LLM2CERT: LLM SYNTHESIS OF CERTIFICATES WITH SMT/LEAN VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Lyapunov functions and barrier certificates give state-space guarantees of stability and safety, but synthesizing and verifying them—especially for nonlinear, high-dimensional systems—remains an open challenge. Existing pipelines verify candidates with numerical solvers (sum-of-squares programs), interval methods, or satisfiability-modulo-theories (SMT) solvers, which often require relaxations, are conservative, or scale poorly with model dimension and nonlinearity. These limitations have also shaped benchmark design: standard test suites are dominated by low-dimensional polynomial systems chosen to suit what solvers can handle rather than what engineering applications require. We address both sides of this challenge. First, we introduce DYNACERTBENCH, a benchmark of 105 instances: 80 non-redundant systems curated from the literature, extended with high-dimensional and control-constrained variants and new nonlinearly coupled systems, comprehensively spanning stability, region-of-attraction, forward-invariance, and safety tasks. Second, we present LLM2CERT, an LLM-driven framework that synthesizes candidate certificates and verifies them against their proof obligations: each obligation is routed to an SMT solver or directly to the Lean Proof-Repair agent. On DYNACERTBENCH, LLM2CERT verifies 81 of 105 instances (77.1%), a 17.1 percentage point improvement over one-shot proposal without feedback. On the 81 fixed certificates, LLM2CERT verifies 22 that All-SMT leaves unresolved.

1 INTRODUCTION

Safety-critical autonomous systems, such as robotic platforms, autonomous vehicles, and aerospace systems, require formal guarantees that go beyond finite simulation and testing. For dynamical systems, Lyapunov functions and barrier certificates are the standard mathematical witnesses of stability and safety (Khalil & Grizzle, 2002; Prajna et al., 2007). A Lyapunov function certifies stability by decreasing along system trajectories, while a barrier certificate certifies safety by defining a forward-invariant boundary that separates trajectories from unsafe regions. Because these certificates reduce trajectory-level properties to inequalities over the state space, they avoid explicit trajectory computation and are indispensable for analyzing complex nonlinear systems.

Verification methods for Lyapunov and barrier functions include optimization-based techniques such as sum-of-squares (SOS) programming, interval-based methods such as interval bound propagation (IBP), and formal methods based on Satisfiability Modulo Theories (SMT) solvers. Each comes with caveats. Numerical approaches such as SOS are affected by floating-point errors and numerical instability. SMT solvers for nonlinear real arithmetic can return results only up to a prescribed tolerance, such as δ-sat, and such tolerance-based verdicts do not establish the exact validity of the underlying proof obligation. In practice, both families also rely on ad-hoc relaxations, such as excluding a neighborhood of the origin or replacing transcendental terms by Taylor approximations Chang et al. (2019); Zhou et al. (2022); Taheri et al. (2025), and both scale poorly to high-dimensional state spaces and strongly nonlinear dynamics. These limitations motivate a complementary approach based on proof assistants such as Lean (Moura & Ullrich, 2021), in which the verified statement is an exact mathematical theorem checked by a small trusted kernel, rather than a numerically tolerated or approximately satisfiable surrogate.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

These methodological bottlenecks have not only limited verification itself but have also skewed how verification algorithms are evaluated. Standard benchmarks remain largely homogeneous: most instances are low-dimensional, parameter-free systems on tightly bounded domains. This lack of diversity does not reflect engineering requirements; rather, existing datasets are implicitly designed around the limits of the solvers used to verify them. High-dimensional benchmarks are scarce, and nearly all of those that exist are polynomial. Moving beyond these constraints requires both a rigorous verification paradigm and benchmark tasks that reflect richer, more realistic dynamics.

The major contributions of this work are summarized as follows.

1. **DYNACERTBENCH: A Comprehensive Benchmark for Verifying Dynamical Systems.** We curate and extend a benchmark that covers stability, region-of-attraction, forward-invariance, and safety verification for continuous- and discrete-time systems. It deliberately includes realistic scenarios that existing suites however avoid: high-dimensional state spaces, control-input constraints, strongly nonlinear coupling, and parameterized systems. Every instance is posed with its exact, unrelaxed certificate conditions.
2. **A Formal Verification Paradigm Combining SMT and Lean.** To the best of our knowledge, this is the first work to integrate the Lean theorem prover into the end-to-end synthesis and verification of Lyapunov and barrier certificates for both continuous- and discrete-time dynamical systems. Our method delegates proof obligations that SMT solvers cannot discharge to Lean, enabling the construction of rigorous mathematical proofs. This design eliminates the need for conservative assumptions commonly introduced by SMT-based approaches, including Taylor approximations, δ-satisfiability, and the exclusion of a neighborhood around the origin. Our approach thereby provides machine-checkable formal guarantees of safety and stability for the system under the explicitly stated assumptions.
3. **LLM2CERT: An Agentic Framework for Efficient Formal Verification.** We propose an LLM-driven framework that balances computational efficiency with mathematical soundness. To address the computational bottleneck of Lean, our framework adopts a hybrid verification strategy. Specifically, it utilizes SMT-based for faster verification and counterexample generation, while seamlessly integrating Lean to finalise formal proofs. The two backends are complementary: SMT efficiently discharges obligations, including some for which Lean proof search does not finish within the budget, while Lean resolves others left unresolved by SMT.

2 PROBLEM SETTING AND CERTIFICATES

2.1 DYNAMICAL SYSTEMS AND VERIFICATION TASKS

We consider continuous-time and discrete-time dynamical systems:

$$\dot{x}(t) = f(x(t), u(t)) \quad \text{or} \quad x_{k+1} = F(x_k, u_k), \tag{1}$$

where $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input and $f, F : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$. Given a state-feedback policy $\pi : \mathbb{R}^n \to \mathcal{U}$, we write the closed-loop dynamics as $f_\pi(x) := f(x, \pi(x))$ and $F_\pi(x) := F(x, \pi(x))$, and let $\phi_\pi^{x_0}(t)$ denote the unique closed-loop trajectory from x_0. Let $\mathcal{X} \subset \mathbb{R}^n$ be the domain of interest, and let $\mathcal{X}_I \subseteq \mathcal{X}$ and $\mathcal{X}_U$ denote the initial and unsafe sets, where either $\mathcal{X}_U \subseteq \mathcal{X}$ or $\mathcal{X}_U = \mathbb{R}^n \setminus \mathcal{X}$. Let $\mathcal{N}$ be a neighborhood of the origin, and let $\Omega_c := \{x : V(x) \leq c\}$ be the sublevel set of a Lyapunov candidate V for $c > 0$.

As summarized in Table 1, we consider four verification tasks concerning stability and safety: **(i) stability certification** of the equilibrium; **(ii) region-of-attraction (ROA) verification**; **(iii) forward-invariant set synthesis** avoiding an unsafe set $\mathcal{X}_U$; and **(iv) safety verification**, ensuring that trajectories from $\mathcal{X}_I$ never reach $\mathcal{X}_U$. Each task is established by a scalar certificate $W : \mathbb{R}^n \to \mathbb{R}$, which instantiates either a Lyapunov function V or a barrier function B. The change of W along the dynamics is the Lie derivative $\mathcal{D}_\pi W(x) := \nabla W(x)^\top f_\pi(x)$ in continuous time and the one-step difference $\mathcal{D}_\pi W(x) := W(F_\pi(x)) - W(x)$ in discrete time. Formal definitions and certificate conditions are given in Appendix C.

Remark 1. *Two aspects of our setting differ from prior work.* **(i) Verification vs. optimization:** *ROAs and invariant sets are generally not unique, and prior methods often optimize their size. We*

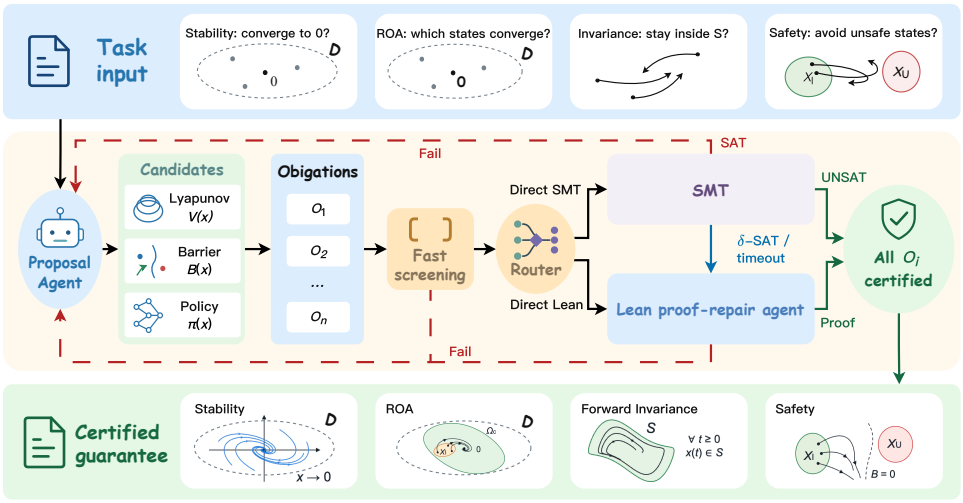

Figure 1: LLM2CERT framework. The Proposal Agent proposes a candidate certificate (and policy), which induces a list of proof obligations. Obligations without a compatible SMT encoding go directly to Lean; all others are checked by SMT first, and those on which the solvers time out or return δ-`sat` are escalated to the Lean proof-repair agent. Counterexamples and unresolved obligations are fed back to the Proposal Agent for the next candidate.

Table 1: Verification tasks, the corresponding certificate conditions, and the theorems (Appendix C) that connect them. Sets $\mathcal{X}$, $\mathcal{X}_I$, $\mathcal{X}_U$, $\mathcal{N}$, and Ω_c are defined in Section 2.1.

Task	**Certificate conditions**	**Theorems**
Stability	$V(0) = 0, V(x) > 0$ for $x \neq 0$, and $\mathcal{D}_\pi V \leq 0$ (or < 0 for $x \neq 0$)	Thm. 1
ROA	Lyapunov V and corr. sublevel set Ω_c with $\mathcal{X}_I \subseteq \Omega_c \subseteq \mathcal{N}$	Cor. 1
Invariant Set	Barrier B with $B < 0$ on $\mathcal{X}_U$, boundary/invariance condition	Thm. 2, Cor. 2
Safety	Barrier B with $B \geq 0$ on $\mathcal{X}_I$, $B < 0$ on $\mathcal{X}_U$, barrier condition	Cor. 3

instead verify the set induced by a proposed candidate W. **(ii) Control input:** *Vertovec et al. (2025) verify that an admissible control exists at each state ($\forall x\, \exists u$), whereas we verify W against a specific feedback policy $u = \pi(x)$, i.e., on the closed-loop system.*

2.2 Soundness of Certificate Conditions

Soundness has two layers. First, the certificate conditions for each task in Table 1 must imply the corresponding property: if a candidate satisfies the conditions, the theorem holds. We prove every such condition-to-theorem implication in Lean, and each proof is checked by the Lean. Second, given a candidate, we establish that it satisfies the certificate conditions. This layer is provided by LLM2CERT (Section 3.4), which accepts a candidate only when every condition has been discharged by an SMT `unsat` verdict or a kernel-checked Lean proof. Together, the two layers give the end-to-end guarantee Proposed candidate $\xrightarrow{\text{LLM2CERT}}$ Verified candidate $\xrightarrow{\text{Lean}}$ Theorems holds..

3 LLM2CERT: LLM-based Certificate Synthesis with SMT and Lean Verification

LLM2CERT combines LLM-based candidate synthesis with verification by SMT solvers and Lean (Figure 1). Throughout the verification of a candidate, the candidate and its obligations remain fixed: only proofs are searched for, and the target statements are never relaxed. The shaded boxes follow one single running example (System 88) through the whole section.

3.1 Candidate Search, Obligation List, and Fast Screening

Given a typed specification $R = (f/F, \mathcal{X}, \mathcal{N}, \mathcal{X}_I, \mathcal{X}_U, \mathcal{U}, \text{Task})$, the Proposal Agent proposes, at each iteration k, a symbolic certificate and, for controlled systems, a feedback policy. A format

check parses the response into a candidate $C^{(k)} = (W^{(k)}, \pi^{(k)})$; if parsing fails, the error message is returned to the Proposal Agent for correction. A parsed candidate is compiled into a typed expression tree $A^{(k)}$, and the certificate theorem of the task (Table 1) instantiated at $A^{(k)}$ yields the obligation list $\mathcal{M}(A^{(k)}) = \{O_1, \ldots, O_n\}$, with one obligation per theorem premise. The obligations retain the original domain, boundary, and parameter constraints of R without relaxation.

Before formal verification, fast screening evaluates each obligation on finite sampled states. A detected violation rejects the candidate, and the violating samples are returned to the Proposal Agent as counterexample feedback for the next candidate $C^{(k+1)}$. Because screening samples a continuous domain, passing it provides no guarantee: all obligations in $\mathcal{M}(A^{(k)})$ remain to be formally verified.

Running example: Candidate and obligations

We use the two-dimensional continuous-time safety instance (`Extend-S011`) (Taheri et al., 2025):

$$\dot{x}_1 = x_2 + u_0, \qquad \dot{x}_2 = -\frac{3}{10}x_1 - \frac{1}{5}(1 + \cos x_1)x_2 - \frac{1}{10}x_1^3 + u_1.$$

The specification is $\mathcal{X} = \mathbb{R}^2$, $\mathcal{X}_I = \{x : \|x\|_2 \le 3/5\}$, $\mathcal{X}_U = \{x : \|x - (2,2)^\mathsf{T}\|_2 \le 1\}$, and $\mathcal{U} = [-1, 1]^2$. At iteration $k = 1$, the Proposal Agent returns the candidate $C^{(1)} = (W^{(1)}, \pi^{(1)})$ with barrier function $W^{(1)}$ and policy $\pi^{(1)}(x) = (u_0(x), u_1(x))^\mathsf{T}$ where

$$W^{(1)}(x) = 1 - x_1^2 - x_2^2, \qquad u_0(x) = -\frac{x_1}{1 + x_1^2}, \qquad u_1(x) = -\frac{x_2}{1 + x_2^2}.$$

Writing $L_{f_{\pi^{(1)}}} W^{(1)}$ for the Lie derivative of $W^{(1)}$ along the closed-loop dynamics $f_{\pi^{(1)}}$, the safety task (Cor. 3) induces four obligations:

$$O_1 : \pi^{(1)}(x) \in \mathcal{U} \quad (x \in \mathcal{X}), \qquad O_2 : W^{(1)}(x) \ge 0 \quad (x \in \mathcal{X}_I),$$
$$O_3 : W^{(1)}(x) < 0 \quad (x \in \mathcal{X}_U), \qquad O_4 : L_{f_{\pi^{(1)}}} W^{(1)}(x) > 0 \quad (x \in \mathcal{X},\ W^{(1)}(x) = 0).$$

Fast screening finds no violation, so the candidate is passed to the Verification Router (Fig. 1), which dispatches each obligation to a backend: obligations without a compatible SMT encoding go to Lean, and all others go to SMT first and are escalated to Lean if the solvers δ-`sat` or timeout.

3.2 Verification Router

The Verification Router selects a backend for each obligation according to its type and solver compatibility. Obligations that have no compatible SMT encoding, such as radial unboundedness, are sent directly to Lean. All remaining obligations are sent to the compatible SMT solvers (Z3, cvc5, Yices, and dReal), each with a 30 s timeout. An obligation has the form $O_i : \forall z,\ P_i(z; R, C^{(k)}) \Rightarrow Q_i(z; R, C^{(k)})$, where z collects all quantified state and parameter variables and P_i retains the original premises; the solvers are given the counterexample query $\exists z,\ P_i(z; R, C^{(k)}) \wedge \neg Q_i(z; R, C^{(k)})$.

An SMT solver returns one of four outcomes: `unsat`, `sat`, δ-`sat`, or timeout. An `unsat` result verifies the obligation. A `sat` result means the candidate violates the obligation, and the solver's model is returned to the Proposal Agent as a counterexample for the next candidate $C^{(k+1)}$. A dReal δ-`sat` result is inconclusive: it establishes satisfiability only of a δ-weakened version of the query, not of the query itself, so it neither refutes nor verifies the obligation. Obligations that end in δ-`sat` or timeout therefore remain unresolved by SMT and are escalated to the Lean Proof-Repair Agent, which attempts to construct a kernel-checked proof of the unchanged statement in Lean.

Running example: SMT verification and escalation to Lean

cvc5 verifies O_1, and Z3 verifies O_2 and O_3. The remaining obligation O_4 contains $\cos x_1$, which Z3 and Yices do not support. dReal supports $\cos x_1$ but requires finite variable bounds, which are absent here because $\mathcal{X} = \mathbb{R}^2$. cvc5 is therefore the only applicable backend; it checks the counterexample query $\exists x \in \mathcal{X} : W^{(1)}(x) = 0 \wedge L_{f_{\pi^{(1)}}} W^{(1)}(x) \le 0$ but reaches the 30 s timeout. O_4 remains unresolved and is passed to Lean.

3.3 LEAN PROOF-REPAIR AGENT

For each obligation routed to Lean, the framework constructs the Lean context and theorem statement from the fixed candidate and obligation. The Lean Proof-Repair Agent may generate only the proof body, while the definitions, hypotheses, and theorem statement remain fixed. Each proof attempt is elaborated and checked in the pinned Lean environment.

A failed attempt returns *structured repair feedback*: the classified failure (an unknown identifier, an application type mismatch, or an unsolved goal), its source location, and the remaining goal. For an unknown identifier, the feedback also includes the declaration lines of matching lemmas found by a local search in Mathlib. The agent revises the failing part of its previous proof using this feedback.

A proof is accepted only if it type-checks against the fixed theorem and passes an audit that rejects proof placeholders (the `sorry` tactic in Lean) and any axiom beyond Lean's standard ones. The repair loop is bounded by a proof-time budget and a maximum number of rounds. If no proof is accepted within these limits, the obligation remains unresolved, and its last proof attempt and diagnostics are returned to the Proposal Agent to inform the synthesis of the next candidate $C^{(k+1)}$.

> **Running example: Repairing the proof of O_4 in Lean**
>
> The first two proof attempts fail on a missing lemma name and an implicit-argument mismatch; the compiler feedback guides a third attempt, which is accepted. On $W^{(1)} = 0$, the proof derives $x_1^2 + x_2^2 = 1$ and $x_i^2 \le 1$. Writing $q = -7/5 + x_1^2/5$, it establishes $qx_1x_2 \ge -7/10$, $2x_i^2/(1 + x_i^2) \ge x_i^2$, and $1 + \cos x_1 \ge 0$. The tactic `ring` exposes the identity
>
> $$L_{f_{\pi^{(1)}}} W^{(1)} = \frac{2x_1^2}{1+x_1^2} + \frac{2x_2^2}{1+x_2^2} + qx_1x_2 + \frac{2}{5}(1+\cos x_1)x_2^2,$$
>
> and `nlinarith` combines these bounds to conclude $L_{f_{\pi^{(1)}}} W^{(1)} \ge 1 - 7/10 = 3/10 > 0$.

3.4 SOUNDNESS AND TERMINATION

The outcome for a candidate is determined at the obligation level. A candidate $C^{(k)}$ is **verified** when every obligation in $\mathcal{M}(A^{(k)})$ has proof evidence, i.e., an SMT `unsat` result or an accepted kernel-checked Lean proof; **rejected** when fast screening detects a violation or SMT returns `sat`; and **unresolved** when at least one obligation is open after the Lean budget is exhausted. A rejected or unresolved candidate returns its counterexamples or failure diagnostics to the Proposal Agent, which generates $C^{(k+1)}$; the loop continues until a candidate is verified or the time budget is exhausted. Exhausting the candidate budget without a verified certificate is reported as **search exhaustion**.

> **Running example: Aggregating evidence**
>
> The recorded final label is `hybrid_verified`: O_1–O_3 are discharged by SMT with `unsat`, and O_4 by a Lean proof. This example shows how Lean discharges an obligation that no SMT solver could verify within the allotted time, and consequently how LLM2CERT establishes the exact safety conditions of Cor. 3 for the synthesized candidate.

4 DYNACERTBENCH: BENCHMARK CONSTRUCTION

4.1 MOTIVATION AND DESIGN GOALS

Existing certificate-synthesis benchmarks routinely compromise the verification target to accommodate solver limitations: they exclude a neighborhood of the origin, replace transcendental terms by Taylor approximations, or accept δ-`sat` verdicts. They have also remained largely homogeneous over the past decade, consisting mostly of low-dimensional polynomial systems. This implicit adaptation to solver limitations distorts the evaluation of algorithms. DYNACERTBENCH addresses this gap in two ways. First, every instance is posed with its exact verification target: as detailed in Appendix F, conditions that source papers verified *only approximately* are restated as the *strict*

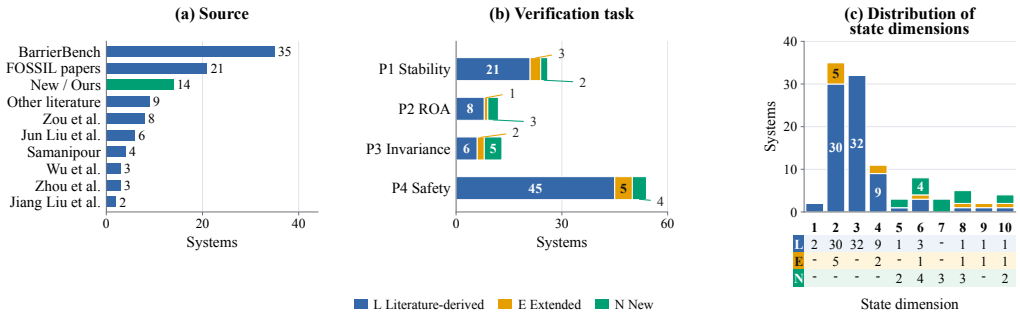

Figure 2: DYNACERTBENCH overview by (a) source, (b) verification task, and (c) state dimension, split into the Literature-derived (L), Extended (E), and New (N) tiers.

conditions of Table 1. Second, it systematically adds high-dimensional, control-constrained, and nonlinearly coupled scenarios.

