Carnegie Mellon University UNIVERSITY OF Southampton

Pegasus: a framework for sound continuous invariant generation

Andrew Sogokon[2,1], Stefan Mitsch[1], Yong Kiam Tan[1], Katherine Cordwell[1], and André Platzer[1]

[1]Carnegie Mellon University, USA
[2]University of Southampton, UK

FM 2019, 3rd World Congress on Formal Methods, Porto

October 20, 2019

Introduction

What this talk is about

Theorem proving in cyber-physical systems (CPS).

Why? **Fully rigorous proofs of correctness**.

Important for **safety-critical** embedded systems.

Introduction

What this talk is about

Theorem proving in cyber-physical systems (CPS).

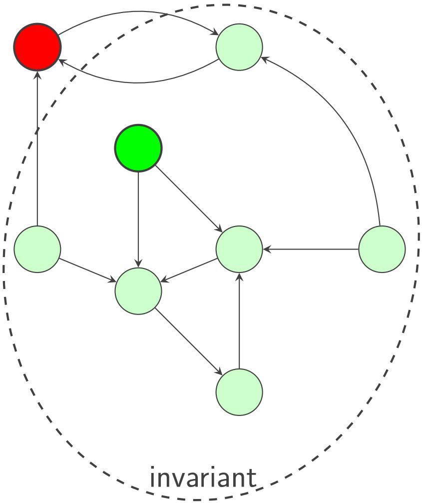
Why? **Fully rigorous proofs of correctness**.

Important for **safety-critical** embedded systems.

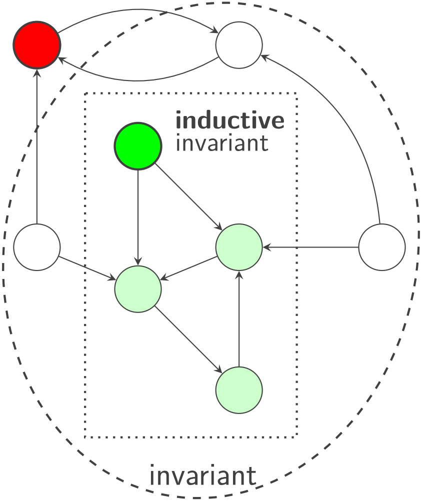
Problem: Theorem proving in CPS is **not fully automatic**.

Safety verification relies on finding the right **invariants**.

Invariants in verification



Invariants in verification



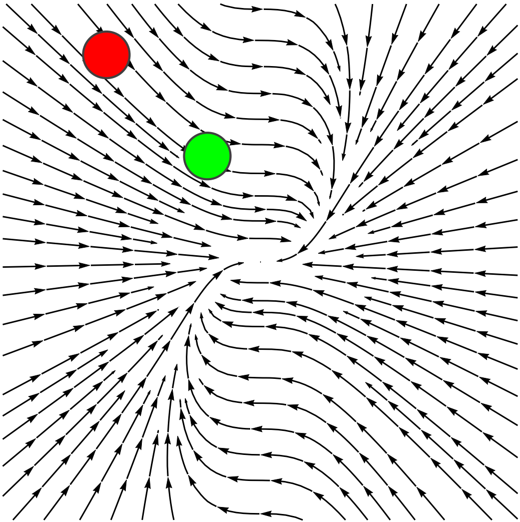
Continuous invariants

ODEs:

$\vec{x}' = f(\vec{x})$

$\vec{x} \in \mathbb{R}^n$

$\text{Init} \subseteq \mathbb{R}^n$



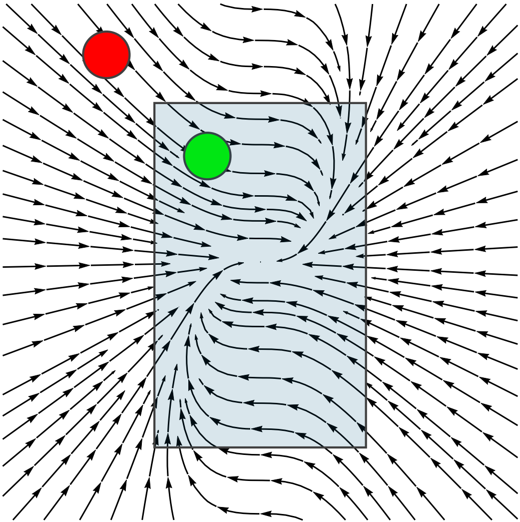
Continuous invariants

ODEs:

$\vec{x}' = f(\vec{x})$

$\vec{x} \in \mathbb{R}^n$

$\mathsf{Init} \subseteq \mathbb{R}^n$

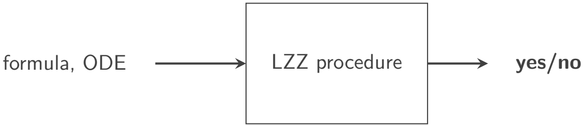


Checking continuous invariants

Checking whether a formula defines a continuous (inductive) invariant is **decidable** (Liu, Zhan & Zhao, EMSOFT 2011).

