "Verification of neural-network control systems by integrating taylor models and zonotopes," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 8169–8177, AAAI Press, 2022.

[22] H. Tran, D. M. Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Star-based reachability analysis of deep neural networks," in *Formal Methods - The Next 30 Years - Third World Congress, FM 2019, Porto, Portugal, October 7-11, 2019, Proceedings* (M. H. ter Beek, A. McIver, and J. N. Oliveira, eds.), vol. 11800 of *LNCS*, (Cham), pp. 670–686, Springer, 2019.

[23] H. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part I* (S. K. Lahiri and C. Wang, eds.), vol. 12224 of *LNCS*, (Cham), pp. 3–17, Springer, 2020.

[24] W. Xiang, H. Tran, X. Yang, and T. T. Johnson, "Reachable set estimation for neural network control systems: A simulation-guided approach," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 5, pp. 1821–1830, 2021.

[25] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verifying the safety of autonomous systems with neural network controllers," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 1, pp. 7:1–7:26, 2021.

[26] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig 2.0: Verification of neural network controllers using Taylor model preconditioning," in *Computer Aided Verification - 33rd International Conference, CAV 2021, Virtual Event, July 20-23, 2021, Proceedings, Part I* (A. Silva and K. R. M. Leino, eds.), vol. 12759 of *LNCS*, (Cham), pp. 249–262, Springer, 2021.

[27] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, "ReachNN: Reachability analysis of neural-network controlled systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, pp. 106:1–106:22, 2019.

[28] J. Fan, C. Huang, X. Chen, W. Li, and Q. Zhu, "ReachNN*: A tool for reachability analysis of neural-network controlled systems," in

Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings (D. V. Hung and O. Sokolsky, eds.), vol. 12302 of *LNCS*, (Cham), pp. 537–542, Springer, 2020.

[29] S. Dutta, X. Chen, and S. Sankaranarayanan, "Reachability analysis for neural feedback systems using regressive polynomial rule inference," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019* (N. Ozay and P. Prabhakar, eds.), (New York, NY, USA), pp. 157–168, ACM, 2019.

[30] M. E. Akintunde, E. Botoeva, P. Kouvaros, and A. Lomuscio, "Formal verification of neural agents in non-deterministic environments," *Auton. Agents Multi Agent Syst.*, vol. 36, no. 1, p. 6, 2022.

[31] C. Sidrane, A. Maleki, A. Irfan, and M. J. Kochenderfer, "OVERT: an algorithm for safety verification of neural network control policies for nonlinear systems," *Journal of Machine Learning Research*, vol. 23, no. 117, pp. 1–45, 2022.

[32] S. Teuber, S. Mitsch, and A. Platzer, "Provably safe neural network controllers via differential dynamic logic," in *Advances in Neural Information Processing Systems* (A. Globerson, L. Mackey, A. Fan, C. Zhang, D. Belgrave, J. Tomczak, and U. Paquet, eds.), Curran Associates, Inc., 2024.

[33] A. Szász, B. Bánhelyi, and M. Jelasity, "No soundness in the real world: On the challenges of the verification of deployed neural networks," in *Forty-second International Conference on Machine Learning*, 2025.

[34] D. Lohar, E. Darulova, S. Putot, and E. Goubault, "Discrete choice in the presence of numerical uncertainties," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2381–2392, 2018.

[35] S. Gopinath, N. Ghanathe, V. Seshadri, and R. Sharma, "Compiling kb-sized machine learning models to tiny iot devices," in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019* (K. S. McKinley and K. Fisher, eds.), (New York, NY, USA), pp. 79–95, ACM, 2019.

[36] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 1737–1746, JMLR.org, 2015.

[37] L. C. Cordeiro, M. L. Daggitt, J. Girard-Satabin, O. Isac, T. T. Johnson, G. Katz, E. Komendantskaya, A. Lemesle, E. Manino, A. Sinkarovs, and H. Wu, "Neural network verification is a programming language challenge," in *Programming Languages and Systems - 34th European Symposium on Programming, ESOP 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3-8, 2025, Proceedings, Part I* (V. Vafeiadis, ed.), vol. 15694 of *LNCS*, pp. 206–235, Springer, 2025.