4.2 DATA COLLECTION AND CURATION

To construct a non-redundant benchmark, we collected 233 candidate entries from 15 representative papers and BarrierBench (Taheri et al., 2025). We then applied five curation rules—merging exact duplicates, consolidating multiple configurations of the same dynamics, keeping one representative per system family, removing structurally redundant systems, and excluding out-of-scope cases (Appendix F, Table 14)—which left 80 distinct core systems (Table 15). Notably, only 7 of the 29 nominally high-dimensional entries ($n \geq 5$) survive curation (3 polynomial and 4 non-polynomial). To address this scarcity, we add two tiers beyond the **Literature-derived** core: **Extended** instances, which scale the state dimension, enlarge the domain, or add control constraints or perturbations to retained systems, and **New** instances, which are constructed high-dimensional systems with genuine nonlinear coupling. Unlike BarrierBench, which is limited to safety verification, DYNACERTBENCH spans all four tasks in Table 1.

4.3 DATASET STATISTICS AND ANALYSIS

Figure 2 summarizes the full benchmark by source, task, and state dimension. The Extended and New tiers concentrate the high-dimensional instances: they raise the number of instances with $n \geq 5$ from 7 to 25. In Section 5 we report results per tier, so that the added high-dimensional instances are not conflated with the literature-derived core.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks and Evaluation protocol. We evaluate LLM2CERT on DYNACERTBENCH (Section 4), reporting results separately for the Literature-derived, Extended, and New tiers. The benchmark covers stability, region-of-attraction, forward-invariance, and safety tasks for both continuous- and discrete-time systems. All methods are evaluated against the same original verification targets. A candidate counts as verified only if every obligation is discharged by a definitive SMT `unsat` result or a kernel-checked Lean proof.

Baselines. We compare LLM2CERT with five complete pipelines that pair different synthesizers with different verifiers (Table 2(a)). Method 1 synthesizes polynomial certificates by SMT-based CEGIS with Z3 and supports only polynomial systems (Ahmed et al., 2020). Method 2, FOSSIL 2.0, trains neural candidates by CEGIS and verifies them with its native SMT verifier (Edwards et al., 2024). Method 3, the neural-CBF method of Vertovec et al. (2025), learns neural candidates and certifies them with linear bound propagation (LBP). Method 4, the RL-based method of Zou et al. (2025), generates analytic Lyapunov expressions and verifies them with its native falsification and SMT pipeline. Method 5, SynNLF, learns a polynomial neural Lyapunov candidate and checks it through SOS constraints encoded as LMI feasibility problems (Zhao et al., 2025). For each baseline

Table 2: End-to-end synthesis and verification results. (a) Comparison with existing pipelines under the same time limit; each baseline runs on the subset of systems it supports, and the bottom row aggregates over the union of these subsets. (b) Ablations of LLM2CERT on the full 105-system benchmark under strict conditions.

(a) Comparison with existing end-to-end methods

Method	Synthesis	Verification	Systems	Native	Strict	LLM2CERT
1	SMT(Z3)	SMT(Z3)	8	0/8	–	8/8
2	NN CEGIS	SMT	49	8/49	5/8	37/49
3	NN CEGIS	LBP	8	4/8	4/4	7/8
4	RL	SMT	7	1/7	0/1	5/7
5	Poly. NN	SOS/LMI	8	7/8	3/7	8/8
Verified by at least one baseline			51	19/51	11/19	39/51

(b) End-to-end ablations for LLM2CERT

Method	Literature	Extended	New	Overall
One-shot Proposal[¶]	54/80	4/11	5/14	63/105
LLM2CERT (SMT only)	53/80	3/11	1/14	57/105
LLM2CERT (Lean only)	57/80	2/11	1/14	60/105
LLM2CERT (Full)	**66/80**	**7/11**	**8/14**	**81/105**

Note. [¶]This row is reconstructed from the successful runs that did not require feedback.

we report *Native*, the number of systems it verifies under its own conditions (e.g., with an excluded origin neighborhood or a δ-`sat` tolerance), and *Strict*, the number of those native certificates that also satisfy the unrelaxed obligations of Table 1 when re-checked by our verifier.

Comparisons. (1) End-to-end comparisons. We compare four configurations using the same implementation: *One-shot Proposal*, which generates one candidate without any feedback; LLM2CERT (SMT only); LLM2CERT (Lean only); and the full LLM2CERT (SMT with Lean). External tools are evaluated on their compatible benchmark subsets. **(2) Fixed candidate for backend analysis.** To isolate the effect of the verification backend, RQ1–RQ3 use a fixed candidate: we freeze the candidate certificates and the exact verification obligations produced by the full LLM2CERT run. Thus, all strategies are compared on identical proof obligations. This is necessary because different obligations makes the comparison of verification backend confounded. Overall, the former includes candidate search and feedback, whereas the latter holds the candidate certificates and verification obligations fixed to distinguish backend behavior from changes in candidate search.

Metrics and implementation. We report the number of verified systems, wall-clock time, and the number of candidate iterations. The Proposal Agent proposes at most $K_{\text{cand}} = 5$ candidates, one per round. For each obligation, the Lean Proof-Repair Agent uses at most $K_{\text{lean}} = 6$ rounds within $900\,\text{s}$. All methods are limited to a total wall-clock time of $T_{\text{run}} = 2400\,\text{s}$ per system. The full LLM2CERT run took $2.5\,\text{h}$. On the same fixed certificates (RQ1/RQ2), All-SMT took $0.25\,\text{h}$, All-Lean took $1.4\,\text{h}$ and the RQ3 repair sweep took $2.1\,\text{h}$. These wall-clock intervals sum to $6.2\,\text{h}$ and recorded 6.75M model tokens. Further implementation details are given in Appendix E.1.

5.2 MAIN EXPERIMENT: END-TO-END SYNTHESIS AND VERIFICATION

How does LLM2CERT compare with existing end-to-end synthesis methods? Table 2(a) compares complete synthesis and verification pipelines on their supported benchmark subsets. *Native* reports each baseline under the conditions stated in its paper, and *Strict* does not permit any change to the obligations. The baselines verify 19/51 instances natively, and 11 of these native successes remain valid under strict conditions (4 certificates are δ-`sat`, and 4 SynNLF certificates fail the strict recheck), whereas LLM2CERT verifies 39/51 instances.

Among these 51 systems, LLM2CERT verifies 22 that no baseline verifies natively, and 17 that a baseline also verifies. Conversely, the baselines verify two systems that LLM2CERT leaves unresolved. This suggests that combining neural certificate synthesis with LLM2CERT is a promising direction for future research. Overall, these results demonstrate broader end-to-end coverage of LLM2CERT under unchanged obligations.

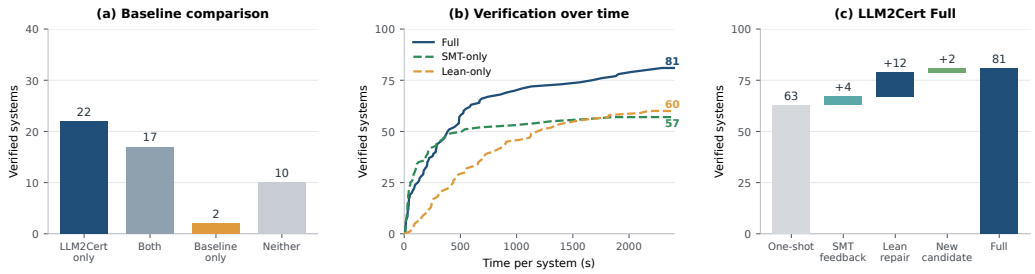

Figure 3: **End-to-end results.** (a) Systems verified by LLM2CERT and by the union of baselines on the 51 baseline-supported systems. (b) Verified systems vs. per-system wall-clock time for the three LLM2CERT configurations. (c) Gain contributed by each component over one-shot proposal.

5.3 END-TO-END ABLATIONS

What do the feedback loop and Lean contribute to LLM2CERT? LLM proposal alone is not sufficient. Table 2(b) shows that One-shot Proposal verifies only 63/105 instances, whereas the full LLM2CERT verifies 81/105, an improvement of 15.0, 27.3, and 21.4 percentage points on the Literature-derived, Extended, and New tiers, respectively. The SMT-only configuration corresponds to the setup of BarrierBench (Taheri et al., 2025), which also feeds SMT counterexamples back to the LLM. SMT-only and Lean-only verify 57/105 and 60/105 systems, respectively, whereas the full framework reaches 81/105, showing that neither backend alone recovers the coverage of their integration. LLM2CERT also verifies 8 of the 10 systems that Taheri et al. (2025) reports as unsolved. Fig. 3(b) shows verified systems over time for the three configurations, and Fig. 3(c) attributes the gain over one-shot proposal to SMT feedback, Lean proof repair, and new candidates. These results show that feedback and proof repair are most effective when combined within the full framework: counterexamples and unresolved obligations guide subsequent candidates, while routing and structured proof repair discharge the obligations that one-shot synthesis leaves open. The results also demonstrate that Lean and SMT play complementary roles in LLM2CERT.

5.4 VERIFICATION FRAMEWORK ANALYSIS

To separate the verification backend from candidate search, RQ1–RQ3 use fixed candidates. **RQ1** asks when Lean adds value over SMT-only verification; **RQ2** asks whether per-obligation routing improves success and cost; **RQ3** asks whether structured, compiler-guided feedback improves Lean proof repair. RQ1 and RQ2 fix the 81 certificates verified by the full framework, together with their 296 obligations, and RQ3 fixes the failed proof attempts from those runs. An analysis of the 24 unresolved systems is given in Appendix E.5.

RQ1: When does Lean add value beyond SMT-only verification? We compare three strategies on the same obligations: ALL-SMT, ALL-LEAN, and ROUTED. ALL-SMT and ALL-LEAN allow 900 s per obligation. ROUTED gives SMT 30 s first and sends unresolved obligations to Lean with a 900 s limit. Table 3 shows that Lean's additional coverage is concentrated on the non-polynomial.

The two backends exhibit complementary coverage. ALL-LEAN verifies 8 certificates that ALL-SMT leaves unresolved, all of them non-polynomial, while ALL-SMT resolves additional certificates for which Lean proof search does not finish within the budget. On polynomial systems, SMT already covers all certificates verified by Lean, whereas Lean contributes exclusive successes on the non-polynomial subset. This complementary behavior motivates obligation-level routing: SMT handles obligations in its effective fragments efficiently, while Lean extends coverage beyond them.

RQ2: Does per-obligation routing improve success and time cost? The two backends have different coverage–time profiles. ALL-SMT obtains its coverage early: 58 of the 81 fixed certificates are verified within 30 s per obligation, and this number rises only to 59 at 900 s. ALL-LEAN verifies 42/81. On the systems it verifies, its mean time is 490.58 s for polynomial systems and 893.17 s for non-polynomial systems, against 1.65 s and 8.30 s for ALL-SMT, and 39 certificates remain unresolved. SMT is thus fast where it succeeds. Lean verifies 8 certificates that SMT leaves unresolved,

Table 3: Fixed-candidate verification on the 81 certificates verified by the full framework, split by polynomial and non-polynomial systems.

	Polynomial ($N_{\text{sys}} = 41$)			Non-polynomial ($N_{\text{sys}} = 40$)		
Strategy	Verified	Time/sys.	Lean/sys.	Verified	Time/sys.	Lean/sys.
ALL-SMT	**41/41**	**1.65**	0.0%	18/40	**8.30**	0.0%
ALL-LEAN	25/41	490.58	85.4%	17/40	893.17	85.0%
ROUTED	**41/41**	**2.49**	0.0%	**40/40**	**239.05**	72.5%

Note. Time/sys.: mean verification time (s) over each strategy's verified systems. Lean/sys.: fraction of systems with a Lean-verified obligation.

Table 4: Lean proof repair under three feedback strategies ($N_{\text{obl}} = 26$).

Strategy	Repaired obl.	Rounds/obl.	Time/obl.
GENERIC RETRY	1/26	3.58	873.66
RAW DIAGNOSTICS	17/26	2.35	491.02
STRUCTURED REPAIR†	**21/26**	**2.19**	**407.42**

Note. Times in seconds. Unrepaired obligations are charged the full 900 s budget. †Measured from the recorded RQ1 traces.

all non-polynomial, while SMT verifies 25 certificates that Lean unresolved. This complementarity motivates per-obligation routing: default to SMT and fall back to Lean only when necessary.

ROUTED implements this strategy and verifies all 81 certificates. Of the 296 obligations, SMT discharges 253 (85.5%) by `unsat`, and 43 reach Lean: 21 after an SMT result other than `unsat` (8 δ-`sat`, 13 timeout or solver error), 19 by a direct router decision, and 3 with no SMT encoding (radial unboundedness). Selective use of Lean thus covers more than either backend alone at a fraction of the all-Lean cost (Table 3).

RQ3: Does structured repair feedback improve Lean proof completion? Lean is the expensive backend, so the cost of LLM2CERT depends on how quickly the Lean Proof-Repair Agent converges. The agent used in RQ1–RQ2 already receives our structured repair feedback (Section 3.3); RQ3 isolates the contribution of this feedback by replacing it with less informative alternatives.

Among the 105 systems, 26 obligations from 19 systems required Lean repair: 18 after SMT and 8 through direct routing. Of these, 21 were eventually repaired and 5 remained unresolved. Starting from the same failed first attempts, we compare STRUCTURED REPAIR with two alternatives: GENERIC RETRY, which reports only failure, and RAW DIAGNOSTICS, which returns the unprocessed Lean compiler output, each under a $900,\text{s}$ budget per obligation.

Table 4 shows that feedback quality substantially affects proof repair. GENERIC RETRY repairs only 1/26 obligations and no complete system, whereas RAW DIAGNOSTICS repairs 17/26 obligations and 13/19 systems. STRUCTURED REPAIR achieves the highest completion rate, repairing 21/26 obligations and 15/19 systems, while requiring fewer rounds and reducing mean repair time to $557.52,\text{s}$ per system, 17.0% below RAW DIAGNOSTICS. Structured feedback therefore improves both proof completion and the efficiency of Lean verification within LLM2CERT.

6 CONCLUSION

We studied certificate synthesis and verification for dynamical systems end-to-end. Solver limitations have long pushed both certificate verification and benchmark design toward conservative relaxations of the true stability and safety conditions. We instead keep the original, unrelaxed certificate conditions as a fixed target and introduce LLM2CERT, an agentic framework that synthesizes candidate certificates and verifies every obligation either by an SMT solver or by a kernel-checked Lean proof, routing per obligation and falling back to Lean when solvers time out or return δ-`sat`. On DYNACERTBENCH, a curated benchmark that restores the high-dimensional, constrained, and nonlinearly coupled systems that solver-oriented datasets have excluded, LLM2CERT verifies 77.1% of instances end to end and, where solvers alone fail, obtains machine-checked proofs of the exact conditions rather than of a relaxed surrogate. We see this as a foundational first step toward kernel-checked proof as a practical verifier for synthesized dynamical-system certificates.

AI USE STATEMENT

Generative AI tools (OpenAI GPT-6 Astra and GPT-5.6-sol) were used to assist with and clean up code, and to proofread this manuscript and correct grammar. The authors have reviewed all AI-assisted work, and take full responsibility for the final content.

REPRODUCIBILITY STATEMENT

Appendix E.1 lists the hardware, the pinned Lean 4 and Mathlib revision, the SMT solver versions and settings, the LLM configuration, and all budgets (K_{cand}, K_{lean}, T_{run}, and the per-solver and per-obligation timeouts) used in the reported runs, and Algorithms 1 and 2 give the exact search and verification procedures. Appendix D gives the complete serialized prompts, JSON schemas, and case traces are provided in the supplementary material. The certificate conditions and the Lean-proved condition-to-theorem implications are stated in Appendix C. Appendix F specifies every instance of DYNACERTBENCH (dynamics, domains, initial and unsafe sets, control constraints, and task) together with the curation rules and source-by-source counts (Tables 14 and 15), and Appendix E reports the per-task, per-obligation, and cost breakdowns behind the main-text tables. The code of LLM2CERT, the benchmark files, and the verified certificates with their SMT logs and Lean proof scripts are included in the supplementary material and will be released publicly upon acceptance.

ETHICS STATEMENT

This work does not involve human subjects, personal data, or crowdsourced annotation. The benchmark instances are derived from published papers and from BarrierBench, all of which are cited in Appendix F. We are not aware of any foreseeable harmful applications specific to this work.

REFERENCES

Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. Fossil: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th international conference on hybrid systems: computation and control*, pp. 1–11, 2021.

Ayush Agrawal and Koushil Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *Robotics: Science and systems*, volume 13, pp. 1–10. Cambridge, MA, USA, 2017.

Daniele Ahmed, Andrea Peruffo, and Alessandro Abate. Automated and sound synthesis of lyapunov functions with smt solvers. In *International conference on tools and algorithms for the construction and analysis of systems*, pp. 97–114. Springer, 2020.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pp. 3420–3431. Ieee, 2019.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

Katherine M Collins, Simon Frieder, Jonas Bayer, Jacob Loader, Jeck Lim, Peiyang Song, Fabian Zaiser, Lexin Zhou, Shanda Li, Sam Looi, et al. Characterizing initial human-ai proof formalization workflows. *arXiv preprint arXiv:2606.04273*, 2026.

Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

Grigory Devadze, Victor Magron, and Stefan Streif. Computer-assisted proofs for lyapunov stability via sums of squares certificates and constructive analysis: G. devadze et al. *Journal of Automated Reasoning*, 69(1):2, 2025.

Moritz Doll and Iman Shames. Foundations of machine-checked control theory in lean. *arXiv preprint arXiv:2607.19727*, 2026.

Alec Edwards, Andrea Peruffo, and Alessandro Abate. Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models. In *Proceedings of the 27th ACM international conference on hybrid systems: computation and control*, pp. 1–10, 2024.

Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. Keymaera x: An axiomatic tactical theorem prover for hybrid systems. In *International Conference on Automated Deduction*, pp. 527–538. Springer, 2015.

Sicun Gao, Soonho Kong, and Edmund M Clarke. dreal: An smt solver for nonlinear theories over the reals. In *International conference on automated deduction*, pp. 208–214. Springer, 2013.

Robert Joseph George, Suozhi Huang, Peiyang Song, and Anima Anandkumar. Leanprogress: Guiding search for neural theorem proving via proof progress prediction. *arXiv preprint arXiv:2502.17925*, 2025.

Robert Joseph George, Jennifer Cruden, Will Adkisson, Xiangru Zhong, Huan Zhang, and Anima Anandkumar. Torchlean: Formalizing neural networks in lean. *arXiv preprint arXiv:2602.22631*, 2026.

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026.

Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

Adarsh Kumarappan, Mohit Tiwari, Peiyang Song, Robert Joseph George, Chaowei Xiao, et al. Leanagent: Lifelong learning for formal theorem proving. In *International Conference on Learning Representations*, volume 2025, pp. 73525–73564, 2025.

Haoyu Li, Xiangru Zhong, Bin Hu, and Huan Zhang. Two-stage learning of stabilizing neural controllers via zubov sampling and iterative domain expansion. *Advances in Neural Information Processing Systems*, 38:73980–74009, 2026a.

Zenan Li, Ziran Yang, Peiyang Song, Zhaoyu Li, and Kaiyu Yang. P ^{3}: Joint program-and-proof planning for verified code generation. *arXiv preprint arXiv:2608.09277*, 2026b.

Xiaohan Lin, Qingxing Cao, Yinya Huang, Haiming Wang, Jianqiao Lu, Zhengying Liu, Linqi Song, and Xiaodan Liang. Fvel: Interactive formal verification environment with large language models via theorem proving. *Advances in Neural Information Processing Systems*, 37:54932–54946, 2024.

Jiang Liu, Naijun Zhan, Hengjun Zhao, and Liang Zou. Abstraction of elementary hybrid systems by variable transformation. In *International Symposium on Formal Methods*, pp. 360–377. Springer, 2015.

Jun Liu and Maxwell Fitzsimmons. A globally asymptotically stable planar homogeneous polynomial vector field with no polynomial lyapunov function. *arXiv preprint arXiv:2607.16171*, 2026.

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Towards learning and verifying maximal neural lyapunov functions. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 8012–8019. IEEE, 2023.

Jun Liu, Yiming Meng, Maxwell Fitzsimmons, and Ruikun Zhou. Tool lyznet: A lightweight python tool for learning and verifying neural lyapunov functions and regions of attraction. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 1–8, 2024.

Udayan Mandal, Guy Amir, Haoze Wu, Ieva Daukantas, Fletcher Lee Newell, Umberto J Ravaioli, Baoluo Meng, Michael Durling, Milan Ganai, Tobey Shim, et al. Formally verifying deep reinforcement learning controllers with lyapunov barrier certificates. In *FMCAD*, pp. 95–106, 2024.