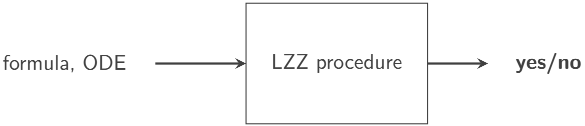
Checking continuous invariants

Checking whether a formula defines a continuous (inductive) invariant is **decidable** (Liu, Zhan & Zhao, EMSOFT 2011).



Checking continuous invariants

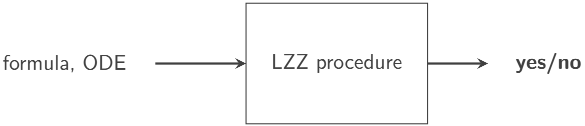
Checking whether a formula defines a continuous (inductive) invariant is **decidable** (Liu, Zhan & Zhao, EMSOFT 2011).



A **complete axiomatization** of continuous invariants in differential dynamic logic dL (Platzer & Tan, LICS 2018).

Checking continuous invariants

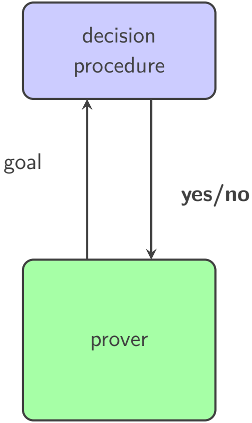
Checking whether a formula defines a continuous (inductive) invariant is **decidable** (Liu, Zhan & Zhao, EMSOFT 2011).



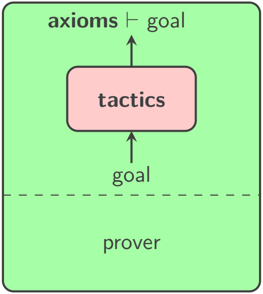
A **complete axiomatization** of continuous invariants in differential dynamic logic dL (Platzer & Tan, LICS 2018).

Handling decidable problems

Design choices in proof assistants



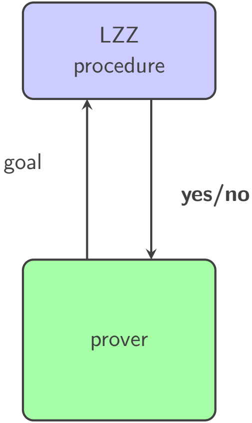
Using external oracles



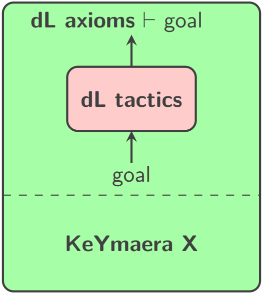
Formal proof using tactics

Handling invariants

Design choices in proof assistants



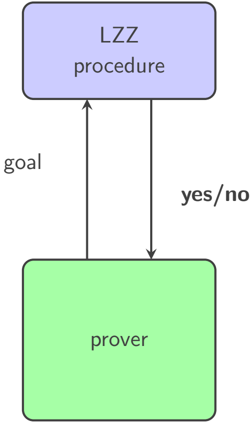
“PVS-style”



LCF-style

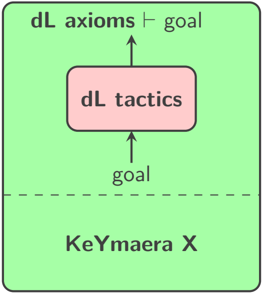
Handling invariants

Design choices in proof assistants



"PVS-style"

Less soundness-critical code



LCF-style

Generating continuous invariants

Carnegie Mellon University University of Southampton

Excellent progress made this decade on the **invariant checking problem**.

$$\{\mathsf{inv}\}\ ODE\ \{\mathsf{inv}\} \qquad (\text{in dL} \quad \mathsf{inv} \rightarrow [ODE]\ \mathsf{inv})$$

Generating continuous invariants

Excellent progress made this decade on the **invariant checking problem**.

$$\{\mathsf{inv}\}\ ODE\ \{\mathsf{inv}\} \qquad (\text{in dL} \quad \mathsf{inv} \rightarrow [ODE]\ \mathsf{inv})$$

The **invariant generation problem** is much more difficult.