[38] T. A. Henzinger, M. Lechner, and D. Zikelic, "Scalable verification of quantized neural networks," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 3787–3795, AAAI Press, 2021.

[39] D. Lohar, C. Jeangoudoux, A. Volkova, and E. Darulova, "Sound mixed fixed-point quantization of neural networks," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 5s, pp. 136:1–136:26, 2023.

[40] H. Thevendhriya, S. Ghosh, and D. Lohar, "Towards precision-aware safe neural-controlled cyber-physical systems," *IEEE Embedded Systems Letters (ESL)*, 2024.

[41] E. Darulova, E. Horn, and S. Sharma, "Sound mixed-precision optimization with rewriting," in *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2018, Porto, Portugal, April 11-13, 2018* (C. Gill, B. Sinopoli, X. Liu, and P. Tabuada, eds.), pp. 208–219, IEEE Computer Society / ACM, 2018.

[42] Vivado Lab Solutions, "Vivado design suite," 2021.

[43] S. Teuber, D. Lohar, and B. Beckert, "Of good demons and bad angels: Guaranteeing safe control under finite precision," *CoRR*, vol. abs/2507.22760, 2025.

[44] S. Mitsch and A. Platzer, "Modelplex: verified runtime validation of verified cyber-physical system models," *Formal Methods Syst. Des.*, vol. 49, no. 1-2, pp. 33–74, 2016.

[45] R. Bohrer, Y. K. Tan, S. Mitsch, M. O. Myreen, and A. Platzer, "Veriphy: verified controller executables from verified cyber-physical system models," in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, Philadelphia, PA, USA, June 18-22, 2018* (J. S. Foster and D. Grossman, eds.), (New York, NY, USA), pp. 617–630, ACM, 2018.

[46] A. Platzer, "Differential dynamic logic for hybrid systems," *J. Autom. Reason.*, vol. 41, no. 2, pp. 143–189, 2008.

[47] A. Platzer, "Differential game logic," *ACM Trans. Comput. Log.*, vol. 17, no. 1, p. 1, 2015.

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

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

[50] A. Platzer, "The complete proof theory of hybrid systems," in *Proceedings of the 2012 27th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2012*, pp. 541–550, {IEEE} Computer Society, 2012.

[51] D. Harel, *First-Order Dynamic Logic*, vol. 68 of *LNCS*. Heidelberg: Springer, 1979.

[52] D. Harel, D. Kozen, and J. Tiuryn, *Dynamic Logic*. MIT Press, 2000.

[53] A. Platzer, *Logical Foundations of Cyber-Physical Systems*. Cham: Springer, 2018.

[54] S. M. Loos and A. Platzer, "Differential refinement logic," in *LICS* (M. Grohe, E. Koskinen, and N. Shankar, eds.), (New York, NY, USA), pp. 505–514, ACM, 2016.

[55] E. Prebet and A. Platzer, "Uniform substitution for differential refinement logic," in *IJCAR* (C. Benzmüller, M. J. Heule, and R. A. Schmidt, eds.), vol. 14740 of *LNCS*, (Cham), pp. 196–215, Springer, 2024.

[56] A. A. Martinez, R. Majumdar, I. Saha, and P. Tabuada, "Automatic verification of control system implementations," in *Proceedings of the 10th International conference on Embedded software, EMSOFT 2010, Scottsdale, Arizona, USA, October 24-29, 2010* (L. P. Carloni and S. Tripakis, eds.), (New York, NY, USA), pp. 9–18, ACM, 2010.

[57] W. Chiang, M. Baranowski, I. Briggs, A. Solovyev, G. Gopalakrishnan, and Z. Rakamaric, "Rigorous floating-point mixed-precision tuning," in *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017* (G. Castagna and A. D. Gordon, eds.), (New York, NY, USA), pp. 300–315, ACM, 2017.