Frederik Baymler Mathiesen, Simeon C Calvert, and Luca Laurenti. Safety certification for stochastic systems via neural barrier functions. *IEEE Control Systems Letters*, 7:973–978, 2022.

Yiming Meng, Ruikun Zhou, Amartya Mukherjee, Maxwell Fitzsimmons, Christopher Song, and Jun Liu. Physics-informed neural network policy iteration: Algorithms, convergence, and verification. *arXiv preprint arXiv:2402.10119*, 2024.

Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *International Conference on Automated Deduction*, pp. 625–635. Springer, 2021.

Azim Ospanov, Farzan Farnia, and Roozbeh Mohit. Apollo: Automated llm and lean collaboration for advanced formal reasoning. *Advances in Neural Information Processing Systems*, 38:41599–41633, 2026.

Antonis Papachristodoulou and Stephen Prajna. On the construction of lyapunov functions using the sum of squares decomposition. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pp. 3482–3487. IEEE, 2002.

Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In *International conference on tools and algorithms for the construction and analysis of systems*, pp. 370–388. Springer, 2021.

Stephen Prajna, Ali Jadbabaie, and George J Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8): 1415–1428, 2007.

Hadi Ravanbakhsh and Sriram Sankaranarayanan. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots*, 43(2):275–307, 2019.

Mengxin Ren and Hanrui Zhao. Formal safety verification for nonlinear systems with generative barrier certificate. In *International Conference on Intelligent Computing*, pp. 245–256. Springer, 2026.

Damien Rouhling. A formal proof in coq of a control function for the inverted pendulum. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pp. 28–41, 2018.

Pouya Samanipour and Hasan A Poonawala. Relu barrier functions for nonlinear systems with constrained control: A union of invariant sets approach. *arXiv preprint arXiv:2603.15286*, 2026.

Andrew Sogokon, Khalil Ghorbal, Yong Kiam Tan, and André Platzer. Vector barrier certificates and comparison systems. In *International Symposium on Formal Methods*, pp. 418–437. Springer, 2018.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Lean copilot: Large language models as copilots for theorem proving in lean. *arXiv preprint arXiv:2404.12534*, 2024.

Ali Taheri, Alireza Taban, Sadegh Soudjani, and Ashutosh Trivedi. Barrierbench: Evaluating large language models for safety verification in dynamical systems. *arXiv preprint arXiv:2511.09363*, 2025.

Amitayush Thakur, George Tsoukalas, Yeming Wen, Jimmy Xin, and Swarat Chaudhuri. An in-context learning agent for formal theorem-proving. *arXiv preprint arXiv:2310.04353*, 2023.

Amitayush Thakur, Jasper Lee, George Tsoukalas, Meghana Sistla, Matthew Zhao, Stefan Zetzsche, Greg Durrett, Yisong Yue, and Swarat Chaudhuri. Clever: A curated benchmark for formally verified code generation. *Advances in Neural Information Processing Systems*, 38, 2026.

Goli Vaisi, James Ferlez, and Yasser Shoukry. Extracting forward invariant sets from neural network-based control barrier functions. In *Proceedings of the 28th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 1–11, 2025.

Nikolaus Vertovec, Frederik Baymler Mathiesen, Thom Badings, Luca Laurenti, and Alessandro Abate. Scalable verification of neural control barrier functions using linear bound propagation. *arXiv preprint arXiv:2511.06341*, 2025.

Evan Wang, Simon Chess, Daniel Lee, Siyuan Ge, Ajit Mallavarapu, Jarod Alper, and Vasily Ilin. Learning to repair lean proofs from compiler feedback. *arXiv preprint arXiv:2602.02990*, 2026.

Qiuye Wang, Mingshuai Chen, Bai Xue, Naijun Zhan, and Joost-Pieter Katoen. Encoding inductive invariants as barrier certificates: Synthesis via difference-of-convex programming. *Information and Computation*, 289:104965, 2022.

Hao Wu, Shenghua Feng, Ting Gan, Jie Wang, Bican Xia, and Naijun Zhan. On completeness of sdp-based barrier certificate synthesis over unbounded domains. In *International Symposium on Formal Methods*, pp. 248–266. Springer, 2024.

Junlin Wu, Andrew Clark, Yiannis Kantaros, and Yevgeniy Vorobeychik. Neural lyapunov control for discrete-time systems. *Advances in neural information processing systems*, 36:2939–2955, 2023.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. *Advances in neural information processing systems*, 35:32353–32368, 2022.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*, 2024.

Bai Xue. Sufficient and necessary barrier-like conditions for safety and reach-avoid verification of stochastic discrete-time systems. *Automatica*, 187:112919, 2026.

Bai Xue, Qiuye Wang, Naijun Zhan, and Martin Fränzle. Robust invariant sets generation for state-constrained perturbed polynomial systems. In *Proceedings of the 22nd ACM international conference on hybrid systems: Computation and control*, pp. 128–137, 2019.

Bai Xue, Naijun Zhan, Martin Fränzle, Ji Wang, and Wanwei Liu. Reach-avoid verification based on convex optimization. *IEEE Transactions on Automatic Control*, 69(1):598–605, 2023.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in neural information processing systems*, 36:21573–21612, 2023.

Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. Formal reasoning meets llms: Toward ai for mathematics and verification. *Communications of the ACM*, 69(3):66–73, 2026.

Zhe Ye, Hantao Lou, Yuechun Sun, Peiyang Song, Zhengxu Yan, Timothe Kasriel, Qingyang Zhang, Kaiyu Yang, Soonho Kong, Jingxuan He, et al. Vero: Can ai agents build formally verified software repositories? *arXiv preprint arXiv:2608.13522*, 2026a.

Zhe Ye, Zhengxu Yan, Jingxuan He, Timothe Kasriel, Kaiyu Yang, and Dawn Song. Verina: Benchmarking verifiable code generation. In *International Conference on Learning Representations*, volume 2026, pp. 38933–38972, 2026b.

Hongchao Zhang, Junlin Wu, Yevgeniy Vorobeychik, and Andrew Clark. Exact verification of relu neural control barrier functions. *Advances in neural information processing systems*, 36:5685–5705, 2023.

Hongchao Zhang, Zhizhen Qin, Sicun Gao, and Andrew Clark. Seev: Synthesis with efficient exact verification for relu neural barrier functions. *Advances in Neural Information Processing Systems*, 37:101367–101392, 2024.

Hanrui Zhao, Niuniu Qi, Mengxin Ren, Banglong Liu, Shuming Shi, and Zhengfeng Yang. Learning-enabled polynomial lyapunov function synthesis via high-accuracy counterexample-guided framework. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10275–10284. IEEE, 2025.

Haoyu Zhao, Ziran Yang, Jiawei Li, Deyuan He, Zenan Li, Chi Jin, Venugopal V Veeravalli, Aarti Gupta, and Sanjeev Arora. Algoveri: An aligned benchmark for verified code generation on classical algorithms. *arXiv preprint arXiv:2602.09464*, 2026.

Hengjun Zhao, Xia Zeng, Taolue Chen, and Zhiming Liu. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, pp. 1–11, 2020.

Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural lyapunov control of unknown nonlinear systems with stability guarantees. *Advances in Neural Information Processing Systems*, 35:29113–29125, 2022.

Haohan Zou, Jie Feng, Hao Zhao, and Yuanyuan Shi. Analytical lyapunov function discovery: An rl-based generative approach. *arXiv preprint arXiv:2502.02014*, 2025.

A Limitations and Future Work

Our current framework has several limitations that suggest natural directions for future work. First, DynaCertBench, while broader than prior benchmarks along the dimensionality and coupling axes, remains a curated collection of 105 instances rather than an exhaustive sample of certificate-synthesis problems. Extending the benchmark to a wider range of high-dimensional, hybrid, and uncertain dynamics would provide a more comprehensive evaluation of formal certificate synthesis. Second, the present formulation focuses primarily on deterministic dynamical systems; extending the framework to stochastic systems with random disturbances and probabilistic guarantees is an important next step. Third, the current synthesis loop targets certificates for prescribed verification domains rather than explicitly optimizing the certified set itself. Incorporating region-size objectives into the proposal and verification loop could enable the formal synthesis of larger regions of attraction and invariant sets.

B Related work

Certificate synthesis and learner–verifier loops. Functional certificates turn trajectory-level properties into inequalities over the state space, but the certificate itself must still be discovered. Certificate synthesis typically restricts the witness to a parameterized family, using SOS or semidefinite relaxations for polynomial templates (Papachristodoulou & Prajna, 2002; Prajna et al., 2007) and bilinear or difference-of-convex formulations when certificates interact with multipliers or controllers (Xue et al., 2019; Wang et al., 2022; Xue et al., 2023). Learner–verifier loops instead alternate candidate search with global counterexample generation. This pattern spans polynomial and SMT-guided synthesis (Ravanbakhsh & Sankaranarayanan, 2019; Ahmed et al., 2020) as well as neural Lyapunov functions, barrier certificates, and controller co-synthesis (Chang et al., 2019; Abate et al., 2020; Peruffo et al., 2021; Dai et al., 2021; Abate et al., 2021; Edwards et al., 2024). The candidate representation determines the verifier: analytic expressions favor nonlinear SMT, piecewise-linear networks admit mixed-integer encodings, and neural certificates motivate activation-region or bound-propagation methods (Gao et al., 2013; Wu et al., 2023; Zhang et al., 2023; 2024; Vertovec et al., 2025; Li et al., 2026a). Language models have recently proposed and repaired symbolic barriers using numerical, SOS, or SMT feedback (Taheri et al., 2025; Ren & Zhao, 2026). Across these approaches, soundness rests on the final verifier, while template choices, strengthenings, excluded neighborhoods, and numerical approximations can change the proposition actually certified.

Formal Verification of Dynamical Systems. For a fixed certificate, SMT checks the existential negation of universal inequalities, including nonlinear transcendental formulas through delta-complete procedures (Gao et al., 2013). Algebraic methods seek checkable nonnegativity decompositions, whereas deductive hybrid-system provers reason about differential and discrete transitions in a specialized calculus (Papachristodoulou & Prajna, 2002; Prajna et al., 2007; Fulton et al., 2015). These artifacts provide different assurance: a delta-qualified verdict, an exact algebraic certificate, and a deductive proof are not interchangeable. Proof assistants further reduce the trusted base by checking explicit proof terms. Control-theoretic developments include a Rocq stability proof for an inverted-pendulum controller and constructive checking of exact SOS Lyapunov certificates in Minlog (Rouhling, 2018; Devadze et al., 2025). Recent Lean work formalizes general stability theory, neural-network semantics, and checked bounds for learned controllers (Doll & Shames, 2026; George et al., 2026). Related verification pipelines also check learned controllers against Lyapunov or barrier conditions (Mandal et al., 2024). Our setting connects these layers: it freezes each synthesized candidate's original contract, decomposes it into obligations, and seeks a replayable Lean proof rather than only a solver verdict.

LLMs for Formal and Agentic Reasoning. LLMs support autoformalization (Wu et al., 2022) and proof generation through retrieved premises, tactic suggestions, synthetic data, and reinforcement learning (Yang et al., 2023; Song et al., 2024; Xin et al., 2024; Hubert et al., 2026; George et al., 2025). Agentic systems additionally exploit proof-state history and backtracking (Thakur et al., 2023; Kumarappan et al., 2025) or repair failed subproofs from compiler diagnostics (Ospanov et al., 2026; Wang et al., 2026). Theorem-proving environments have also been used for LLM-assisted program verification (Lin et al., 2024; Zhao et al., 2026; Ye et al., 2026b; Thakur et al., 2026; Ye et al., 2026a; Li et al., 2026b). Most such systems begin with a fixed formal theorem

(Collins et al., 2026; Yang et al., 2026). Certificate synthesis adds an earlier semantic choice: the agent must select a witness whose exact stability or safety obligations are both true and provable. We therefore separate candidate synthesis, route selection, and Lean proof construction, returning structured failures without allowing the loop to change the frozen theorem.

C Dynamical Models and Notation

Let $\mathbb{R}$ and $\mathbb{R}^n$ denote the sets of real numbers and n-dimensional real vectors, respectively. We consider continuous-time and discrete-time dynamical systems of the form

$$\dot{x}(t) = f(x(t), u(t)), \tag{2}$$

$$x_{k+1} = F(x_k, u_k), \tag{3}$$

where $u \in \mathcal{U} \subseteq \mathbb{R}^m$ denotes the control input, $f : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$ and $F : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$. Given a state-feedback policy $\pi : \mathbb{R}^n \to \mathcal{U}$, define the closed-loop dynamics as

$$f_\pi(x) := f(x, \pi(x)), \qquad F_\pi(x) := F(x, \pi(x)).$$

C.1 System Properties

Given a policy π and an initial state $x_0 \in \mathbb{R}^n$, Throughout this paper, $t \geq 0$ denotes $t \in [0, \infty)$ for continuous-time systems and $t \in \mathbb{N}_0$ for discrete-time systems. Let $\phi_\pi^{x_0}$ denote the corresponding closed-loop trajectory, the trajectory satisfies $\phi_\pi^{x_0}(0) = x_0$ and

$$\dot{\phi}_\pi^{x_0}(t) = f_\pi\big(\phi_\pi^{x_0}(t)\big)$$

for continuous-time systems, and

$$\phi_\pi^{x_0}(t+1) = F_\pi\big(\phi_\pi^{x_0}(t)\big)$$

for discrete-time systems. We assume that the closed-loop trajectory is uniquely defined for all future time.

Definition 1 (Equilibrium point)**.** *A state $x^* \in \mathbb{R}^n$ is an equilibrium point of the closed-loop system if $f_\pi(x^*) = 0$ in continuous time or $F_\pi(x^*) = x^*$ in discrete time. Without loss of generality, we take the equilibrium point of interest to be the origin.*

Problem 1 (Stability certification)**.** *Given the equilibrium point $x^* = 0$, determine whether it is stable, asymptotically stable, or exponentially stable under the closed-loop dynamics.*

Problem 2 (Region-of-attraction verification)**.** *Given a set of initial states $\mathcal{X}_I \subseteq \mathbb{R}^n$, determine whether*

$$\lim_{t \to \infty} \phi_\pi^{x_0}(t) = 0, \qquad \forall x_0 \in \mathcal{X}_I.$$

If the property holds, we write $\mathcal{X}_I \subseteq \mathcal{R}_A(0)$.

Problem 3 (Forward-invariant set synthesis)**.** *Given a domain of interest $\mathcal{X} \subseteq \mathbb{R}^n$ and an unsafe set $\mathcal{X}_U \subseteq \mathcal{X}$(or $\mathcal{X}_U = \mathbb{R}^n \setminus \mathcal{X}$), find a nonempty set $\mathcal{S} \subseteq \mathcal{X}$ such that $\mathcal{S}$ is disjoint from $\mathcal{X}_U$ and is forward invariant under the dynamics, i.e., every trajectory starting from $\mathcal{S}$ remains in $\mathcal{S}$ for all future time. Formally:*

$$\mathcal{S} \cap \mathcal{X}_U = \varnothing \quad \textit{and} \quad \phi_\pi^{x_0}(t) \in \mathcal{S}, \quad \forall x_0 \in \mathcal{S},\ \forall t \geq 0.$$

Problem 4 (Safety verification,Prajna et al. (2007))**.** *Given a domain of interest $\mathcal{X} \subseteq \mathbb{R}^n$, an initial set $\mathcal{X}_I \subseteq \mathcal{X}$, and an unsafe set $\mathcal{X}_U \subseteq \mathcal{X}$(or $\mathcal{X}_U = \mathbb{R}^n \setminus \mathcal{X}$) with $\mathcal{X}_I \cap \mathcal{X}_U = \varnothing$, determine that there does not exist any time instant $T \geq 0$ and initial state $x_0 \in \mathcal{X}_I$ such that the trajectory $\phi_\pi^{x_0}(t)$ remains in $\mathcal{X}$ for all $t \in [0, T)$ but reaches $\mathcal{X}_U$ at time T, i.e.,*

$$\forall x_0 \in \mathcal{X}_I, \forall T \in (0, \infty), [\forall t \in [0, T], \phi_\pi^{x_0}(t) \in \mathcal{X}] \implies [\forall t \in [0, T], \phi_\pi^{x_0}(t) \in \mathcal{X} \setminus \mathcal{X}_U]$$

Remark 2. *We consider two unsafe-set formulations. (1) The unsafe set is a subset of the domain, $\mathcal{X}_U \subseteq \mathcal{X}$. This is a common way to formulate safety specifications: unsafe states are given explicitly and trajectories must avoid them while they remain in $\mathcal{X}$ Abate et al. (2021); Vertovec et al. (2025). (2) A safe set is prescribed and trajectories must remain within it Mathiesen et al. (2022); Xue (2026). This formulation is recovered by taking $\mathcal{X}$ as the safe set and setting $\mathcal{X}_U = \mathbb{R}^n \setminus \mathcal{X}$. More generally, some formulations include both a safe set and an unsafe set, which need not be complementary.*

The Lyapunov conditions in Theorem 1 and Corollary 1 address Problems 1 and 2. For the two safety-related problems, Theorem 2 establishes the basic barrier conditions, and Corollaries 2 and 3 give sufficient conditions for Problems 3 and 4, respectively.

C.2 CERTIFICATE CONDITIONS

For a scalar certificate function $W : \mathbb{R}^n \to \mathbb{R}$, continuously differentiable in the continuous-time case, define

$$\mathcal{D}_\pi W(x) := \begin{cases} \nabla W(x)^\top f_\pi(x), & \text{continuous time,} \\ W(F_\pi(x)) - W(x), & \text{discrete time.} \end{cases} \tag{4}$$

Lyapunov certificates.

Theorem 1 (Khalil & Grizzle (2002))**.** Let $\mathcal{N} \subseteq \mathbb{R}^n$ be a neighborhood of the origin, and let $V : \mathbb{R}^n \to \mathbb{R}$ be continuous, and continuously differentiable in the continuous-time case. Suppose that

$$V(0) = 0, \qquad V(x) > 0 \quad \text{for all } x \in \mathcal{N} \setminus \{0\}.$$

Then:

1. If $\mathcal{D}_\pi V(x) \leq 0$ for all $x \in \mathcal{N}$, then the origin is stable in the sense of Lyapunov on $\mathcal{N}$.
2. If $\mathcal{D}_\pi V(x) < 0$ for all $x \in \mathcal{N} \setminus \{0\}$, then the origin is asymptotically stable on $\mathcal{N}$.
3. If V is positive definite and radially unbounded on $\mathbb{R}^n$, and $\mathcal{D}_\pi V(x) < 0$ for all $x \neq 0$, then the origin is globally asymptotically stable.
4. Local exponential stability: If there exist $c_1, c_2 > 0$ and $c_3 > 0$ such that

   $$c_1\|x\|^2 \leq V(x) \leq c_2\|x\|^2, \qquad \mathcal{D}_\pi V(x) \leq -c_3 V(x), \qquad \forall x \in \mathcal{N},$$

   then the origin is exponentially stable. In discrete time, the condition $\mathcal{D}_\pi V(x) \leq -c_3 V(x)$ is replaced by $V(F_\pi(x)) \leq \rho\, V(x)$ for some $\rho \in (0, 1)$, which equivalently means $c_3 \in (0, 1]$ in the above form. Global exponential stability holds if these inequalities are satisfied on $\mathbb{R}^n$.

Corollary 1 (Region-of-attraction certificate)**.** Assume that the origin is an equilibrium of the closed-loop system. Let $\mathcal{N} \subseteq \mathbb{R}^n$ be a neighborhood of the origin, let $V : \mathbb{R}^n \to \mathbb{R}$ be continuous and, in the continuous-time case, continuously differentiable, and let $c > 0$. Define the sublevel set $\Omega_c := \{x \in \mathbb{R}^n : V(x) \leq c\}$. Suppose that:

1. $V(0) = 0$.
2. $V(x) > 0$ for all $x \in \Omega_c \setminus \{0\}$.
3. $\mathcal{D}_\pi V(x) < 0$ for all $x \in \Omega_c \setminus \{0\}$.
4. Ω_c is compact and satisfies $\mathcal{X}_I \subseteq \Omega_c \subseteq \mathcal{N}$.

Then $\mathcal{X}_I \subseteq \mathcal{R}_A(0)$, i.e., Ω_c is a region of attraction.

Barrier certificates. We consider the following barrier certificate conditions for continuous-time and discrete-time systems Prajna et al. (2007); Agrawal & Sreenath (2017); Ames et al. (2019).

Theorem 2. Let $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{X}_U \subseteq \mathcal{X}$(or $\mathcal{X}_U = \mathbb{R}^n \setminus \mathcal{X}$). Let $B : \mathbb{R}^n \to \mathbb{R}$ be continuous and, in the continuous-time case, continuously differentiable. Suppose that:

1. $B(x) < 0$ for all $x \in \mathcal{X}_U$;
2. **Continuous time:** for all $x \in \mathcal{X}$ with $B(x) = 0$, $\nabla B(x) \neq 0$ and $\mathcal{D}_\pi B(x) > 0$;
3. **Discrete time:** for all $x \in \mathcal{X}$ with $B(x) \geq 0$, $B(F_\pi(x)) \geq 0$.

Then there do not exist $x_0 \in \mathcal{X}$ with $B(x_0) \geq 0$ and $T \geq 0$ such that

$$\phi_\pi^{x_0}(T) \in \mathcal{X}_U \quad \text{and} \quad \phi_\pi^{x_0}(t) \in \mathcal{X}, \qquad \forall t \in [0, T).$$

Define the superlevel set $\mathcal{C}_B := \{x \in \mathbb{R}^n : B(x) \geq 0\}$.

Corollary 2 (Forward-invariant set synthesis)**.** If B satisfies the conditions of Theorem 2 and $\emptyset \neq \mathcal{C}_B \subseteq \mathcal{X}$, then $\mathcal{C}_B$ is forward invariant and $\mathcal{C}_B \cap \mathcal{X}_U = \varnothing$. Hence, $\mathcal{S} = \mathcal{C}_B$ is a solution to Problem 3.

Corollary 3 (Safety verification)**.** If B satisfies the conditions of Theorem 2 and $B(x) \geq 0$ for all $x \in \mathcal{X}_I$, then Problem 4 is verified.

Table 5: Prompt interfaces and evidence flow.

Agent	Message role	Injected by harness	LLM output
Proposal Agent	system + user	specification R, task profile, locked domain/sets, previous counterexample or diagnostics	JSON candidate $C^{(k)} = (W^{(k)}, \pi^{(k)})$
Lean Proof-Repair Agent	system + user	locked theorem, definitions, candidate hash, compiler diagnostics, Mathlib search results	JSON proof body, helper lemmas, or `cannot_complete`
Local LeanRepair	user	frozen proof, target source span, compiler context, verified declarations	`LeanLocalPatch` or `cannot_patch`

C.3 Verification Principles

This paper uses two kinds of verification tools.

1. SMT solvers (Z3, cvc5, Yices, and dReal). All of them perform counterexample search: the negation of the universal condition $\forall x.\, P(x)$ is encoded as a satisfiability problem over the reals. If the solver returns `unsat`, no counterexample exists and the original condition holds. Z3, cvc5, and Yices give exact verdicts on decidable arithmetic fragments, whereas dReal performs a δ-complete search for general nonlinear real arithmetic. A dReal δ-`sat` result is inconclusive with respect to the original formula: it means that a δ-weakened version of the problem is satisfiable, not that the original formula is.

2. The proof assistant (a.k.a. interactive theorem provers, ITP) Lean, which takes a fundamentally different approach. Rather than searching for counterexamples, Lean constructs a formal proof term for $\forall x.\, P(x)$ in dependent type theory. The condition is established once the proof term passes the kernel's type check.

The essential distinction lies in the underlying foundations: SMT solvers appeal to model theory and establish the non-existence of a model over the reals, whereas Lean appeals to proof theory and establishes the existence of a derivation in a formal system. Whichever tool is used, once all certificate conditions have been verified, the corresponding Lyapunov or barrier theorem implies the desired stability or safety property.

D Prompts Used in LLM2Cert

This appendix gives the template-level prompt interfaces used by LLM2Cert. Runtime values are injected by the harness at the marked placeholders. The complete serialized prompts, JSON schemas, and case traces are provided in the supplementary material. Table 5 summarizes the message roles and the evidence flow between the harness, the LLM, and the verification backends. Table 6 summarizes how the harness processes LLM outputs.

Model and decoding configuration. The model, reasoning-effort setting, output-length limit, and budgets used in the reported runs are listed in Appendix E.1. The reported runs use `gpt-5.6-sol` through the OpenRouter chat-completions API, with reasoning effort `high`. The maximum output length is 16,384 tokens. The reported runs cost $16.73 for the full pipeline, $15.40 for Lean-only(main experiments), $4.47 for SMT-only(main experiments), $5.69 for All-Lean (RQ2), $6.39 for the repair comparison (RQ3). Other agent attempts, including debugging, aborted runs, and earlier sweeps cost $227.10. The total is $275.77.

The prompts below are organized by agent and repair stage. The Proposal Agent uses one output protocol across tasks; the task profile determines whether $W^{(k)}$ is a Lyapunov certificate, a region-of-attraction certificate, or a barrier certificate. The Lean 4 Proof-Repair Agent operates on one locked obligation at a time and returns only proof bodies, helper lemmas, or a precise blocker.

Table 6: Harness processing of LLM outputs.

LLM output	Harness action	Result
Candidate JSON	format check, semantic check, compile to $A^{(k)}$	obligation list $\mathcal{M}(A^{(k)})$ and fast screening
Counterexample claim	exact rational replay	accepted as evidence only if replayed; otherwise diagnostic
Lean proof body	elaboration in pinned Lean/Mathlib, axiom audit	accepted proof, repair feedback, or unresolved trace
`cannot_complete`	check blocker and optional counterexample	continue proof search or return diagnostics to Proposal Agent

D.1 Proposal Agent

As described in Section 3.1, the Proposal Agent receives the typed specification R and proposes a candidate

$$C^{(k)} = (W^{(k)}, \pi^{(k)})$$

at iteration k. The candidate is first checked for format and semantic validity. Once accepted, it is compiled into the candidate representation $A^{(k)}$, which induces the obligation list $\mathcal{M}(A^{(k)})$.

The same output protocol is used across tasks, while the task profile specifies whether $W^{(k)}$ is interpreted as a Lyapunov certificate V, a region-of-attraction certificate, or a barrier certificate B.

D.1.1 Initial Proposal Prompt

Role:

You are a control theory expert specializing in nonlinear dynamical systems, certificate construction, and formal verification.

System information:

The system name, problem identifier, dynamics, time semantics, state variables, control variables, formal domain, screening box, initial set, unsafe set, equilibrium information, fixed parameters, control set, and controller observation pattern are inserted here.

Verification task:

The task profile is inserted here. For stability and region-of-attraction tasks, the profile specifies the required Lyapunov conditions on V and, when applicable, the level set $\Omega_c = \{x \mid V(x) \leq c\}$. For invariance and safety tasks, the profile specifies the required barrier conditions on B, including the initial-set, unsafe-set, boundary, or one-step conditions.

At iteration k, propose a candidate certificate and, when applicable, a feedback controller. The candidate must satisfy the supplied theorem conditions rather than a relaxed numerical surrogate.

Use continuous-time Lie-derivative semantics only for continuous-time systems. For discrete-time systems, use only the one-step closed-loop map F_{cl} and the corresponding change in the certificate.

If fixed parameters are present, construct one state-only certificate that works uniformly over the full parameter interval. Parameters may appear in the dynamics and in the construction explanation, but not as additional certificate variables.

Candidate construction:

Use only the declared state variables. Every controller assignment must respect the supplied observation pattern and control bounds. Prefer a constant, linear, or affine controller before using a nonlinear controller, and avoid unnecessarily large gains.

Use exact rational constants and the supported expression language. The certificate and controller must be returned as concise expressions rather than recursive AST nodes.

The construction summary is a construction note, not a certificate. It must identify the useful auxiliary quantities, give a factorized form of the Lie expression or one-step change, and state the explicit bounds needed by the downstream SMT and Lean verification stages.

Output format:

Return exactly one JSON object matching the task-specific output contract. Return the certificate, controller assignments when applicable, a construction summary, and the required assumptions or comparison constants.

Do not return Markdown, code fences, recursive AST nodes, unsupported function names, or any text outside the JSON object.

Transition to screening. After format and semantic checking, the candidate is compiled into $A^{(k)}$ and induces the finite obligation list $\mathcal{M}(A^{(k)})$. Fast screening checks these obligations on a finite sample-based schedule. A detected violation rejects the candidate and is returned to the Proposal Agent as structured evidence. Passing screening does not establish a theorem and does not remove any obligation from $\mathcal{M}(A^{(k)})$.

D.1.2 Proposal Repair Prompt

A rejected candidate is not repaired by changing the verification target. Only a confirmed and independently replayed counterexample authorizes a new candidate-generation turn. Solver timeouts and proof diagnostics remain inconclusive and are returned as diagnostic evidence.

Next turn:

The system message still contains the problem specification, task conditions, and output contract. Do not wait for these sections to be repeated. The following message contains only new evidence from the previous attempt.

Diagnose the failed condition and propose the next candidate. Change the expression structure when coefficient tuning cannot address the evidence. Preserve the continuous-time or discrete-time semantics of the task.

New evidence:

The candidate index, failed candidate, independently verified counterexample, rejected candidate history, and parser or semantic errors are inserted here.

Task-specific reminders:

For stability, a counterexample in the fixed claimed domain must be addressed by changing the certificate or controller; it may not be removed by introducing an unsupported level restriction.

For region-of-attraction verification, the level c may be changed, but the new candidate must still satisfy the locked containment and Lyapunov obligations.

For invariance and safety, preserve the original working domain, initial set, unsafe set, and barrier semantics.

Output format:

Return exactly one JSON object matching the original task-specific output contract. Return concise certificate and controller expressions, not recursive AST nodes or explanatory text outside the JSON object.

Transition to formal verification. When an obligation is selected for Direct Lean, or when SMT leaves the obligation unresolved, the framework constructs a locked Lean theorem from the fixed candidate and obligation. The Lean agent is allowed to generate proof terms, but not to change R, $C^{(k)}$, $A^{(k)}$, the theorem statement, or the obligation list.

D.2 Lean Proof-Repair Agent

The Lean Proof-Repair Agent operates on one obligation at a time. The framework compiles each response in the pinned Lean/Mathlib environment. The proof interface is proof-body-only: the theorem header and generated definitions remain fixed.

D.2.1 INITIAL LEAN PROOF PROMPT

Role:

Complete one fixed Lean 4/Mathlib verification condition.

Goal:

Return the complete term inserted after `:=` in `proof_body`. The theorem statement, generated definitions, dynamics, certificate, controller, and candidate hash are locked.

For a Lyapunov obligation, prove exactly the stated equilibrium, positivity, comparison, Lie-derivative, one-step decrease, radial-unboundedness, or exponential-decay condition.

For a barrier or safety obligation, prove exactly the locked boundary, Lie-derivative, initial-set, unsafe-set, or one-step condition. For an existential violation statement, begin from the actual proposition, usually with `rintro` $\langle x, hx \rangle$.

Fixed inputs:

The candidate hash, condition identifier and kind, locked Lean definitions, locked theorem statement, public framework lemmas, previous proof body, compiler diagnostics, and proof mode are inserted here.

Division of labour:

The agent does not need a local Lean workspace. The harness compiles the response against the pinned Lean 4 and Mathlib environment and returns exact diagnostics on later rounds.

Constraints:

Do not change the theorem statement, generated definitions, system, controller, certificate, or candidate hash.

Do not use `sorry`, `admit`, user-defined axioms, `unsafe`, imports, external assumptions, Markdown, or a natural language proof sketch. Do not declare a new top-level theorem inside `proof_body`.

Helper lemmas are optional. Each helper entry must contain one focused named declaration and is compiled independently. Use local `have` and `calc` steps for one-use facts.

Use the generated unfolding and closed-loop simplification lemmas. Use supplied smoothness hypotheses directly. Establish local sign, interval, trigonometric, and algebraic facts before invoking `linarith` or `nlinarith`.

Qualified `Real.*` names are prechecked for existence. This is an early name check, not a restriction to the `Real` namespace.

Acceptance condition:

The proof must compile, preserve the locked theorem hash, and pass the axiom audit. The accepted dependency set is

$$\{\texttt{propext}, \texttt{Classical.choice}, \texttt{Quot.sound}\}.$$

Output format:

Return exactly one JSON object with the fields `status`, `helper_lemmas`, `proof_body`, `blocking_reason`, and `counterexample`.

Return `candidate_proof` only with a concrete proof body. Return `cannot_complete` only with a precise blocker. A claimed counterexample must contain exact rational coordinates and is replayed independently by the harness.

D.2.2 COMPILER-GROUNDED LEAN REPAIR

A failed first proof does not trigger an unrestricted retry. The framework classifies the failure and returns the relevant evidence to the same locked proof context.

Output-format failure. If the response is not a valid structured object, the framework performs a format-only repair. This repair changes the JSON shape, not the mathematical proof.

Unknown declaration. Qualified `Real.*` names are checked before full elaboration. If a name is unknown, local Mathlib search results and Lean-verified replacement declarations are returned.

Syntax, elaboration, or type error. Lean diagnostics, source locations, and relevant goal states are passed back with the previous proof. The agent repairs the failing helper or main proof while preserving accepted parts.

Tactic failure or unsolved goal. The repair prompt asks the agent to expose intermediate sign, interval, and algebraic facts before retrying `linarith`, `nlinarith`, or other tactics.

Forbidden proof content. Responses containing `sorry`, `admit`, user-defined `axiom`, `unsafe`, or `import` are rejected.

Timeout or proof-budget exhaustion. The obligation remains unresolved. Its diagnostic trace is returned to the Proposal Agent; the locked theorem is not silently changed.

Axiom-audit failure. A compiled proof is rejected if `#print axioms` contains `sorryAx` or an axiom outside the accepted dependency set.

Rejected falsity claim. A `cannot_complete` response claiming falsity must include an exact rational counterexample. The harness replays it independently. A rejected counterexample does not trigger candidate regeneration and the agent is instructed to continue proving the theorem.

The implementation bounds these repairs by three format-only repairs, six ordinary Lean proof rounds, and three qualified-name repair rounds. In the optional contextual mode, a small number of line-local repairs may be requested before returning to the full proof-repair path.

D.2.3 LEAN PROOF FOLLOW-UP PROMPT

Next turn:

The system message still contains the Lean role and output contract. Helper lemmas remain one named declaration per entry and are compiled independently. Keep accepted helpers unchanged and edit only the failing helper or main proof.

New evidence:

The candidate hash, condition identifier and kind, locked definitions, locked theorem statement, previous proof body, compiler diagnostics, and repair mode are inserted here.

Continue from the previous proof. Fix every reported error while preserving the locked theorem and candidate. Do not return a proof sketch or change the verification target.

Output format:

Return exactly one JSON object: either `candidate_proof` with a concrete proof body or `cannot_complete` with the precise blocker.

D.2.4 LOCAL LEANREPAIR PROMPT

When contextual local repair is enabled, the framework can freeze the current proof and request a patch for one compiler-selected source span.

Local repair turn:

The initial proof is frozen. For this turn only, do not return the full `helper_lemmas`/`proof_body` document. Repair exactly the framework-selected source line and return a `LeanLocalPatch`.

Compiler-grounded request:

The base output hash, patch identifier, target kind and index, target line, old source text, source window, compiler context, and Lean-verified declarations are inserted here.

Constraints:

Return replacement code relative to the existing indentation. Preserve all unrelated proof text. The replacement may be a single Lean line or a local multi-line `by` proof, but it may not declare a new top-level lemma.

Do not reuse declarations reported as unknown by the compiler. A new qualified declaration may be used only when its exact declaration appears in the Lean-verified declaration list. Otherwise prove the required fact locally from accepted surrounding facts.

Do not use `sorry`, `admit`, `axiom`, `import`, `unsafe`, Markdown, or explanations outside the structured output. If the selected line cannot be repaired without changing another source line, return `cannot_patch`.

Output format:

Return exactly one JSON object. Echo the supplied patch identifier. Return `patch` with concrete `replacement_code`, or return `cannot_patch` with an empty replacement and a precise reason.

D.3 Example Input–Output Trace

To illustrate how the prompt interfaces are instantiated in practice, we show one trace from the safety instance (System 88). The trace is taken from the recorded local run rather than reconstructed from a manually written example.

The human user does not directly provide a separate message at each stage. Instead, the local harness converts the current specification and verification evidence into the `system` and `user` messages sent to the LLM.

1. Proposal input. The harness supplies the Proposal Agent with the system specification:

$$\dot{x}_1 = x_2 + u_0,$$
$$\dot{x}_2 = -\frac{3}{10}x_1 - \frac{1}{5}(1+\cos x_1)x_2 - \frac{1}{10}x_1^3 + u_1,$$

together with

$$\mathcal{X} = \mathbb{R}^2, \qquad \mathcal{X}_I = \{x : \|x\|_2 \le 3/5\}, \qquad \mathcal{U} = [-1,1]^2,$$

and the unsafe set

$$\mathcal{X}_U = \left\{x : x_1, x_2 \in [1,3],\ (x_1 - \tfrac{1}{2})^2 + (x_2 - \tfrac{1}{2})^2 \le 1\right\}.$$

The task profile identifies the problem as continuous-time safety verification and requires a state barrier, a bounded controller, initial-set containment, unsafe-set separation, and a strict boundary Lie-derivative condition.

2. First Proposal-Agent output. The first structured response was mathematically plausible but failed the local expression parser because the certificate was returned with an assignment:

```
"certificate":  "B = 1 - x1**2 - x2**2".
```

The parser expects the expression itself rather than the assignment `B =`. The harness therefore returned a format/semantic diagnostic to the Proposal Agent without constructing a candidate AST.

3. Proposal repair input and output. The next user message to the LLM contained only the new evidence:

```
The system message still has the problem, the locked task conditions,
and the output schema.  Below is only new evidence.

parse_or_semantic_errors:
candidate 0: parse error:
certificate: invalid expression
'B = 1 - x1**2 - x2**2'
```

The Proposal Agent then returned the corrected candidate

$$B(x) = 1 - x_1^2 - x_2^2, \qquad u_0(x) = -\frac{x_1}{1+x_1^2}, \qquad u_1(x) = -\frac{x_2}{1+x_2^2}.$$

Its construction summary stated that the initial disk satisfies $B \ge 16/25$, the unsafe set satisfies $B < 0$.

4. Routing output. The deterministic capability analysis routed the polynomial obligations to SMT. The transcendental strict Lie obligation was first exposed to the verification router, which returned

`route = smt, solver_order = [cvc5].`

The cvc5 query timed out, so the unresolved obligation was passed to the Lean 4 Proof-Repair Agent.

5. Initial Lean-proof input and output. The Lean prompt contained the locked theorem, generated definitions, candidate hash, obligation identifier, public framework lemmas, and the construction summary. The first proof response used the intended boundary identity, but cited the nonexistent declaration `neg_one_le_cos`. The framework returned the compiler error and a local Mathlib search result suggesting `Real.one_sub_sq_div_two_le_cos`.

The second proof response used the suggested theorem with the wrong application syntax. Lean reported that the theorem was already specialized and should not be applied as an ordinary function. The next repair replaced the application with a type-correct form:

```
simpa using
  (Real.one_sub_sq_div_two_le_cos (x := a))
```

The third proof response compiled successfully and produced

`#print axioms` = [`propext`, `Classical.choice`, `Quot.sound`].

The strict Lie obligation was therefore accepted by the Lean kernel.

6. Final harness result. The remaining obligations were discharged as follows:

Obligation	Backend	Result
Unsafe separation	Z3	UNSAT
Initial containment	Z3	UNSAT
Control admissibility	CVC5	UNSAT
Barrier condition	audited static lemma	accepted
Strict Lie condition	Lean 4	kernel proof accepted
Safety instantiation	audited generic theorem	accepted

The candidate was finally marked **verified**. This trace shows that the LLM proposes certificates and proof terms, while the local harness performs parsing, candidate freezing, routing, solver checks, Lean compilation, and axiom auditing.

E Experimental Details

E.1 Implementation Details

Hardware and software. The experiments were conducted on a MacBook Pro with a 12-core Apple M4 Pro and 24 GB of memory. The toolchain uses Lean 4.32.2 and Mathlib revision `905b95818eb3`, together with Z3 4.15.4, cvc5 1.3.3, Yices 2.7.0, and dReal 4.21.06.1. Z3, cvc5, and Yices run on the host, while dReal runs in a pinned `linux/amd64` Docker image with precision $\delta = 10^{-3}$ and solver seed 0.

Model and budgets. The Proposal Agent and Lean Proof-Repair Agent use `gpt-5.6-sol` with reasoning effort `high`; the Verification Router uses the same model. The maximum output length is 16,384 tokens. The end-to-end experiment uses at most $K_{\mathrm{cand}} = 5$ candidate rounds for each system under a shared $T_{\mathrm{run}} = 2400\,\mathrm{s}$ budget. Each compatible SMT solver has a $30\,\mathrm{s}$ timeout in Routed verification. For an obligation sent to Lean, the Proof-Repair Agent uses at most $K_{\mathrm{lean}} = 6$ proof rounds within $900\,\mathrm{s}$.

Algorithm 1 End-to-end certificate search under a shared deadline

Require: Original problem P, candidate limit K_{cand}, deadline T
$H \leftarrow \varnothing$
for $k = 1, \ldots, K_{\mathrm{cand}}$ **do**
  if the active-work budget is exhausted **then**
    return UNRESOLVED
  end if
  $A \leftarrow \textsc{GenerateAndParse}(P, H)$
  if no well-formed candidate is obtained **then**
    Add parsing diagnostics to H; **continue**
  end if
  Freeze A and its obligation manifest $M(A)$
  $e \leftarrow \textsc{FastCheck}(P, A)$
  if e is an independently validated violation **then**
    Add e to H; **continue**
  end if
  $v \leftarrow \textsc{VerifyFrozen}(P, A, M(A), T)$
  if all required obligations are verified **then**
    return $(A, \textsc{Verified})$
  end if
  Add unresolved diagnostics or validated counterexamples from v to H
end for
return UNRESOLVED

Fast screening. Fast screening uses 1,500 samples from the initial set, 1,500 samples from the unsafe set, and 2,000 samples from the analysis or boundary region, depending on the task. At most five violating samples are returned to the Proposal Agent. Screening is used only to reject candidate certificates or provide feedback; passing the screening step never discharges a formal verification obligation.

Search and verification procedure. Algorithms 1 and 2 give the implementation used in the experiments. Candidate certificates and their verification obligations are frozen before formal verification. The candidate, hypotheses, and theorem statements are not changed during the subsequent SMT or Lean verification process.