[58] N. Damouche and M. Martel, "Mixed precision tuning with salsa," in *Proceedings of the 8th International Joint Conference on Pervasive and Embedded Computing and Communication Systems, PECCS 2018, Porto, Portugal, July 29-30, 2018* (L. Gomes, A. Ahrens, C. Benavente-Peces, and M. S. Obaidat, eds.), pp. 185–194, SciTePress, 2018.

[59] S. Bessaï, D. Ben Khalifa, H. Benmaghnia, and M. Martel, "Fixed-point code synthesis based on constraint generation," in *Design and Architecture for Signal and Image Processing - 15th International Workshop, DASIP 2022, Budapest, Hungary, June 20-22, 2022, Proceedings* (K. Desnos and S. A. Pertuz, eds.), vol. 13425 of *LNCS*, (Cham), pp. 108–120, Springer, 2022.

[60] A. Zeller, "Yesterday, my program worked. today, it does not. why?," in *Software Engineering - ESEC/FSE'99, 7th European Software Engineering Conference, Held Jointly with the 7th ACM SIGSOFT Symposium on the Foundations of Software Engineering, Toulouse, France, September 1999, Proceedings* (O. Nierstrasz and M. Lemoine, eds.), vol. 1687 of *LNCS*, (Heidelberg), pp. 253–267, Springer, 1999.

[61] L. M. de Moura and N. S. Bjørner, "Z3: an efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings* (C. R. Ramakrishnan and J. Rehof, eds.), vol. 4963 of *LNCS*, (Berlin, Heidelberg), pp. 337–340, Springer Berlin Heidelberg, 2008.

[62] N. S. Bjørner and L. Nachmanson, "Arithmetic solving in Z3," in *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part I* (A. Gurfinkel and V. Ganesh, eds.), vol. 14681 of *LNCS*, (Cham), pp. 26–41, Springer, 2024.

[63] G. Kremer, A. Reynolds, C. W. Barrett, and C. Tinelli, "Cooperating techniques for solving nonlinear real arithmetic in the cvc5 SMT solver (system description)," in *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings*

(J. Blanchette, L. Kovács, and D. Pattinson, eds.), vol. 13385 of *LNCS*, (Cham), pp. 95–105, Springer, 2022.

[64] W. R. Inc., "Mathematica, Version 14.1." Champaign, IL, 2024.

[65] M. König, A. W. Bosman, H. H. Hoos, and J. N. van Rijn, "Critically assessing the state of the art in neural network verification," *J. Mach. Learn. Res.*, vol. 25, pp. 12:1–12:53, 2024.

[66] C. Brix, M. N. Müller, S. Bak, T. T. Johnson, and C. Liu, "First three years of the international verification of neural networks competition (VNN-COMP)," *Int. J. Softw. Tools Technol. Transf.*, vol. 25, no. 3, pp. 329–339, 2023.

[67] C. Brix, S. Bak, T. T. Johnson, and H. Wu, "The fifth international verification of neural networks competition (VNN-COMP 2024): Summary and results," *CoRR*, vol. abs/2412.19985, 2024.

[68] J. Jeannin, K. Ghorbal, Y. Kouskoulas, A. C. Schmidt, R. W. Gardner, S. Mitsch, and A. Platzer, "A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system," *Int. J. Softw. Tools Technol. Transf.*, vol. 19, no. 6, pp. 717–741, 2017.

[69] M. Wu, J. C. Rosenberg, and N. Fulton, "A formally verified plasma vertical position control algorithm," in *Formal Methods for Industrial Critical Systems - 25th International Conference, FMICS 2020, Vienna, Austria, September 2-3, 2020, Proceedings* (M. H. ter Beek and D. Nickovic, eds.), vol. 12327 of *LNCS*, (Cham), pp. 170–188, Springer, 2020.