E.2 Additional Results for the Main Experiment

Table 7 complements Table 2 by grouping the end-to-end results by task and reporting the recorded computational cost of the successful runs. The full pipeline obtains verified certificates in all four task categories. Among successful runs, ROA has the largest recorded token cost per system, and forward-invariance has the largest wall-clock time.

Table 7: End-to-end results grouped by task.

Task	Verified	Success	Tokens/sys. (k)	Time/sys. (s)
Stability	21/26	80.8%	16.00	613.57
ROA	8/12	66.7%	19.63	427.48
Forward-invariance	7/13	53.8%	16.07	704.73
Safety	45/54	83.3%	13.95	377.58
Overall	81/105	77.1%	15.22	471.96

Model usage. Across the 81 successful systems, the end-to-end experiment records 1.23M model tokens, corresponding to 15.22k tokens per verified system. Across all 105 systems, the recorded total is 3.48M tokens. Usage information is available for 488 of the 707 recorded model calls, so these values represent recorded rather than complete billed usage. The Lean Proof-Repair Agent has the largest recorded usage per call, consistent with the higher cost of the obligations that reach Lean verification.

Algorithm 2 Verification of a frozen obligation manifest

Require: Frozen candidate A, manifest M, strategy s
Check candidate and manifest hashes
for all active obligations O_i **do**
  Preserve the original definitions, hypotheses, and target of O_i
  Select the backend according to s and solver compatibility
  if O_i is assigned to SMT **then**
    Run compatible solvers on the negation of O_i
    if a definitive UNSAT verdict is obtained **then**
      Mark O_i discharged; **continue**
    else if a reported witness validates a violation of O_i **then**
      return REJECTED
    else if $s =$ ALL-SMT **then**
      Mark O_i unresolved; **continue**
    end if
  end if
  if $s =$ ALL-SMT and no compatible SMT encoding exists **then**
    Mark O_i unresolved; **continue**
  end if
  Request a proof of the unchanged Lean theorem
  Compile the proof and helper lemmas
  Repair using the selected feedback while the proof budget remains
  Mark O_i discharged only after successful compilation and proof audit
end for

Table 8: Mean recorded model usage per end-to-end call.

Agent	Calls	Input (k)	Reasoning (k)	Output (k)
Proposal synthesis	119	2.24	3.66	0.44
Verification router	170	1.90	0.08	0.08
Lean proof agent	199	7.35	3.17	1.39

E.3 ADDITIONAL ANALYSIS FOR RQ1 AND RQ2

Task-wise results. Table 9 gives the same fixed-candidate comparison as Table 3, but groups the 81 certificates according to task. Routed gives SMT 30 s and sends the remaining obligations to Lean.

Table 9: Task-wise verification on the 81 fixed certificates.

Task	All-SMT	All-Lean	Routed	Lean/active	Routed time
Stability	13/21	10/21	15/21	9/93	30.36
ROA	6/8	2/8	6/8	3/41	1.97
Forward-invariance	4/7	3/7	5/7	3/24	124.73
Safety	36/45	27/45	40/45	11/138	70.74
Total	59/81	42/81	66/81	26/296	59.40

Routed verifies 66 of the 81 certificates, and it exceeds both single backends in every task except ROA, where it matches All-SMT at 6/8. The widest gap against All-Lean is safety, 40/45 against 27/45. Routed verification time also differs across tasks, and is highest for forward-invariance, motivating the obligation-level analysis below.

Verification cost. Table 10 complements the runtime comparison in the main text with recorded model-token usage. All three strategies use the same 81 frozen certificates and unchanged verification obligations. Tokens/sys. averages all 81 certificates. Time/sys. averages the certificates that the strategy verifies, as in Table 3.

Table 10: Cost of fixed-candidate verification.

Strategy	Verified	Tokens/sys. (k)	Time/sys. (s)	Lean obl.
ALL-SMT	59/81	0	3.68	0
ALL-LEAN	42/81	15.65	653.53	296
ROUTED	66/81	2.74	59.40	26

Compared with All-Lean, Routed reduces the recorded verification-token cost by 82.5%, from 1.27M to 0.22M, and the mean verified time by 90.9%, from 653.53s to 59.40s, while verifying 66 of the 81 fixed certificates. The reduction comes mainly from avoiding Lean proof generation for obligations already discharged by SMT.

Verification paths. Among the 296 active obligations, SMT discharges 270 (91.2%) directly by `UNSAT` within 30 s, while 26 ultimately require Lean. Among these 26, eight follow a recorded δ-`SAT` result, 15 follow an SMT timeout, and three have no SMT encoding. All three are radial-unboundedness obligations. A δ-`SAT` result alone is not treated as a counterexample; a reported witness must independently violate the original verification condition.

Condition-wise verification. Table 11 groups the same 296 obligations according to their verification condition. The time column sums per-obligation Routed time. Continuous decrease/exponential-decay and discrete-inductiveness conditions account for 20 of the 26 Lean-routed obligations and 73.3% of this summed time. Equilibrium, positivity/quadratic-bound, set-containment, and control-admissibility obligations are all discharged without Lean in this cohort. These results suggest that future routing can further use the structure of individual obligations when assigning verification budgets.

Table 11: Routed verification by obligation type.

Obligation	Active	Lean	Time (s)
Equilibrium/value	58	0	101.86
Positivity/quadratic bounds	35	0	63.06
Set containment	58	0	68.32
Unsafe separation	52	3	2563.01
Control admissibility	9	0	24.32
Continuous decrease/exp. decay	73	17	12058.77
Discrete inductiveness	8	3	817.92
Radial unboundedness	3	3	1878.51
Total	296	26	17575.77

E.4 ADDITIONAL ANALYSIS FOR RQ3

The main text reports repair success, repair rounds, and timeout-charged time for the 19-system, 26-obligation repair set. Here, we additionally report the model-token cost of the three feedback strategies. Each arm re-called the model from the same failed first attempts. Usage was returned for 53 of 95 Generic Retry calls, 43 of 71 Raw Diagnostics calls, and 45 of 66 Structured Repair calls.

Table 12: Recorded token usage for RQ3 proof repair.

Strategy	Calls	Input(k)	Reasoning(k)	Output(k)	Total/obl.(k)
GENERIC RETRY	95	534.64	129.07	68.02	28.14
RAW DIAGNOSTICS	71	445.62	58.75	74.74	22.27
STRUCTURED REPAIR	66	534.71	70.92	90.59	26.78

Structured Repair requires 66 post-failure calls, compared with 71 for Raw Diagnostics and 95 for Generic Retry. Its recorded token use is 4.9% below Generic Retry and 20.2% above Raw Diagnostics, because each structured call carries a longer prompt. The token ordering is therefore separate from the repair-time ordering in Table 4, where Structured Repair is the fastest of the three.

Of the 26 repair-set obligations, 13 are continuous Lie-decrease or exponential-decay conditions, two are discrete-inductiveness conditions, six are positivity or quadratic-bound conditions, two concern radial unboundedness, and one each concerns equilibrium, unsafe separation, and set containment. Thus, 15 of the 26 repair-set obligations arise from continuous decrease/exponential decay or discrete inductiveness, consistent with the condition-wise results in Table 11.

E.5 Analysis of Unresolved Systems

The full end-to-end experiment leaves 24 of the 105 systems unresolved. All 24 terminate under the shared 2400 s budget, and none reaches the five-candidate limit. Table 13 reports the last recorded event of these runs. Most unresolved runs terminate after entering Lean verification rather than before a candidate is parsed. These labels describe the last recorded event and are not treated as unique mathematical causes of failure.

Six runs never obtain a parsed candidate. The other 18 enter Lean. Three of them also record a replay-confirmed violation on a separate obligation: DCB-EXT-P2-001, DCB-OURS-P2-002, and DCB-OURS-P2-003. Their run status remains the shared deadline, except DCB-OURS-P2-002, which stops on a helper-module timeout. DCB-OURS-P4-002 stops on an API rate limit during a Lean call. DCB-LIT-P4-001 receives `cannot_complete` on its Lie condition, without a checker-confirmed counterexample. These cases are consistent with the condition-wise results above, where the remaining verification cost is concentrated in the harder dynamical obligations.

Table 13: Last recorded event for the 24 unresolved runs.

Last recorded event	Systems
Lean call blocked by the shared deadline	10
No parsed candidate before the shared deadline	6
Invalid Lean proof output	3
Lean helper-module timeout	2
Lean compiler timeout	1
API rate limit during a Lean call	1
Lean agent returned `cannot_complete`	1
Total	24

F DynaCertBench: Benchmark Details

F.1 Benchmark Collection and Curation

Candidate benchmark records. We collected 233 candidate records from fifteen representative source papers and BarrierBench (Taheri et al., 2025). Specifically, 133 records were extracted from the fifteen papers and 100 records were obtained from BarrierBench. Each record captures a concrete experimental configuration, comprising the plant dynamics along with its assigned verification domain, controller, and solver settings. Because multiple records often instantiate the same underlying system under different test conditions, we systematically deduplicate and curate them into an independent core benchmark suite.

Curation rules. To build a non-redundant and structurally rigorous benchmark, we identify each instance by its intrinsic plant dynamics and control specifications. Implementation-level variations—such as solver hyperparameters, neural architectures, or scaled verification domains—are consolidated under their corresponding systems. Following the operational criteria summarized in Table 14, we apply five mutually exclusive rules: (1) *Rule 1 (Exact duplicates)*: merge records with identical dynamics and specifications across papers, retaining only the canonical instance; (2) *Rule 2 (Multiple configurations)*: consolidate records sharing identical vector fields under varying domains or hyperparameters into a single representative benchmark; (3) *Rule 3 (Family variants)*: retain one canonical representative for parametric sweeps or closely related models within the same system family; (4) *Rule 4 (Structural redundancy and degradation)*: exclude systems where nonlinearities are canceled by feedback linearization, dimensions are artificially padded with passive/decoupled coordinates, or dynamics reduce to simple linear contractions; and (5) *Rule 5 (Out-of-scope mod-*

No.	Curation rule	Operational criterion	Treatment	Source papers	BarrierBench	Total
1	Exact duplicate benchmark instance	Dynamics, task, sets, controller, and constraints are equivalent; only provenance, method, certificate architecture, or result differs.	Retain one canonical instance and merge redundant provenance records.	24	0	24
2	Multiple configurations of the same dynamics	The dynamics are identical, but the domain, task specification, controller, or solver setting differs.	Retain one canonical configuration and discard redundant variants.	22	1	23
3	Closely related variants in the same system family	The basic plant and interaction structure are unchanged; differences mainly concern parameters, coefficients, or minor local modifications.	Retain one representative system from the family.	31	13	44
4	Simplified or structurally redundant dynamics	Feedback cancels nonlinear terms, or additional states are linear, passive, decoupled, or already represented by a simpler system.	Exclude from the independent core benchmark suite.	0	54	54
5	Outside the current core experimental scope	The record is a theoretical illustration, counterexample, hybrid/abstraction model, or simulation-only case.	Exclude from the continuous formal verification track.	8	0	8
Not counted as independent core systems				**85**	**68**	**153**
Retained core systems				**48**	**32**	**80**
Candidate records				**133**	**100**	**233**

Table 14: Curation of the 233 candidate records into independent core benchmark systems. Each non-retained record is mapped to exactly one primary curation rule in the order shown.

No.	Source	Assigned task(s)	Reported records	Literature-derived	Reported $n \geq 5$	Retained $n \geq 5$
1	Abate et al. (2021)	P1 (8), P4 (3)	17	11	2	1
2	Edwards et al. (2024)	P1 (3), P2 (1), P4 (2)	21	6	1	0
3	Zhao et al. (2020)	P4 (1)	5	1	0	0
4	Liu et al. (2024)	P2 (2)	13	2	9	1
5	Sogokon et al. (2018)	P4 (2)	2	2	0	0
6	Wu et al. (2024)	P4 (3)	22	3	0	0
7	Zou et al. (2025)	P1 (8)	12	8	5	4
8	Zhou et al. (2022)	P2 (2)	3	2	0	0
9	Liu et al. (2015)	P4 (2)	8	2	0	0
10	Liu et al. (2023)	P2 (2)	14	2	0	0
11	Meng et al. (2024)	P2 (1)	6	1	2	0
12	Samanipour & Poonawala (2026)	P3 (3)	3	3	0	0
13	Vaisi et al. (2025)	P3 (2)	2	2	0	0
14	Liu & Fitzsimmons (2026)	P1 (1)	1	1	0	0
15	Wu et al. (2023)	P3 (1)	4	1	1	1
16	Khalil & Grizzle (2002) (Book)	P1 (1)	–	1	–	0
Subtotal: 15 source papers and one book			**133**	**48**	**20**	**7**
17	BarrierBench Taheri et al. (2025)	P4 (32)	100	32	9	0
Total			**233**	**80**	**29**	**7**

Table 15: Source contributions to the final Literature-derived tier. Reported counts refer to the 233-record candidate pool, which excludes the additional textbook example. The retained counts and task assignments refer to the 80 Literature-derived instances. P1–P4 denote stability certification, region-of-attraction verification, forward-invariant-set synthesis, and safety verification, respectively.

els): remove hybrid-reset systems, theoretical counterexamples, and simulation-only cases lacking formal verification specifications.

Curation results and high-dimensional scarcity. The curation process accounts for all 153 excluded records: 24 exact duplicates, 23 alternative configurations, 44 parametric family variants, 54 structurally simplified systems, and 8 out-of-scope entries. Consequently, the Literature-derived tier contains **80 instances**: 47 from the source papers, one from the textbook, and 32 from BarrierBench, as summarized in Table 15.

Crucially, this curation process highlights the severe scarcity of genuinely high-dimensional systems ($n \geq 5$) in the existing literature. While 29 candidate records nominally feature $n \geq 5$, only **7** survive after filtering: the 9 high-dimensional records from Liu et al. (2024) collapse to a single canonical system under Rule 2, while all 9 high-dimensional instances in BarrierBench are excluded under Rule 4 due to feedback linearization, linear-chain decoupling, or passive state padding.

Motivation for dataset extension. This acute shortage of coupled high-dimensional benchmarks motivates the three-tier taxonomy of our dataset: (1) *Literature-derived*: the curated core of 80

non-redundant baseline systems; (2) *Extended*: containing 11 instances that extend source systems through changes to state dimension, verification domains, or control constraints; and (3) *New*, containing 14 constructed systems with coupled, non-polynomial dynamics in five to ten dimensions. The additional tiers broaden the structural and verification settings represented in the benchmark.

Counting both fixed-dimension systems with $n \geq 5$ and dimension-parameterized system families as high-dimensional benchmark instances, the Extended and New tiers contribute four and fourteen additional instances, respectively. The total number increases from 7 to 25, while their share rises from 8.75% of the Literature-derived tier to 23.81% of the full benchmark, an increase of 15.06 percentage points.

F.2 Experiments details in main text

We report each method's native numerical semantics. FOSSIL 2.0 uses neural CEGIS and SMT to refute the strict certificate conditions; its released dReal configuration has precision $\delta_{\mathrm{F}} = 10^{-5}$, while the paper's Lyapunov example uses a Torus domain excluding $B_{0.01}(0)$ (the optional barrier belt affects training only) (Edwards et al., 2024). The RL method sets the dReal $\epsilon_{\mathrm{R}} = e^{-3}$ and the precision/margin to $\delta_{\mathrm{R}} = e^{-12}$, and checks $V(x) > \delta_{\mathrm{R}}$ and $L_f V(x) < -\delta_{\mathrm{R}}$ only on $D \setminus B_{\epsilon_{\mathrm{R}}}(0)$ (Zou et al., 2025). SynNLF learns a polynomial neural Lyapunov candidate and applies a floating-point SOS/LMI feasibility check through MOSEK. Also, its theorem introduces a slack $\varepsilon > 0$, while the released validator uses a derivative offset $\tau_{\mathrm{S}} = 5 \times 10^{-4}$ and a default upper bound $\beta_{\mathrm{S}} = 10^{6}$ (Zhao et al., 2025). We treat dReal δ-`sat` as well as non-definitive numerical SOS outcomes, as inconclusive and do not transfer any method-specific precision, domain exclusion, or tolerance to the SMT+Lean verification.

F.3 Problem 1 : Stability certification

System 1 (Exercise 4.30, Khalil & Grizzle (2002)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2}\tan\left(\frac{\pi x_1}{2} + \frac{\pi}{4}\right) + x_2 + \frac{1}{2} \\ x_1 - \frac{1}{2}\tan\left(\frac{\pi x_2}{2} + \frac{\pi}{4}\right) + \frac{1}{2} \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.01^2 \leq x_1^2 + x_2^2 \leq 0.25^2\}$.
- Original verified type: ϵ-asymptotic stability.
- Verified method in the original paper: analytical Lyapunov analysis.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 0.25^2\}$.
- Our verified type: local asymptotic stability.

System 2 (Table 1 row 1 (NonPoly0), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + x_1 x_2 \\ -x_2 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.01^2 \leq x_1^2 + x_2^2 \leq 1^2\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1^2\}$.
- Our verified type: local asymptotic stability.

System 3 (Table 1 row 2 (NonPoly1), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + 2x_1^2 x_2 \\ -x_2 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1, x_2 \geq 0,\ 0 < x_1^2 + x_2^2 \leq 10^2\}$.
- Original verified type: local asymptotic stability.
- Verified method in the original paper: Z3.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1, x_2 \geq 0,\ x_1^2 + x_2^2 \leq 10^2\}$.
- Our verified type: local asymptotic stability.

System 4 (Table 1 row 3 (NonPoly2), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -x_1 \\ -2x_2 + 0.1x_1x_2^2 + x_3 \\ -x_3 - 1.5x_2 \end{pmatrix}.$$

- $x^\star = (0, 0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1, x_2, x_3 \geq 0,\ 0 < x_1^2 + x_2^2 + x_3^2 \leq 10^2\}$.
- Original verified type: local asymptotic stability.
- Verified method in the original paper: Z3.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1, x_2, x_3 \geq 0,\ x_1^2 + x_2^2 + x_3^2 \leq 10^2\}$.
- Our verified type: local asymptotic stability.

System 5 (Table 1 row 4 (NonPoly3), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -3x_1 - 0.1x_1x_2^3 \\ -x_2 + x_3 \\ -x_3 \end{pmatrix}.$$

- $x^\star = (0, 0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1, x_2, x_3 \geq 0,\ 0 < x_1^2 + x_2^2 + x_3^2 \leq 10^2\}$.
- Original verified type: local asymptotic stability.
- Verified method in the original paper: Z3.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1, x_2, x_3 \geq 0,\ x_1^2 + x_2^2 + x_3^2 \leq 10^2\}$.
- Our verified type: local asymptotic stability.

System 6 (Table 1 row 5 (Poly1), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -x_1^3 - x_1x_3^2 \\ -x_2 - x_1^2x_2 \\ -x_3 + 3x_1^2x_3 - 3x_3 \end{pmatrix}.$$

- $x^\star = (0, 0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid 0.1^2 \leq x_1^2 + x_2^2 + x_3^2 \leq 10^2\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 \leq 10^2\}$.
- Our verified type: local asymptotic stability.

System 7 (Table 1 row 6 (Poly2), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1^3 + x_2 \\ -x_1 - x_2 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.01^2 \leq x_1^2 + x_2^2 \leq 10^2\}$.

- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: Z3.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 10^2\}$.
- Our verified type: local asymptotic stability.

System 8 (Table 1 row 7 (Poly3), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1^3 - x_2^2 \\ x_1 x_2 - x_2^3 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.1^2 \leq x_1^2 + x_2^2 \leq 10^2\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 10^2\}$.
- Our verified type: local asymptotic stability.

System 9 (Table 1 row 8 (Poly4), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 - 1.5x_1^2 x_2^3 \\ -x_2^3 + 0.5x_1^3 x_2^2 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.1^2 \leq x_1^2 + x_2^2 \leq 10^2\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 10^2\}$.
- Our verified type: local asymptotic stability.

System 10 (Table 4 #2 (SineModel), Edwards et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 + u_1 \\ u_2 + \dfrac{mgL\sin(x_1) - bx_2}{mL^2} \end{pmatrix},$$

where $m = 0.15$, $g = 9.81$, $L = 0.5$, and $b = 0.1$.

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.1^2 \leq x_1^2 + x_2^2 \leq 1^2\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1^2\}$.
- Our verified type: local asymptotic stability.

System 11 (Table 4 #14 (DTAhmadi2), Edwards et al. (2024)).

The discrete-time transition map is:

$$\mathbf{x}^+ = F(\mathbf{x}) = \begin{pmatrix} 0.687x_1 + 0.558x_2 - 0.0001x_1x_2 \\ -0.292x_1 + 0.773x_2 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.1^2 \leq x_1^2 + x_2^2 \leq 1^2\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1^2\}$.

– Our verified type: local asymptotic stability.

System 12 (Table 4 #15 (DTAhmadi), Edwards et al. (2024)).

The discrete-time transition map is:

$$\mathbf{x}^+ = F(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} 0.3x_1 + u_1 \\ -x_1 + 0.5x_2 + \dfrac{7}{18}x_1^2 + u_2 \end{pmatrix}.$$

– $x^\star = (0, 0)$.
– Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.1^2 \le x_1^2 + x_2^2 \le 1^2\}$.
– Original verified type: local ϵ-asymptotic stability.
– Verified method in the original paper: dReal.
– Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \le 1^2\}$.
– Our verified type: local asymptotic stability.

System 13 (Table 1 row 4, Zou et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -3x_1^3 + 3x_1x_3 - 9x_1 \\ -x_1^3 - 5x_2 + 5x_3^3 \\ -9x_3^3 \end{pmatrix}.$$

– $x^\star = (0, 0, 0)$.
– Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid e^{-6} \le x_1^2 + x_2^2 + x_3^2,\ |x_i| \le 1,\ i = 1, 2, 3\}$.
– Original verified type: local ϵ-asymptotic stability.
– Verified method in the original paper: dReal.
– Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid |x_i| \le 1,\ i = 1, 2, 3\}$.
– Our verified type: local asymptotic stability.

System 14 (Table 1 row 5, Zou et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\sin(x_1)\cos(x_1) - x_2 - \sin(x_3)\cos(x_3) \\ x_2 - x_3 \end{pmatrix}.$$

– $x^\star = (0, 0, 0)$.
– Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid e^{-6} \le x_1^2 + x_2^2 + x_3^2,\ |x_i| \le 1.5,\ i = 1, 2, 3\}$.
– Original verified type: local ϵ-asymptotic stability.
– Verified method in the original paper: dReal.
– Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid |x_i| \le 1.5,\ i = 1, 2, 3\}$.
– Our verified type: local asymptotic stability.

System 15 (Table 1 row 6, Zou et al. (2025)).

The source system is expressed in the center-of-inertia coordinates $(\delta_1, \delta_2, \omega_1, \omega_2)$, which satisfy $\delta_1 + \delta_2 = 0$. We therefore remove the redundant common-angle coordinate and define $q := \delta_1 - \delta_2, \delta_1 = \frac{q}{2}, \delta_2 = -\frac{q}{2}$. Let $\mathbf{x} = (q, \omega_1, \omega_2) \in \mathbb{R}^3$. The reduced vector field is

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{q} \\ \dot{\omega}_1 \\ \dot{\omega}_2 \end{pmatrix} = \begin{pmatrix} \omega_1 - \omega_2 \\ \frac{1}{2}\big(-1 - 2\omega_1 - \sin(q) + \cos(q)\big) \\ \frac{1}{2}\big(-1 - 2\omega_2 + \sin(q) + \cos(q)\big) \end{pmatrix}.$$

– $x^\star = (0, 0, 0)$.
– Original $\mathcal{N} = \left\{(q, \omega_1, \omega_2) \in \mathbb{R}^3 \;\middle|\; e^{-6} \le \frac{1}{2}q^2 + \omega_1^2 + \omega_2^2,\ |q| \le 1.5,\ |\omega_i| \le 2,\ i = 1, 2\right\}$.
– Original verified type: local ϵ-asymptotic stability.

- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \left\{(q, \omega_1, \omega_2) \in \mathbb{R}^3 \;\middle|\; |q| \le 1.5,\ |\omega_i| \le 2,\ i = 1, 2\right\}$.
- Our target: local asymptotic stability of the origin.

System 16 (Table 1 row 7, Zou et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} -x_1 + 0.5x_2 - 0.1x_5^2 \\ -0.5x_1 - x_2 \\ -x_3 + 0.5x_4 - 0.1x_1^2 \\ -0.5x_3 - x_4 \\ -x_5 + 0.5x_6 \\ -0.5x_5 - x_6 + 0.1x_2^2 \end{pmatrix}.$$

- $x^\star = (0, 0, 0, 0, 0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^6 \mid e^{-6} \le \sum_{i=1}^6 x_i^2,\ |x_i| \le 1,\ i = 1, \ldots, 6\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^6 \mid |x_i| \le 1,\ i = 1, \ldots, 6\}$.
- Our verified type: local asymptotic stability.

System 17 (Table 1 row 8, Zou et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{3}{2}x_6x_4 - 2x_1 - \frac{5}{2}x_2 - \frac{1}{2}x_4\sin(x_2)\cos(x_4) \\ x_4 \\ \frac{3}{2}x_6x_2 - 2x_3 - 10x_4 + \frac{1}{2}x_2\sin(x_2)\cos(x_4) \\ x_6 \\ -5x_5 - \frac{4}{5}x_6 \end{pmatrix}.$$

- $x^\star = (0, 0, 0, 0, 0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^6 \mid e^{-6} \le \sum_{i=1}^6 x_i^2,\ |x_i| \le 3,\ i = 1, \ldots, 6\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal and LaSalle's invariance principle.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^6 \mid |x_i| \le 3,\ i = 1, \ldots, 6\}$.
- Our verified type: local asymptotic stability.

System 18 (Table 1 row 9, Zou et al. (2025)).

The source system is expressed in the center-of-inertia coordinates $(\delta_1, \delta_2, \delta_3, \omega_1, \omega_2, \omega_3)$, which satisfy $\delta_1 + \delta_2 + \delta_3 = 0$. We therefore remove the redundant common-angle coordinate and define $\delta_3 := -(\delta_1 + \delta_2)$ and $\bar{\omega} := \frac{1}{3}(\omega_1 + \omega_2 + \omega_3)$. Let $\mathbf{x} = (\delta_1, \delta_2, \omega_1, \omega_2, \omega_3) \in \mathbb{R}^5$. The reduced vector field is

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\delta}_1 \\ \dot{\delta}_2 \\ \dot{\omega}_1 \\ \dot{\omega}_2 \\ \dot{\omega}_3 \end{pmatrix} = \begin{pmatrix} \omega_1 - \bar{\omega} \\ \omega_2 - \bar{\omega} \\ -\omega_1 - \frac{1}{2}\sum_{j=1}^3 \sin(\delta_1 - \delta_j) \\ -\omega_2 - \frac{1}{2}\sum_{j=1}^3 \sin(\delta_2 - \delta_j) \\ -\omega_3 - \frac{1}{2}\sum_{j=1}^3 \sin(\delta_3 - \delta_j) \end{pmatrix}.$$

- $x^\star = (0, 0, 0, 0, 0)$.
- Original $\mathcal{N} = \{(\delta_1, \delta_2, \omega_1, \omega_2, \omega_3) \in \mathbb{R}^5 \mid e^{-6} \le \delta_1^2 + \delta_2^2 + (\delta_1 + \delta_2)^2 + \sum_{i=1}^3 \omega_i^2,\ |\delta_1| \le 0.75,\ |\delta_2| \le 0.75,\ |\delta_1 + \delta_2| \le 0.75,\ |\omega_i| \le 1.2,\ i = 1, 2, 3\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal and LaSalle's invariance principle.

- Our $\mathcal{N} = \{(\delta_1, \delta_2, \omega_1, \omega_2, \omega_3) \in \mathbb{R}^5 \mid |\delta_1| \le 0.75,\ |\delta_2| \le 0.75,\ |\delta_1 + \delta_2| \le 0.75,\ |\omega_i| \le 1.2,\ i = 1, 2, 3\}$.
- Our target: local asymptotic stability of the origin.

Remark 3. *The original system is stated in coordinates with $\delta_1 + \delta_2 + \delta_3 = 0$. We substitute $\delta_3 := -(\delta_1 + \delta_2)$ and impose $|\delta_1 + \delta_2| \le 0.75$ to eliminate the equilibrium line $\{(c, c, c, 0, 0, 0)\}$ present in the six-dimensional formulation. This reduction makes the single-point stability problem well-defined.*

System 19 (Table 1 row 10, Zou et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{pmatrix} = \begin{pmatrix} -x_1 + 0.5x_2 - 0.1x_5^2 \\ -0.5x_1 - x_2 + 0.1x_8 \\ -x_3 + 0.5x_4 - 0.1x_1^2 \\ -0.5x_3 - x_4 \\ -x_5 + 0.5x_6 \\ -0.5x_5 - x_6 + 0.1x_2^2 \\ x_8 \\ -\sin(x_7)\cos(x_7) - x_8 - \sin(x_9)\cos(x_9) - 0.1x_2 \\ x_8 - x_9 \end{pmatrix}.$$

- $x^\star = (0, 0, 0, 0, 0, 0, 0, 0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^9 \mid e^{-6} \le \sum_{i=1}^9 x_i^2,\ |x_i| \le 1.5,\ i = 1, \ldots, 9\}$.
- Original verified type: local ϵ-asymptotic stability.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^9 \mid |x_i| \le 1.5,\ i = 1, \ldots, 9\}$.
- Our verified type: local asymptotic stability.

System 20 (Appendix Table 6, Zou et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -8x_1x_2^2 - 10x_2^4 \\ -8x_2^3 + 3x_2^2 - 8x_2 \\ -x_3 \end{pmatrix}.$$

- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid e^{-6} \le x_1^2 + x_2^2 + x_3^2,\ |x_i| \le 1,\ i = 1, 2, 3\}$.
- Original verified type: local ϵ-stability
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid |x_i| \le 1,\ i = 1, 2, 3\}$.
- Our verified type: local stability.

System 21 (Main theorem, Liu & Fitzsimmons (2026)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 4x^3 - x^2y - 6xy^2 - y^3 \\ x^3 + 4x^2y + xy^2 - 6y^3 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \mathbb{R}^2$.
- Original verified type: global asymptotic stability.
- Verified method in the original paper: human verfied by product.
- Our $\mathcal{N} = \mathbb{R}^2$.
- Our verified type: global asymptotic stability.

F.4 PROBLEM 2 : REGION-OF-ATTRACTION VERIFICATION

System 22 (Section 5.3 and Appendix C.3.3, Zhou et al. (2022)).

The closed-loop vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\theta} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} \omega \\ \dfrac{mg\ell \sin(\theta) + u - 0.1\omega}{m\ell^2} \end{pmatrix}.$$

$$u = 20 \tanh(-23.286329\theta - 5.27055\omega),$$

where $m = 0.15$ and $\ell = 0.5$.

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.4^2 \leq x_1^2 + x_2^2 \leq 4^2\}$.
- Original verified type: ϵ-ROA.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 4^2\}$.
- Our verified type: ROA verification.

System 23 (Section 5.1 and Appendix C.3.1, Zhou et al. (2022)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_2 \\ x_1 + (x_1^2 - 1)x_2 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.2^2 \leq x_1^2 + x_2^2 \leq 1.2^2\}$.
- Original verified type: ϵ-ROA.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1.2^2\}$.
- Our verified type: ROA verification.

System 24 (Example 3.1, Liu et al. (2023)).

The vector flow field is:

$$\dot{\mathbf{x}} = (\dot{x}) = \left(-x + x^3\right).$$

- $x^\star = 0$.
- Original $\mathcal{N} = (-1, 1)$.
- Original verified type: analytical ROA construction.
- Verified method in the original paper: analytical Zubov construction.
- Our $\mathcal{N} = (-1, 1)$.
- Our verified type: ROA verification.

System 25 (Example 6.2 and Table II, Liu et al. (2023)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -2x_1 + \frac{1}{3}x_1^3 - x_2 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = [-6, 6]^2$.
- Original verified type: ROA verification.
- Verified method in the original paper: dReal with Jacobian-based local verification.
- Our $\mathcal{N} = [-6, 6]^2$.
- Our verified type: ROA verification.

System 26 (Table 4 #3 (LorenzSystem), Edwards et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -10(x_1 - x_2) + u_1 \\ 28x_1 - x_2 - x_1 x_3 + u_2 \\ x_1 x_2 - \dfrac{8}{3} x_3 + u_3 \end{pmatrix}.$$

- $x^\star = (0, 0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid 0.01^2 \le x_1^2 + x_2^2 + x_3^2 \le 2^2\}$.
- $\mathcal{X}_I = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 \le 0.3^2\}$.
- $u_1, u_2, u_3 \in \mathbb{R}^n$
- Original verified type: ϵ-ROA.
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1^2 + x_2^2 + x_3^2 \le 2^2\}$.
- Our verified type: ROA verification.

System 27 (Example 5.2 and Table 2, Liu et al. (2024)). The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -0.5x_2 - \sin\left(x_1 + \dfrac{\pi}{3}\right) + \sin\left(\dfrac{\pi}{3}\right) \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = [-2, 3] \times [-3, 1.5]$.
- Original verified type: ROA verification.
- Verified method in the original paper: dReal with Jacobian-based local verification.
- Our $\mathcal{N} = [-2, 3] \times [-3, 1.5]$.
- Our verified type: ROA verification.

System 28 (Example 5.4 and Table 3, Liu et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \end{pmatrix} = \begin{pmatrix} -x_1 + 0.5x_2 - 0.1x_9^2 \\ -0.5x_1 - x_2 \\ -x_3 + 0.5x_4 - 0.1x_1^2 \\ -0.5x_3 - x_4 \\ -x_5 + 0.5x_6 + 0.1x_7^2 \\ -0.5x_5 - x_6 \\ -x_7 + 0.5x_8 \\ -0.5x_7 - x_8 \\ -x_9 + 0.5x_{10} \\ -0.5x_9 - x_{10} + 0.1x_2^2 \end{pmatrix}.$$

- $x^\star = \mathbf{0}$.
- Original $\mathcal{N} = [-1, 1]^{10}$.
- Original verified type: ROA verification.
- Verified method in the original paper: dReal with Jacobian-based local verification.
- Our $\mathcal{N} = [-1, 1]^{10}$.
- Our verified type: ROA verification.

System 29 (Section 5.1, with verification details in Appendix E.2 ($n = 1$), Meng et al. (2024)).

The closed-loop scalar dynamics are:

$$\dot{x} = -x - x^3,$$

- Equilibrium: $x^\star = 0$.
- Original $\mathcal{N} = \{x \in \mathbb{R} \mid 0.1 \le |x| \le 1\}$.

- Original verified type: ϵ-ROA
- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{x \in \mathbb{R} \mid |x| \leq 1\}$.
- Our verified type: ROA verification.

F.5 PROBLEM 3 :FORWARD-INVARIANT SET SYNTHESIS

System 30 (Appendix A.1 (PVTOL), Wu et al. (2023)).

Let the state be ordered as

$$\mathbf{x} = (p_x, p_z, \phi, v_x, v_z, \omega)^\top \in \mathbb{R}^6,$$

where (p_x, p_z) are the planar position, ϕ is the roll angle, (v_x, v_z) are the body-frame velocities, and ω is the angular rate. Using the sampling period $\Delta t = 0.05$, the discrete-time transition map is:

$$\mathbf{x}^+ = F(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} p_x + 0.05\,(v_x \cos\phi - v_z \sin\phi) \\ p_z + 0.05\,(v_x \sin\phi + v_z \cos\phi) \\ \phi + 0.05\,\omega \\ v_x + 0.05\,(v_z\omega - 9.8\sin\phi) \\ v_z + 0.05\left(-v_x\omega - 9.8\cos\phi + \dfrac{u_1+u_2}{4}\right) \\ \omega + 0.05\left(\dfrac{0.25}{0.0475}(u_1 - u_2)\right) \end{pmatrix},$$

where $\mathbf{u} = (u_1, u_2)^\top \in [0, 39.2]^2$.

- $\mathcal{X} = \left\{\mathbf{x} \in \mathbb{R}^6 \mid \|\mathbf{x}\|_\infty \leq 1\right\}$.
- Unsafe region: $\mathcal{X}_U = \left\{\mathbf{x} \in \mathbb{R}^6 \mid \|\mathbf{x}\|_\infty > 1\right\}$.
- Verification target: forward-invariance verification.

System 31 (Example 1, Samanipour & Poonawala (2026)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ \sin(x_1) + u \end{pmatrix},$$

where $u = -3x_1 - 3x_2$ with $u \in [-1.5, 1.5]$.

- $\mathcal{X} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq \pi\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^2 \mid \|x\|_\infty > \pi\}$.
- Verified type: forward-invariance verification.
- Verified method in the original paper: linear programming and facet-wise verification.

System 32 (Example 2, Samanipour & Poonawala (2026)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1^2 + x_1x_2 + x_1 \\ x_1x_2 + x_2^2 + x_2 \end{pmatrix}.$$

- $\mathcal{X} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq 2\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^2 \mid -1 \leq x_1, x_2 \leq 0\}$.
- Verified type: forward-invariance verification.
- Verified method in the original paper: linear programming and facet-wise verification.

System 33 (Example 3, Samanipour & Poonawala (2026)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_2 \\ \dfrac{u + 0.1\, x_4^2 \sin x_3 - 0.981 \sin x_3 \cos x_3}{1 + 0.1 \sin^2 x_3} \\ x_4 \\ \dfrac{10.791 \sin x_3 - u \cos x_3 - 0.1\, x_4^2 \sin x_3 \cos x_3}{1 + 0.1 \sin^2 x_3} \end{pmatrix},$$

where $u = x_1 + 2.4109\, x_2 + 34.3620\, x_3 + 10.7009\, x_4$ with $u \in [-30, 30]$.

- $\mathcal{X} = \{x \in \mathbb{R}^4 \mid \|x\|_\infty \leq 1\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^4 \mid \|x\|_\infty > 1\}$.
- Verified type: forward-invariance verification.
- Verified method in the original paper: linear programming and facet-wise verification.

System 34 (Section 6.1.1, Vaisi et al. (2025)).

The discrete-time transition map is:

$$\mathbf{x}^+ = F(\mathbf{x}) = \begin{pmatrix} x_1 + 0.01\, x_2 \\ x_2 + 0.01(9.8 \sin x_1 + u) \end{pmatrix},$$

where $u \in [-10, 10]$.

- $\mathcal{X} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq \pi/4\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^2 \mid \|x\|_\infty > \pi/4\}$.
- Verified type: forward-invariance verification.
- Verified method in the original paper: neural-network reachability and hyperplane-region enumeration.

System 35 (Section 6.1.2, Vaisi et al. (2025)).

The discrete-time transition map is:

$$\mathbf{x}^+ = F(\mathbf{x}) = \begin{pmatrix} x_2 \\ 29.4 \sin x_1 + 300 \cos x_1 \tan x_3 + 15u \cos x_1 \cos^2 x_3 \\ u \end{pmatrix},$$

where $u \in [-10, 10]$.

- $\mathcal{X} = \{x \in \mathbb{R}^3 \mid \|x\|_\infty \leq 2.2\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^3 \mid \|x\|_\infty > 2.2\}$.
- Verified type: forward-invariance verification.
- Verified method in the original paper: neural-network reachability and hyperplane-region enumeration.

F.6 Problem 4 :Safety verification

System 36 (Theorem 4, Sogokon et al. (2018)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_1 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_I = \{(x_1, x_2) \mid x_1 \leq 0 \wedge x_2 \leq 0\}$.
- $\mathcal{X}_U = \{(x_1, x_2) \mid x_1 > 0\}$.

- Verified type: barrier/safety verification.
- Verified method in the original paper: vector barrier certificates.

System 37 (Example 1, Sogokon et al. (2018)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 + 2x_3 - x_1^2 + x_1x_3 - x_3^2 \\ -2 + x_1 - x_2 - x_2^2 \\ -2 - x_2 - x_3 + x_1^2 - x_1x_3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_I = \{\mathbf{x} \in \mathbb{R}^3 \mid -x_i \geq 0,\ i = 1, 2, 3\}$.
- $\mathcal{X}_U = \{\mathbf{x} \in \mathbb{R}^3 \mid x_1 + x_2 + x_3 \geq 1\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: vector barrier certificates and SOS programming.

System 38 (Example 3.1, Zhao et al. (2020)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -x_1 + \frac{1}{3}x_1^3 - x_2 \end{pmatrix}.$$

- $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^2 \mid -3.5 \leq x_1 \leq 2,\ -2 \leq x_2 \leq 1\}$.
- $\mathcal{X}_I = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 - 1.5)^2 + x_2^2 \leq 0.25\}$.
- $\mathcal{X}_U = \{\mathbf{x} \in \mathbb{R}^2 \mid (x_1 + 1)^2 + (x_2 + 1)^2 \leq 0.16\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: iSAT3.

System 39 (Table 1 row 9 (Barr1), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 + 2x_1x_2 \\ -x_1 - x_2^2 + 2x_1^2 \end{pmatrix}.$$

- $\mathcal{X} = [-2, 2]^2$.
- $\mathcal{X}_I = [0, 1] \times [1, 2]$.
- $\mathcal{X}_U = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 + x_2^2 \leq 0\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: dReal.

System 40 (Table 1 row 12 (Barr4), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \sin(\phi) \\ \cos(\phi) \\ -\sin(\phi) + \dfrac{3(x_1 \sin(\phi) + x_2 \cos(\phi))}{0.5 + x_1^2 + x_2^2} \end{pmatrix}.$$

- $\mathcal{X} = [-2, 2] \times [-2, 2] \times [-1.57, 1.57]$.
- $\mathcal{X}_I = [-0.1, 0.1] \times [-2, -1.8] \times [-0.52, 0.52]$.
- $\mathcal{X}_U = \{(x_1, x_2, \phi) \in \mathbb{R}^3 \mid x_1^2 + x_2^2 \leq 0.04\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: dReal.

System 41 (Table 1 row 17 (Hi-Ord8), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_0 \\ \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ -20x_7 + 170x_6 + 800x_5 + 2273x_4 \\ \quad + 3980x_3 + 4180x_2 + 2400x_1 + 576x_0 \end{pmatrix}.$$

- $\mathcal{X} = [-2.2, 2.2]^8$.
- $\mathcal{X}_I = [0.9, 1.1]^8$.
- $\mathcal{X}_U = [-2.2, -1.8]^8$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: dReal.

System 42 (Table 1 (arch2[36], instance 1), Wu et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2^2 - 1 \\ 5(x_1 x_2 - 1) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_I = \{\mathbf{x} \in \mathbb{R}^2 \mid 2.25 - x_1^2 - (x_2 - 2)^2 \geq 0\}$.
- $\mathcal{X}_U = \{\mathbf{x} \in \mathbb{R}^2 \mid -1 - x_1 \geq 0 \wedge x_1 + 3 \geq 0 \wedge -2 - x_2 \geq 0 \wedge x_2 + 3 \geq 0\}$.
- Verified type: unbounded-domain barrier verification.
- Verified method in the original paper: SDP-based barrier synthesis.