[70] Y. Selvaraj, W. Ahrendt, and M. Fabian, "Formal development of safe automated driving using differential dynamic logic," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 988–1000, 2023.

[71] A. Platzer and E. M. Clarke, "The image computation problem in hybrid systems model checking," in *Hybrid Systems: Computation and Control, 10th International Workshop, HSCC 2007, Pisa, Italy, April 3-5, 2007, Proceedings* (A. Bemporad, A. Bicchi, and G. C. Buttazzo, eds.), vol. 4416 of *LNCS*, (Heidelberg), pp. 473–486, Springer, 2007.

[72] N. Fulton, S. Mitsch, J. Quesel, M. Völp, and A. Platzer, "KeYmaera X: an axiomatic tactical theorem prover for hybrid systems," in *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings* (A. P. Felty and A. Middeldorp, eds.), vol. 9195 of *LNCS*, (Cham), pp. 527–538, Springer, 2015.

[73] A. Kabra, S. Mitsch, and A. Platzer, "Verified train controllers for the federal railroad administration train kinematics model: Balancing competing brake and track forces," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4409–4420, 2022.

[74] A. Platzer and J. Quesel, "European train control system: A case study in formal verification," in *Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings* (K. K. Breitman and A. Cavalcanti, eds.), vol. 5885 of *LNCS*, pp. 246–265, Springer, 2009.

[75] R. Bohrer, Y. K. Tan, S. Mitsch, A. Sogokon, and A. Platzer, "A formal safety net for waypoint-following in ground robots," *IEEE Robotics Autom. Lett.*, vol. 4, no. 3, pp. 2910–2917, 2019.

[76] E. Prebet, S. Teuber, and A. Platzer, "Verification of autonomous neural car control with KeYmaera X," in *Rigorous State-Based Methods - 11th International Conference, ABZ 2025, Düsseldorf, Germany, Proceedings* (M. Leuschel and F. Ishikawa, eds.), vol. 15728 of *LNCS*, Springer, 2025.

[77] A. Kabra, J. Laurent, S. Mitsch, and A. Platzer, "CESAR: control envelope synthesis via angelic refinements," in *Tools and Algorithms for the Construction and Analysis of Systems - 30th International Conference, TACAS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part I* (B. Finkbeiner and L. Kovács, eds.), vol. 14570 of *LNCS*, pp. 144–164, Springer, 2024.

[78] S. Teuber, S. Mitsch, and A. Platzer, "samysweb/NCubeV: v0.9," Oct. 2024.

[79] A. Biere, "PicoSAT essentials," *J. Satisf. Boolean Model. Comput.*, vol. 4, no. 2-4, pp. 75–97, 2008.

[80] J. E. Holland, M. J. Kochenderfer, and W. A. Olson, "Optimizing the next generation collision avoidance system for safe, suitable, and acceptable operational performance," *Air Traffic Control Quarterly*, vol. 21, no. 3, pp. 275–297, 2013.

[81] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next generation airborne collision avoidance system," *Lincoln Laboratory Journal*, vol. 19, no. 1, pp. 17–33, 2012.

[82] C. Huang, J. Fan, X. Chen, W. Li, and Q. Zhu, "POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems," in *Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings* (A. Bouajjani, L. Holík, and Z. Wu, eds.), vol. 13505 of *LNCS*, (Cham), pp. 414–430, Springer, 2022.

[83] N. Noroozi, P. Karimaghaee, F. Safaei, and H. Javadi, "Generation of Lyapunov functions by neural networks," in *Proceedings of the World Congress on Engineering*, vol. 2008, 2008.

[84] S. M. Richards, F. Berkenkamp, and A. Krause, "The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, vol. 87 of *Proceedings of Machine Learning Research*, pp. 466–476, PMLR, 2018.

[85] A. Peruffo, D. Ahmed, and A. Abate, "Automated and formal synthesis of neural barrier certificates for dynamical models," in *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part I* (J. F. Groote and K. G. Larsen, eds.), vol. 12651 of *LNCS*, (Cham), pp. 370–388, Springer, 2021.

[86] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of Lyapunov neural networks," *IEEE Control. Syst. Lett.*, vol. 5, no. 3, pp. 773–778, 2021.

[87] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "FOSSIL: a software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks," in *HSCC '21: 24th ACM International Conference on Hybrid Systems: Computation and Control, Nashville, Tennessee, May 19-21, 2021* (S. Bogomolov and R. M. Jungers, eds.), (New York, NY, USA), pp. 24:1–24:11, ACM, 2021.

[88] P. Habeeb, L. Gupta, and P. Prabhakar, "Approximate conformance checking for closed-loop systems with neural network controllers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, pp. 4322–4333, 2024.

[89] M. K. Büning, P. Kern, and C. Sinz, "Verifying equivalence properties of neural networks with relu activation functions," in *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings* (H. Simonis, ed.), vol. 12333 of *LNCS*, pp. 868–884, Springer, 2020.

[90] B. Paulsen, J. Wang, and C. Wang, "Reludiff: Differential verification of deep neural networks," *Proceedings - International Conference on Software Engineering*, pp. 714–726, 2020. arXiv: 2001.03662 ISBN: 9781450371216.

[91] Y. Zhang, Z. Zhao, G. Chen, F. Song, M. Zhang, T. Chen, and J. Sun, "QVIP: an ilp-based formal verification approach for quantized neural networks," in *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*, pp. 82:1–82:13, ACM, 2022.

[92] P. Huang, H. Wu, Y. Yang, I. Daukantas, M. Wu, Y. Zhang, and C. W. Barrett, "Towards efficient verification of quantized neural networks," in *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada* (M. J. Wooldridge, J. G. Dy, and S. Natarajan, eds.), pp. 21152–21160, AAAI Press, 2024.

[93] A. Adjé, D. Ben Khalifa, and M. Martel, "Fast and efficient bit-level precision tuning," in *Static Analysis - 28th International Symposium, SAS 2021, Chicago, IL, USA, October 17-19, 2021, Proceedings* (C. Dragoi, S. Mukherjee, and K. S. Namjoshi, eds.), vol. 12913 of *LNCS*, (Cham), pp. 1–24, Springer, 2021.

[94] A. Kumar, V. Seshadri, and R. Sharma, "Shiftry: RNN inference in 2KB of RAM," *Proc. ACM Program. Lang.*, vol. 4, no. OOPSLA, pp. 182:1–182:30, 2020.

[95] E. Park, D. Kim, and S. Yoo, "Energy-efficient neural network accelerator based on outlier-aware low-precision computation," in *45th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2018, Los Angeles, CA, USA, June 1-6, 2018* (M. Annavaram, T. M. Pinkston, and B. Falsafi, eds.), pp. 688–698, IEEE Computer Society, 2018.

[96] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *45th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2018, Los*

Angeles, CA, USA, June 1-6, 2018 (M. Annavaram, T. M. Pinkston, and B. Falsafi, eds.), pp. 764–775, IEEE Computer Society, 2018.

[97] Z. Song, B. Fu, F. Wu, Z. Jiang, L. Jiang, N. Jing, and X. Liang, "DRQ: dynamic region-based quantization for deep neural network acceleration," in *47th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2020, Virtual Event / Valencia, Spain, May 30 - June 3, 2020*, pp. 1010–1021, IEEE, 2020.

[98] D. B. Khalifa and M. Martel, "Rigorous floating-point to fixed-point quantization of deep neural networks on STM32 micro-controllers," in *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 1201–1206, 2024.

[99] F. de Dinechin, "Reflections on 10 years of flopoco," in *26th IEEE Symposium on Computer Arithmetic, ARITH 2019, Kyoto, Japan, June 10-12, 2019* (N. Takagi, S. Boldo, and M. Langhammer, eds.), pp. 187–189, IEEE, 2019.