System 43 (Table 1 (arch3[36], instance 1), Wu et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 - x_1^3 + x_2 - x_1 x_2^2 \\ -x_1 + x_2 - x_1^2 x_2 - x_2^3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_I = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.04 - x_1^2 - x_2^2 \geq 0\}$.
- $\mathcal{X}_U = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.25 - (x_1 - 2.5)^2 - (x_2 - 2.5)^2 \geq 0\}$.
- Verified type: unbounded-domain barrier verification.
- Verified method in the original paper: SDP-based barrier synthesis.

System 44 (Example 5 and Table 1, Liu et al. (2015)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} e^{-x_1} + x_2 - 1 \\ -\sin^2(x_1) \end{pmatrix}.$$

- $\mathcal{X} = [-2, 2]^2$.
- $\mathcal{X}_I = \{(x_1, x_2) \mid (x_1 + 0.5)^2 + (x_2 - 0.5)^2 \leq 0.16\}$.
- $\mathcal{X}_U = \{(x_1, x_2) \mid (x_1 - 0.7)^2 + (x_2 + 0.7)^2 \leq 0.09\}$.
- Verified type: safety verification
- Verified method in the original paper: polynomial abstraction and SOS programming.

System 45 (Example 6 and Table 1, Liu et al. (2015)).

The vector flow field is:

$$\dot{\mathbf{u}} = \begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \end{pmatrix} = \begin{pmatrix} -\dfrac{\beta c u_1 u_2}{u_1+u_2+u_3} - \mu u_1 \\ \dfrac{\beta c u_1 u_2}{u_1+u_2+u_3} - (\mu+\nu) u_2 \\ \nu u_2 - \alpha u_3 \end{pmatrix},$$

where $\beta = 0.2$, $c = 10$, $\mu = 0.008$, $\alpha = 0.95$, and $\nu = 0.1$.

- $\mathcal{X} = \{\mathbf{u} \in \mathbb{R}^3 \mid u_i \geq 0,\ u_1 + u_2 + u_3 \leq 10.013\}$.
- $\mathcal{X}_I = [9.985, 9.995] \times [0.005, 0.015] \times [0, 0.003]$.
- $\mathcal{X}_U = \{\mathbf{u} \mid u_3 > 1\}$.
- Verified type: forward-invariance verification.
- Verified method in the original paper: polynomial abstraction and SOS programming.

System 46 (Table 1 (lotka[14], instance 1), Wu et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_1(1 - x_3) \\ x_2(1 - 2x_3) \\ x_3(x_1 + x_2 - 1) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_I = \{\mathbf{x} \in \mathbb{R}^3 \mid 0.09 - (x_1 - 1)^2 - (x_2 - 1)^2 - (x_3 - 1.5)^2 \geq 0\}$.
- $\mathcal{X}_U = \{\mathbf{x} \in \mathbb{R}^3 \mid 0.09 - (x_1 + 1)^2 - (x_2 + 1)^2 - (x_3 + 1.5)^2 \geq 0\}$.
- Verified type: unbounded-domain barrier verification.
- Verified method in the original paper: SDP-based barrier synthesis.

System 47 (Table 4 #5 (ctrlbarr obstacle), Edwards et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \sin(\phi) \\ \cos(\phi) \\ -\sin(\phi) + u_1 \end{pmatrix}.$$

- $\mathcal{X} = [-10, 10] \times [-10, 10] \times [-\pi, \pi]$.
- $\mathcal{X}_I = [4, 6] \times [4, 6] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$.
- $\mathcal{X}_U = [-9, -7] \times [-9, -6] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: dReal.

System 48 (Table 4 #17 (ctrlbarr 2roomtemp DT), Edwards et al. (2024)).

The discrete-time transition map is:

$$\mathbf{x}^+ = F(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} 0.725x_1 + 0.25x_2 + 0.375 + 0.018(55 - x_1)u_1 \\ 0.25x_1 + 0.71x_2 + 0.6 + 0.018(55 - x_2)u_2 \end{pmatrix}.$$

- $\mathcal{X} = [17, 30]^2$.
- $\mathcal{X}_I = [17, 18]^2$.
- $\mathcal{X}_U = [28, 30]^2$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: dReal.

System 49 (`S001`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.8\,x_1 - 0.15\,x_2 + 0.2\,x_1^2 + u_1 \\ -0.1\,x_3 + 0.1\,x_1 + u_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.4\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 3\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 50 (`S002`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.95\,\sin(x_1) - 0.02\,x_2 + 0.01\,x_3 + u_1 \\ -0.1\,x_3 + 0.02\,x_1 + u_2 \\ -0.03\,x_4 + 0.01\,x_2 + u_3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 0.3\}$.
- $\mathcal{X}_U = \mathbb{R}^4 \setminus \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 2.5\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices with Taylor approximations.

System 51 (`S004`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 10\,(x_2 - x_1) + u_0 \\ 28\,x_1 - x_2 - x_1\,x_3 + u_1 \\ x_1\,x_2 - \frac{8}{3}\,x_3 + u_2 \\ -x_2\,x_3 + 2.7\,x_4 + u_3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 0.1\}$.
- $\mathcal{X}_U = \mathbb{R}^4 \setminus \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 25\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 52 (`S008`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -2\,x_1 + 6.7\,x_2 - x_2\,x_3 + u_0 \\ x_1 + u_1 \\ -x_3 + x_2^2 + u_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.2\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 5\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 53 (`S011`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.3\,x_1 - 0.2\,(1 + \cos(x_1))\,x_2 - 0.1\,x_1^3 + u_1 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^2 \mid \|x\|_2 \leq 0.6\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^2 \mid \|x - (2,2)^\mathsf{T}\|_2 \leq 1\}$.

- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices with Taylor approximations.

System 54 (`S019`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -x_1 + x_2\, x_3 \\ -x_2 + x_1\, x_3 \\ -x_3 - x_1\, x_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.2\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 1\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 55 (`S024`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -x_1 - 0.4\, x_2 + 0.15\, \sin(x_3) + u_1 \\ x_4 + u_2 \\ -0.6\, x_3 - 0.3\, x_4 + u_3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = [0.5, 0.8] \times [0.3, 0.6] \times [0.5, 0.8] \times [0.3, 0.6]$.
- $\mathcal{X}_U = [2.5, 3.5] \times [2, 3] \times [2.5, 3.5] \times [2, 3] \cup [-3.5, -2.5] \times [-3, -2] \times [-3.5, -2.5] \times [-3, -2]$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices with Taylor approximations.

System 56 (`S028`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -0.4\, x_1 + 0.25\, x_2\, x_3 - 0.15\, x_1^3 \\ -0.5\, x_2 + 0.3\, x_1\, x_3 - 0.2\, x_2^3 \\ -0.6\, x_3 + 0.2\, x_1\, x_2 - 0.18\, x_1^2\, x_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.5\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 3.5\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 57 (`S031`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} -0.3\, x_1 + 0.2\, x_2\, x_4 - 0.15\, x_3^2 \\ -0.4\, x_2 + 0.25\, x_1\, x_3 - 0.18\, x_2\, x_4 \\ -0.5\, x_3 + 0.3\, x_1\, x_4 + 0.12\, x_2^2 \\ -0.6\, x_4 + 0.2\, x_2\, x_3 - 0.1\, x_1^2\, x_4 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 0.4\}$.
- $\mathcal{X}_U = \mathbb{R}^4 \setminus \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 3\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 58 (S036, Taheri et al. (2025)).

The update map is:

$$\mathbf{x}_{k+1} = \begin{pmatrix} x_{1,k+1} \\ x_{2,k+1} \end{pmatrix} = \begin{pmatrix} 0.85\, x_{1,k} + u_{0,k} \\ 0.8\, x_{2,k} + 0.15\, x_{1,k} + u_{1,k} \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_0 = [1.5, 2] \times [1.2, 1.8]$.
- $\mathcal{X}_U = [4.5, 5.5] \times [4, 5] \cup [-5.5, -4.5] \times [-5, -4]$.
- Verified type: barrier/safety verification.

System 59 (S037, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.2\, x_1 - 0.1\, (x_1^2 - 1)\, x_2 + u_1 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^2 \mid \|x\|_2 \leq 0.5\}$.
- $\mathcal{X}_U = \mathbb{R}^2 \setminus \{x \in \mathbb{R}^2 \mid \|x\|_2 \leq 2.8\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 60 (S042, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.6\, x_1 - 0.1\, x_2 + 0.05\, x_3 + u_1 \\ x_4 + u_2 \\ -0.8\, x_3 - 0.15\, x_4 + 0.05\, x_1 + u_3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 0.6\}$.
- $\mathcal{X}_U = \mathbb{R}^4 \setminus \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 3.5\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 61 (S048, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -0.5\, x_1 + 0.3\, x_2\, \sin(x_3) - 0.1\, x_1^2\, x_2 \\ -0.6\, x_2 + 0.5\, x_1\, \cos(x_3) + 0.15\, x_3^2\, x_1 \\ -0.7\, x_3 - 0.1\, x_3^3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.4\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 3.5\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices with Taylor approximations.

System 62 (S052, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ x_3 + u_1 \\ -6\, x_3 - 2.92\, x_2 - 1.2\, x_1 + x_1^2 + u_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.1\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 2\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 63 (`S057`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1 + 0.08\, x_2 + x_1^2\, x_2 + u_0 \\ 0.6 - 0.08\, x_2 - x_1^2\, x_2 + u_1 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_0 = [0.1, 0.5] \times [1, 2]$.
- $\mathcal{X}_U = \mathbb{R}^2 \setminus \{x \in \mathbb{R}^2 \mid \|x\|_2 \leq 5\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 64 (`S058`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.1\, x_1 - 0.05\, x_2 + 0.02\, x_1\, x_3 + u_1 \\ -0.2\, x_3 + 0.1\, x_1 + u_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.5\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 3\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 65 (`S059`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -0.25\, x_1 - x_2^2 - x_3^2 + 0.75 + u_0 \\ -x_2 + x_1\, x_3 + u_1 \\ -x_3 + x_1\, x_2 - 0.1\, x_1 + u_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.2\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 3\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 66 (`S060`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1\,(x_1 - 0.1)\,(1 - x_1) - 0.3\, x_1\, x_2 + u_0 \\ -0.2\, x_2 + 0.1\, x_1\, x_2 + u_1 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_0 = [0.2, 0.8] \times [0.1, 0.6]$.
- $\mathcal{X}_U = [1.5, 2] \times [1, 1.5]$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 67 (S062, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -x_1 + x_2\, x_3 + u_1 \\ 1 - x_2^2 + u_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.2\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 2.5\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 68 (S076, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_3 + x_2\, x_1 + u_0 \\ -x_2 - x_1^2 + u_1 \\ -x_1 - 0.5\, x_3 + u_2 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 0.3\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \leq 3.5\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 69 (S077, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.8\, x_1 - 0.3\, x_2\, (1 + x_1^2) + 0.1\, x_3 + u_1 \\ x_4 + u_2 \\ -0.9\, x_3 - 0.25\, x_4\, (1 + x_3^2) + 0.1\, x_1 + u_3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 0.35\}$.
- $\mathcal{X}_U = \mathbb{R}^4 \setminus \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 2.8\}$.
- Verified type: barrier/safety verification.
- Verified method in the original paper: Z3/Yices.

System 70 (S087, Taheri et al. (2025)).

The update map is:

$$\mathbf{x}_{k+1} = \begin{pmatrix} x_{1,k+1} \\ x_{2,k+1} \end{pmatrix} = \begin{pmatrix} x_{1,k} + 0.01\,(-100\, x_{1,k} - 1\, x_{2,k}) \\ x_{2,k} + 0.01\,(1\, x_{1,k} - 100\, x_{2,k}) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_0 = [0.1, 0.4] \times [0.1, 0.55]$.
- $\mathcal{X}_U = [0.45, 0.5] \times [0.6, 1]$.
- Verified type: barrier/safety verification.

System 71 (U000, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} -0.5\, x_1 + 0.3\, x_2 \\ -0.4\, x_2 + 0.15\, \cos(x_1)\, \sin(x_3) \\ -0.6\, x_3 + 0.1\, \sin(x_2)\, \cos(x_4) \\ -0.5\, x_4 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^4 \mid \|x\|_2 \le 0.3\}$.
- $\mathcal{X}_U = \mathbb{R}^4 \setminus \{x \in \mathbb{R}^4 \mid \|x\|_2 \le 2.5\}$.
- Verified type: barrier/safety verification.

System 72 (`U001`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -0.3\,x_1 + 0.5\,\sin(x_2) \\ -0.4\,x_2 - 0.6\,\sin(x_1) + 0.1\,\sin(x_3) \\ -0.5\,x_3 + 0.2\,\sin(x_1)\,\sin(x_2) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \le 0.4\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \le 3\}$.
- Verified type: barrier/safety verification.

System 73 (`U002`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -0.3\,x_1 + 0.2\,\sin(x_2)\,\exp(-x_3^2) \\ -0.4\,x_2 + 0.15\,\cos(x_1)\,\exp(-x_1^2) \\ -0.5\,x_3 + 0.1\,\sin(x_1 + x_2) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x\|_2 \le 0.45\}$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \{x \in \mathbb{R}^3 \mid \|x\|_2 \le 3.2\}$.
- Verified type: barrier/safety verification.

System 74 (`U003`, Taheri et al. (2025)).

The update map is:

$$\mathbf{x}_{k+1} = \begin{pmatrix} x_{1,k+1} \\ x_{2,k+1} \\ x_{3,k+1} \end{pmatrix} = \begin{pmatrix} 0.8\,x_{1,k} + 0.1\,\sin(x_{2,k}) \\ 0.85\,x_{2,k} + 0.08\,x_{1,k}\,x_{3,k} \\ 0.9\,x_{3,k} + 0.08\,\cos(x_{1,k}\,x_{2,k}) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = [0.1, 0.4] \times [0.1, 0.3] \times [0.1, 0.4]$.
- $\mathcal{X}_U = [2, 2.5] \times [1.8, 2.2] \times [1.8, 2.3]$.
- Verified type: barrier/safety verification.

System 75 (`U004`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} -0.65\,x_1 + 0.1\,x_2^2\,\cos(x_3) \\ -0.7\,x_2 + 0.12\,\sin(x_1)\,\tanh(x_4) \\ -0.75\,x_3 + 0.08\,x_1\,x_2 \\ -0.6\,x_4 + 0.06\,(x_1 + x_2) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = [0.2, 0.5] \times [-0.3, 0] \times [0.1, 0.4] \times [-0.2, 0.1]$.
- $\mathcal{X}_U = [-2.2, -1.7] \times [1.6, 2.1] \times [-1.9, -1.4] \times [1.3, 1.8]$.
- Verified type: barrier/safety verification.

System 76 (`U005`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -0.58\,x_1 + 0.12\,\sin(x_2)\,\cos(x_3) \\ -0.62\,x_2 + 0.1\,\exp(-x_1^2)\,\tanh(x_3) \\ -0.66\,x_3 + 0.08\,\sin(x_1 x_2)\,\exp(-x_3^2) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = [-0.3, 0.2] \times [0.1, 0.4] \times [-0.2, 0.3]$.
- $\mathcal{X}_U = [-2.3, -1.8] \times [1.7, 2.2] \times [-2, -1.5]$.
- Verified type: barrier/safety verification.

System 77 (`U006`, Taheri et al. (2025)). The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} u_0 \cos(x_3) \\ u_0 \sin(x_3) \\ u_1 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = [-1, 1] \times [-1, 1] \times [-0.5, 0.5]$.
- $\mathcal{X}_U = \mathbb{R}^3 \setminus \big([-5, 5] \times [-5, 5] \times [-3.14, 3.14]\big)$.
- Verified type: barrier/safety verification.

System 78 (`U007`, Taheri et al. (2025)).

The update map is:

$$\mathbf{x}_{k+1} = \begin{pmatrix} x_{1,k+1} \\ x_{2,k+1} \\ x_{3,k+1} \end{pmatrix} = \begin{pmatrix} 0.75\, x_{1,k} + 0.1\, \tanh(x_{2,k})\, \tanh(x_{3,k}) \\ 0.82\, x_{2,k} + 0.09\, \sin(x_{1,k})\, \exp(-x_{3,k}^2) \\ 0.79\, x_{3,k} + 0.08\, \cos(x_{1,k})\, \cos(x_{2,k}) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x - (1, -0.5, 0.8)^\mathsf{T}\|_2 \le 0.5\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^3 \mid \|x - (-2.5, 2, -2.2)^\mathsf{T}\|_2 \le 0.6\}$.
- Verified type: barrier/safety verification.

System 79 (`U008`, Taheri et al. (2025)).

The update map is:

$$\mathbf{x}_{k+1} = \begin{pmatrix} x_{1,k+1} \\ x_{2,k+1} \\ x_{3,k+1} \end{pmatrix} = \begin{pmatrix} 0.73\, x_{1,k} + 0.13\, \sin(x_{2,k})\, \tanh(x_{3,k}) \\ 0.89\, x_{2,k} + 0.17\, \cos(x_{1,k}^2)\, \exp(-x_{3,k}^2) \\ 0.64\, x_{3,k} + 0.22\, \tanh(x_{1,k}\, x_{2,k}) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^3 \mid \|x - (0.9, -0.7, 1.2)^\mathsf{T}\|_2 \le 0.55\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^3 \mid \|x - (-3.1, 2.8, -2.6)^\mathsf{T}\|_2 \le 0.75\}$.
- Verified type: barrier/safety verification.

System 80 (`U009`, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} -0.62\, x_1 + 0.1\, \sin(x_2 + x_3) \\ -0.67\, x_2 + 0.08\, \cos(x_1)\, \exp(-x_3^2) \\ -0.64\, x_3 + 0.09\, \tanh(x_1)\, \sin(x_2) \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^3$.
- $\mathcal{X}_0 = [-0.2, 0.3] \times [0.1, 0.5] \times [-0.3, 0.2]$.
- $\mathcal{X}_U = [2, 2.6] \times [-2.4, -1.8] \times [2.1, 2.7]$.
- Verified type: barrier/safety verification.

F.7 Extended Tier: Scaled and Perturbed Variants of Retained Systems

We further extend some of the systems described above in several ways, including strengthening the verification objective from asymptotic stability to exponential stability, incorporating control constraints, scaling existing systems to higher dimensions, and introducing parameter variations.

Problem. 1:

System 81 (Extend-Table 1 row 1 (NonPoly0), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -x + xy \\ -y \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- $\mathcal{N} = \mathbb{R}^2$.
- Verified type: Global asymptotic stability

We modify the local asymptotic stability to the global asymptotic stability.

System 82 (Extend-Table 1 row 6 (Poly2), Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_1^3 + x_2 \\ -x_1 - x_2 \end{pmatrix}.$$

- $x^\star = (0, 0)$.
- $\mathcal{N} = \mathbb{R}^2$.
- Verified type: Global exponential stability

We modify the local asymptotic stability to the global exponential stability.

System 83 (Extend-Table 4 #2 (SineModel), Edwards et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 + u_1 \\ u_2 + \dfrac{mgL\sin(x_1) - bx_2}{mL^2} \end{pmatrix},$$

where $m = 0.15$, $g = 9.81$, $L = 0.5$, and $b = 0.1$.

- $x^\star = (0, 0)$.
- $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1^2\}$.
- $u_1, u_2 \in [-5, 5]$ when $\mathbf{x} \in \mathcal{N}$ and $u_1(0) = u_2(0) = 0$.
- Verified type: Exponentially stable on $\mathcal{N}$.
- Verified method in the original paper: dReal.

We add control constrains $u_1, u_2 \in [-5, 5], u_1(0) = u_2(0) = 0$ and change the local asymptotic stability to the local exponentially stability.

Problem. 2:

System 84 (Extend-Section 5.2 and Appendix C.3.2, Zhou et al. (2022)).

The closed-loop vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} \sin(x_2) \\ u - \dfrac{\cos(x_2)}{1 - x_1} \end{pmatrix}$$

- $x^\star = (0, 0)$.
- Original $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid 0.1^2 \leq x_1^2 + x_2^2 \leq 0.8^2\}$.
- Original verified type: ϵ-ROA.

- Verified method in the original paper: dReal.
- Our $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 0.8^2\}$.
- Our verified type: ROA verification.
- $u \in [-5, 5]$ when $\mathbf{x} \in \mathcal{N}$ and $u(0, 0) = 1$.

In the original paper, the control policy was $u = 5\tanh(q_1 x_1 + q_2 x_2 + b)$ which contributes to the $(0, 0)^\top$ is not the equilibrium point. So we just change the goal to find a policy $u \in [-5, 5]$ such that $u\big((0, 0)^\top\big) = 1$.

Problem. 3:

System 85 (Extend–2D quadrotor, Meng et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ 9.81\sin(x_3) + \sin(x_3)u_1 \\ -9.81 + 9.81\cos(x_3) + \cos(x_3)u_1 \\ u_2 \end{pmatrix},$$

where $\mathbf{u} = (u_1, u_2)^\top$ is the control input to be synthesized.

- $\mathcal{X} = \{x \in \mathbb{R}^6 \mid \|x\|_\infty \leq 0.2\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^6 \mid \|x\|_\infty > 0.2\}$.
- $u_1, u_2 \in [-5, 5]$ when $\mathbf{x} \in \mathcal{X}$.
- Verified type: controlled forward-invariant set synthesis.
- Verified method in the original paper: no formal invariance certificate was reported for this instance.

We add control constrains $u_1, u_2 \in [-5, 5]$.

System 86 (Extend–3D quadrotor, Meng et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -9.81\sin(x_8) - \sin(x_8)u_1 \\ 9.81\cos(x_8)\sin(x_7) + \cos(x_8)\sin(x_7)u_1 \\ 9.81\cos(x_8)\cos(x_7) - 9.81 + \cos(x_8)\cos(x_7)u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix},$$

where $\mathbf{u} = (u_1, u_2, u_3, u_4)^\top$ is the control input to be synthesized.

- $\mathcal{X} = \{x \in \mathbb{R}^9 \mid \|x\|_\infty \leq 0.2\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^9 \mid \|x\|_\infty > 0.2\}$.
- $u_i \in [-5, 5]$ when $\mathbf{x} \in \mathcal{X} \quad i = 1, 2, 3, 4$.
- Verified type: controlled forward-invariant set synthesis.
- Verified method in the original paper: no formal invariance certificate was reported for this instance.

We add the control constraints $u_i \in [-5, 5]$ when $\mathbf{x} \in \mathcal{X} \quad i = 1, 2, 3, 4$.

Problem. 4:

System 87 (Extend-S002, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.95\,\sin(x_1) - 0.02\,x_2 + 0.01\,x_3 + u_1 \\ -0.1\,x_3 + 0.02\,x_1 + u_2 \\ -0.03\,x_4 + 0.01\,x_2 + u_3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 0.3\}$.
- $\mathcal{X}_U = \mathbb{R}^4 \setminus \{x \in \mathbb{R}^4 \mid \|x\|_2 \leq 2.5\}$.
- $u_i \in [-5, 5] (i = 0, 1, 2, 3)$ when $\mathbf{x} \in \mathcal{X}$
- Verified type: safety verification.
- Verified method in the original paper: Z3/Yices with Taylor approximations.

We add the control constrains $u_i \in [-5, 5]; i = 0, 1, 2, 3$.

System 88 (Extend-S011, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -0.3\,x_1 - 0.2\,(1 + \cos(x_1))\,x_2 - 0.1\,x_1^3 + u_1 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^2$.
- $\mathcal{X}_0 = \{x \in \mathbb{R}^2 \mid \|x\|_2 \leq 0.6\}$.
- $\mathcal{X}_U = \{x \in \mathbb{R}^2 \mid \|x - (2, 2)^\mathsf{T}\|_2 \leq 1\}$.
- $u_{0,1} \in [-1, 1]$ when $x \in \mathcal{X}$.
- Verified type: safety verification.
- Verified method in the original paper: Z3/Yices with Taylor approximations.

We add the control constrains $u_{0,1} \in [-1, 1]^2$.

System 89 (Extend-S024, Taheri et al. (2025)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -x_1 - 0.4\,x_2 + 0.15\,\sin(x_3) + u_1 \\ x_4 + u_2 \\ -0.6\,x_3 - 0.3\,x_4 + u_3 \end{pmatrix}.$$

- $\mathcal{X} = \mathbb{R}^4$.
- $\mathcal{X}_0 = [0.5, 0.8] \times [0.3, 0.6] \times [0.5, 0.8] \times [0.3, 0.6]$.
- $\mathcal{X}_U = [2.5, 3.5] \times [2, 3] \times [2.5, 3.5] \times [2, 3] \cup [-3.5, -2.5] \times [-3, -2] \times [-3.5, -2.5] \times [-3, -2]$.
- $u_i \in [-1, 1], i = 0, 1, 2, 3$ when $x \in \mathcal{X}$.
- Verified type: safety verification.
- Verified method in the original paper: Z3/Yices with Taylor approximations.

We add the control constrains $u_i \in [-1, 1], i = 0, 1, 2, 3$.

System 90 (Extend–Hi-Ord8 with bounded sinusoidal disturbance, Abate et al. (2021)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_0 \\ \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ -20x_7 + 170x_6 + 800x_5 + 2273x_4 \\ \quad + 3980x_3 + 4180x_2 + 2400x_1 + 576x_0 \\ \quad + \sin(\omega x_0) \end{pmatrix}.$$

- $\mathcal{X} = [-2.2, 2.2]^8$.
- $\mathcal{X}_I = [0.9, 1.1]^8$.
- $\mathcal{X}_U = [-2.2, -1.8]^8$.
- $\omega \in [0.5, 2]$.
- Verified type: safety verification

The safety property is verified uniformly for all admissible values of $\omega \in [0.5, 2]$ which is a time-invariant parameter.

Remark 4. *In this paper, we mainly consider the Barrier function is only related to the x, not with uncertain factor w, i.e. we require that, for any $w \in [0.5, 2]$, the barrier conditions hold.*

System 91 (Safety extension of Example 5.4, Liu et al. (2024)).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \end{pmatrix} = \begin{pmatrix} -x_1 + 0.5x_2 - 0.1x_9^2 + \sin(x_2) \\ -0.5x_1 - x_2 \\ -x_3 + 0.5x_4 - 0.1x_1^2 + e^{-x_3} - 1 \\ -0.5x_3 - x_4 \\ -x_5 + 0.5x_6 + 0.1x_7^2 \\ -0.5x_5 - x_6 \\ -x_7 + 0.5x_8 \\ -0.5x_7 - x_8 \\ -x_9 + 0.5x_{10} \\ -0.5x_9 - x_{10} + 0.1x_2^2 \end{pmatrix}.$$

- $\mathcal{X} = [-1, 1]^{10}$.
- $\mathcal{X}_I = [-0.35, 0.35]^{10}$.
- $\mathcal{X}_U = [0.45, 0.8] \times [-0.2, 0.2] \times [-0.4, 0.4]^8$.
- Verified type: safety verification

We augment the dynamics by introducing two perturbations, $\sin(x_2)$ to the vector field of x_1 and e^{-x_3} to the vector field of x_3.

F.8 NEW: NEWLY DESIGNED CHALLENGE INSTANCES

Given the limited availability of non-polynomial and high-dimensional systems in the existing literature, we additionally introduce the following systems.

Problem. 1 :

System 92 (8D Rational–Cosine Ring). The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \end{pmatrix} = \begin{pmatrix} -3x_1 + x_2 \frac{\cos x_1}{1+x_8^2} \\ -3x_2 + x_3 \frac{\cos x_2}{1+x_1^2} \\ -3x_3 + x_4 \frac{\cos x_3}{1+x_2^2} \\ -3x_4 + x_5 \frac{\cos x_4}{1+x_3^2} \\ -3x_5 + x_6 \frac{\cos x_1}{1+x_4^2} \\ -3x_6 + x_7 \frac{\cos x_2}{1+x_5^2} \\ -3x_7 + x_8 \frac{\cos x_3}{1+x_6^2} \\ -3x_8 + x_1 \frac{\cos x_4}{1+x_7^2} \end{pmatrix}.$$

- $x^\star = (0, 0, 0, 0, 0, 0, 0, 0)$.
- $\mathcal{N} = \mathbb{R}^8$.
- Verified type: Global exponential stability

System 93 (7D Heterogeneous Trigonometric–Rational Network).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} -5x_1 + x_2 \dfrac{\cos(x_1)}{1+x_7^2} \\ -\dfrac{9}{2}x_2 + \dfrac{6}{5}x_4 \dfrac{\sin(2x_2)}{1+2x_1^2} \\ -6x_3 + \dfrac{4}{5}x_6 \dfrac{\cos(x_3/2)}{1+\frac{3}{2}x_2^2} \\ -\dfrac{11}{2}x_4 + \dfrac{3}{2}x_7 \dfrac{\sin(3x_4/2)}{1+3x_3^2} \\ -5x_5 + \dfrac{9}{10}x_1 \dfrac{\cos(2x_5)}{1+\frac{4}{3}x_4^2} \\ -\dfrac{9}{2}x_6 + \dfrac{7}{5}x_5 \dfrac{\sin(x_6/3)}{1+\frac{5}{2}x_5^2} \\ -6x_7 + \dfrac{1}{2}x_3 \dfrac{\cos(5x_7/2)}{1+\frac{1}{2}x_5^2} \end{pmatrix}.$$

- $x^\star = (0, 0, 0, 0, 0, 0, 0)$.
- $\mathcal{N} = \mathbb{R}^7$.
- Verified type: Global exponential stability.

Problem. 2 :

System 94 (Our cyclic ten-dimensional rational system).

The vector flow field is

$$\dot{x}_i = -x_i + \frac{1}{4}x_{i+1} + \frac{1}{20}\frac{x_{i+2}}{1+x_{i+2}^2}, \qquad i = 1, \ldots, 10,$$

where the indices are interpreted cyclically, with $x_{11} = x_1$ and $x_{12} = x_2$.

- $x^\star = \mathbf{0} \in \mathbb{R}^{10}$.
- $\mathcal{N} = \mathbb{R}^{10}$.
- Verified type: ROA.

System 95 (Seven-dimensional multi-machine swing system).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{1}{2}x_2 - \frac{3}{2}\sin x_1 - \sin(x_1 - x_3) - \frac{3}{2}\sin(x_1 - x_7) \\ x_4 \\ -\frac{1}{2}x_4 - \frac{1}{2}\sin x_3 - \sin(x_3 - x_1) - \sin(x_3 - x_5) \\ x_6 \\ -\frac{1}{2}x_6 - \sin x_5 - \sin(x_5 - x_3) - \frac{3}{2}\sin(x_5 - x_7) \\ -\frac{1}{5}\sin x_7 - \frac{3}{2}\sin(x_7 - x_1) - \frac{3}{2}\sin(x_7 - x_5) \end{pmatrix}.$$

- $x^\star = (0, 0, 0, 0, 0, 0, 0)$.
- $\mathcal{N} = [-\pi, \pi]^7$.
- $\mathcal{X}_I = [-0.6, 0.6]^7$.
- Our verified type: ROA verification.

System 96 (Seven-dimensional coupled saturating oscillator).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\frac{2x_1}{1 + x_1^2} + \frac{1}{2}x_3 - 0.3\,x_2 + 0.2\,x_5 \\ x_4 \\ \frac{1}{2}x_1 - \frac{3}{2}x_3 - \sin(x_3) - 0.2\,x_4 + 0.1\,x_6 \\ -0.5\,x_5 - 0.2\,x_2 + 0.1\,x_6 \\ -0.4\,x_6 - 0.1\,x_5 - 0.1\,x_4 + 0.1\,x_7 \\ -0.3\,x_7 - 0.1\,x_6 \end{pmatrix}.$$

- $x^\star = (0, 0, 0, 0, 0, 0, 0)$.
- $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^7 \mid -5 \le x_i \le 5,\ i = 1, \ldots, 7\}$.
- $\mathcal{X}_I = \{\mathbf{x} \in \mathbb{R}^7 \mid -0.6 \le x_i \le 0.6,\ i = 1, \ldots, 7\}$.
- Our verified type: ROA verification.

Problem. 3 :

System 97 (New-6D Cosine Cascade Invariance).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x_2 + u_0 \\ -x_1 - \frac{1}{10}\sin(x_1) - x_2 + x_3 \\ -2x_3 + x_4 \\ -\frac{3}{2}x_4 + x_5 \\ -2x_5 + x_6 \\ -\frac{3}{2}x_6 + u_1 \end{pmatrix}.$$

- $\mathcal{X} = [-1, 1]^6$.
- $\mathcal{X}_{\text{seed}} = \left\{x \in \mathbb{R}^6 \mid \|x\|_2 \le \frac{1}{2}\right\}$.
- $\mathcal{X}_U = \left\{x \in \mathbb{R}^6 \mid \left\|x - \left(0, \frac{4}{5}, 0, 0, 0, 0\right)^\top\right\|_2 \le \frac{1}{10}\right\}$.
- $u_0, u_1 \in [-1, 1]$ for every $x \in \mathcal{X}$.

- The requested output is a nonempty controlled forward-invariant set $\mathcal{S} \subseteq \mathcal{X} \setminus \mathcal{X}_U$ containing $\mathcal{X}_{\text{seed}}$.
- Task type: controlled forward-invariant set synthesis

System 98 (New-6D Exponential Cascade Invariance).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} x_2 + u \\ -x_1 - \dfrac{1}{20}\left(\exp(x_1) - \exp(-x_1)\right) - x_2 + x_3 \\ -2x_3 + x_4 \\ -\dfrac{3}{2}x_4 + x_5 \\ -2x_5 + x_6 \\ -\dfrac{3}{2}x_6 \end{pmatrix}.$$

- $\mathcal{X} = [-1, 1]^6$.
- $\mathcal{X}_I = \left\{x \in \mathbb{R}^6 \mid \left(x_1 + \frac{1}{4}\right)^2 + \left(x_2 - \frac{1}{4}\right)^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \leq \frac{1}{64}\right\}$.
- $\mathcal{X}_U = \left\{x \in \mathbb{R}^6 \mid \left(x_1 - \frac{7}{10}\right)^2 + \left(x_2 + \frac{7}{10}\right)^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \leq \frac{1}{100}\right\}$.
- $u \in [-1, 1]$ for every $\mathbf{x} \in \mathcal{X}$.
- The requested output is a controlled forward-invariant set $\mathcal{S} \subseteq \mathcal{X} \setminus \mathcal{X}_U$ containing $\mathcal{X}_I$.
- Task type: controlled forward-invariant set synthesis.

System 99 (New-5D Mixed Nonlinear Cascade Invariance).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{pmatrix} = \begin{pmatrix} x_2 - \dfrac{1}{2}x_1 \\ -x_1 - \dfrac{x_1}{5(1 + x_1^2)} - 2x_2 + \dfrac{x_3}{10(1 + x_3^2)} \\ x_4 - \dfrac{1}{2}x_3 \\ -x_3 - \dfrac{1}{40}\left(\exp(x_3) - \exp(-x_3)\right) - 2x_4 + \dfrac{1}{5}x_5 \\ -\dfrac{3}{2}x_5 - \dfrac{1}{10}x_5^3 + u \end{pmatrix}.$$

- $\mathcal{X} = [-1, 1]^5$.
- $\mathcal{X}_U = \left\{x \in \mathbb{R}^5 \mid \left(x_1 - \frac{7}{10}\right)^2 + \left(x_2 + \frac{7}{10}\right)^2 + x_3^2 + x_4^2 + x_5^2 \leq \frac{1}{100}\right\}$.
- $u \in [-1, 1]$ for every $x \in \mathcal{X}$.
- Task type: controlled forward-invariant set synthesis.

System 100 (New-6D Opinion Dynamics).

Consider an undirected network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $N = 6$ nodes and adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

where $A_{ij} = 1$ means that nodes i and j are connected. The discrete-time dynamics are

$$x_i(k+1) = x_i(k) + \frac{1}{4}\sum_{j=1}^{6} A_{ij} \tanh\big(x_j(k) - x_i(k)\big), \qquad i = 1, \ldots, 6.$$

- $\mathcal{X} = [-1, 1]^6$.
- $\mathcal{X}_U = \{x \in \mathcal{X} \mid \exists i \in \{1, \ldots, 6\},\ |x_i| \geq \frac{4}{5}\}$.
- Verified type: forward invariance verification.

System 101 (Invariance Verification of a Discrete-Time Controlled Hopfield Network).

Consider the closed-loop 10-dimensional Hopfield network $u(k) = \pi(x(k))$. For every $i \in \{1, \ldots, 10\}$,

$$x_i(k+1) = x_i(k) + 0.05\big(1 - x_i(k)^2\big)\left[-\operatorname{arctanh}\big(x_i(k)\big) + \sum_{j=1}^{10} W_{ij}x_j(k) + \pi_i(x(k))\right].$$

where the symmetric weight matrix is

$$W = \begin{pmatrix} -\frac{3}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & -\frac{3}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & -\frac{3}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & -\frac{3}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{2} & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{2} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{2} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{2} & \frac{1}{4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & -\frac{3}{2} \end{pmatrix}.$$

- Domain set: $\mathcal{X} = [-0.9, 0.9]^{10}$.
- $\mathcal{X}_U = \mathbb{R}^{10} \setminus \mathcal{X}$.
- Control constraint: $\pi(x) \in [-2, 2]$.
- Verified type: forward invariance verification

Problem. 4 :

System 102 (New-8D Chain Safety).

The discrete-time state update is:

$$\mathbf{x}(k+1) = \begin{pmatrix} x_1(k) + \frac{1}{200}\left(-\frac{x_1(k)}{1+x_1(k)^2} - 4x_2(k)\right) \\ x_2(k) + \frac{1}{200}\left(\frac{4x_1(k)}{1+x_1(k)^2} - x_2(k) - \frac{3x_3(k)}{1+x_3(k)^2}\right) \\ x_3(k) + \frac{1}{200}\left(3x_2(k) - \frac{x_3(k)}{1+x_3(k)^2} - 5x_4(k)\right) \\ x_4(k) + \frac{1}{200}\left(\frac{5x_3(k)}{1+x_3(k)^2} - x_4(k) - \frac{2x_5(k)}{1+x_5(k)^2}\right) \\ x_5(k) + \frac{1}{200}\left(2x_4(k) - \frac{x_5(k)}{1+x_5(k)^2} - 4x_6(k)\right) \\ x_6(k) + \frac{1}{200}\left(\frac{4x_5(k)}{1+x_5(k)^2} - x_6(k) - \frac{3x_7(k)}{1+x_7(k)^2}\right) \\ x_7(k) + \frac{1}{200}\left(3x_6(k) - \frac{x_7(k)}{1+x_7(k)^2} - 6x_8(k)\right) \\ x_8(k) + \frac{1}{200}\left(\frac{6x_7(k)}{1+x_7(k)^2} - x_8(k)\right) \end{pmatrix}.$$

- $\mathcal{X} = [-1, 1]^8$.
- $\mathcal{X}_I = [-\frac{3}{20}, \frac{3}{20}]^8$.
- $\mathcal{X}_U = [\frac{3}{4}, \frac{9}{10}] \times [-\frac{1}{10}, \frac{1}{10}]^7$.
- Task type: safety verification.

System 103 (New-6D Arctan Chain Safety).

The discrete-time state update is:

$$\mathbf{x}(k+1) = \begin{pmatrix} x_1(k) + \frac{1}{200}\left(-\arctan(x_1(k)) - 8x_2(k)\right) \\ x_2(k) + \frac{1}{200}\left(8\arctan(x_1(k)) - x_2(k) - 6\arctan(x_3(k))\right) \\ x_3(k) + \frac{1}{200}\left(6x_2(k) - \arctan(x_3(k)) - 10x_4(k)\right) \\ x_4(k) + \frac{1}{200}\left(10\arctan(x_3(k)) - x_4(k) - 4\arctan(x_5(k))\right) \\ x_5(k) + \frac{1}{200}\left(4x_4(k) - \arctan(x_5(k)) - 8x_6(k)\right) \\ x_6(k) + \frac{1}{200}\left(8\arctan(x_5(k)) - x_6(k)\right) \end{pmatrix}.$$

- $\mathcal{X} = [-1, 1]^6$.
- $\mathcal{X}_I = [-\frac{3}{20}, \frac{3}{20}]^6$.
- $\mathcal{X}_U = [\frac{3}{4}, \frac{9}{10}] \times [-\frac{1}{10}, \frac{1}{10}]^5$.
- Task type: safety verification.

System 104 (New-8D Parametric Nonlinear Safety).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} x_2 \\ -x_1 - 0.1\sin(\omega x_1) - x_2 + x_3 \\ -2x_3 + x_4 \\ -1.5x_4 + x_5 \\ -2x_5 + x_6 \\ -1.5x_6 + 0.2x_7 \\ -2x_7 + 0.2x_6 + 0.2x_8 - 0.1x_7\sin(x_8) \\ -1.5x_8 + 0.2x_7 - 0.1x_8\cos(x_7) \end{pmatrix}.$$

- $\mathcal{X} = [-1, 1]^8$.
- $\mathcal{X}_I = \left\{x \in \mathbb{R}^8 \mid \|x\|_2 \le 0.5\right\}$.
- $\mathcal{X}_U = \left\{x \in \mathbb{R}^8 \mid \left\|x - (0, 0.8, 0, 0, 0, 0, 0, 0)^\mathsf{T}\right\|_2 \le 0.1\right\}$.
- $\omega \in [0.5, 2]$ is a fixed uncertain parameter.
- The barrier certificate is state-only and uniform for every fixed $\omega \in [0.5, 2]$.
- Task type: robust safety verification.

System 105 (New-5D Arctangent Perturbation Safety).

The vector flow field is:

$$\dot{\mathbf{x}} = \begin{pmatrix} x_2 + \frac{1}{2}x_5 \\ -x_1 - \omega x_2 + 2x_4 \\ x_4 \\ -2x_3 - \frac{1}{8}\arctan(x_3) - \frac{3}{2}x_4 - 2x_2 + x_5 \\ -\frac{1}{2}x_1 - x_4 - x_5 - \frac{1}{8}\arctan(x_5) \end{pmatrix}.$$

- $\mathcal{X} = [-1, 1]^5$.
- $\mathcal{X}_I = \left\{x \in \mathbb{R}^5 \mid \|x\|_2 \le \frac{1}{2}\right\}$.

- $\mathcal{X}_U = \left\{ x \in \mathbb{R}^5 \mid \left\| x - \left(0, 0, \frac{4}{5}, 0, 0\right)^\mathsf{T} \right\|_2 \leq \frac{1}{10} \right\}$.
- $\omega \in [1, 2]$.
- Task type: robust safety verification.