$$\{\mathsf{pre}\}\ ODE\ \{\mathsf{post}\} \qquad (\text{in dL} \quad \mathsf{pre} \rightarrow [ODE]\ \mathsf{post})$$

Generating continuous invariants

Excellent progress made this decade on the **invariant checking problem**.

$$\{\text{inv}\}\ ODE\ \{\text{inv}\} \qquad (\text{in dL} \quad \text{inv} \rightarrow [ODE]\ \text{inv})$$

The **invariant generation problem** is much more difficult.

$$\{\text{pre}\}\ ODE\ \{\text{post}\} \qquad (\text{in dL} \quad \text{pre} \rightarrow [ODE]\ \text{post})$$

$$\frac{\text{pre} \rightarrow \text{inv} \qquad \text{inv} \rightarrow [ODE]\ \text{inv} \qquad \text{inv} \rightarrow \text{post}}{\text{pre} \rightarrow [ODE]\ \text{post}}$$

Generating continuous invariants

Excellent progress made this decade on the **invariant checking problem**.

$$\{\text{inv}\}\ ODE\ \{\text{inv}\} \qquad (\text{in dL} \quad \text{inv} \rightarrow [ODE]\ \text{inv})$$

The **invariant generation problem** is much more difficult.

$$\{\text{pre}\}\ ODE\ \{\text{post}\} \qquad (\text{in dL} \quad \text{pre} \rightarrow [ODE]\ \text{post})$$

$$\frac{\text{pre} \rightarrow \text{inv} \qquad \text{inv} \rightarrow [ODE]\ \text{inv} \qquad \text{inv} \rightarrow \text{post}}{\text{pre} \rightarrow [ODE]\ \text{post}}$$

Practical bottleneck for proof automation.

Generating continuous invariants

Carnegie Mellon University Southampton

In theory, we can search for invariants using **template formulas**:

$$a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 < 0 \wedge b_0 + b_1 x + b_2 y \geq 0$$

Generating continuous invariants

In theory, we can search for invariants using **template formulas**:

$$a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 < 0 \land b_0 + b_1 x + b_2 y \geq 0$$

Searching for the coefficients using algorithms from **real algebraic geometry** (e.g. CAD).

Generating continuous invariants

Carnegie Mellon University   UNIVERSITY OF Southampton

In theory, we can search for invariants using **template formulas**:

$$a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 < 0 \land b_0 + b_1 x + b_2 y \geq 0$$

Searching for the coefficients using algorithms from **real algebraic geometry** (e.g. CAD). **(However, this is hardly practical)*

Doubly-exponential time complexity in the number of variables (here the number of *coefficients*).

Generating continuous invariants

Carnegie Mellon University   UNIVERSITY OF Southampton

In theory, we can search for invariants using **template formulas**:

$$a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 < 0 \wedge b_0 + b_1 x + b_2 y \geq 0$$

Searching for the coefficients using algorithms from **real algebraic geometry** (e.g. CAD). **(However, this is hardly practical)*

Doubly-exponential time complexity in the number of variables (here the number of *coefficients*).

More practical alternatives are needed.

Generating continuous invariants

More practical methods for invariant generation exist.

These are

- more specialized,
- incomplete,
- have different strengths and limitations,
- create a wide spectrum for what can be tried.

Generating continuous invariants

More practical methods for invariant generation exist.

These are

- more specialized,
- incomplete,
- have different strengths and limitations,
- create a wide spectrum for what can be tried.

Challenge:

- build a **system for navigating this spectrum**,
- use it to improve proof automation in KeYmaera X.

Continuous invariant generator

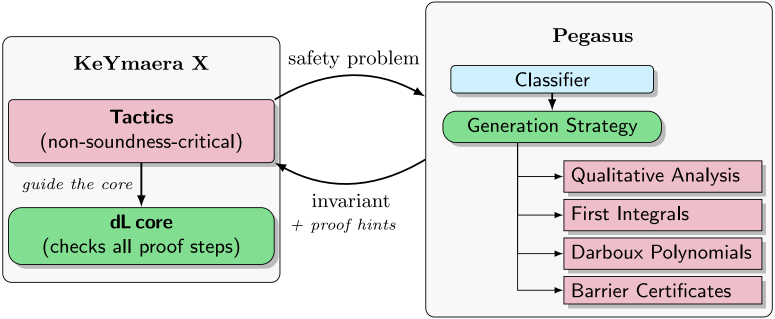
Pegasus is an automatic continuous invariant generator.

http://pegasus.keymaeraX.org

As of version 1.0, Pegasus (implemented in Wolfram Language) has

- a simple continuous safety verification problem classifier,
- implementation of invariant generation methods,
- a strategy for combining invariant generation methods,
- proof hints for KeYmaera X.

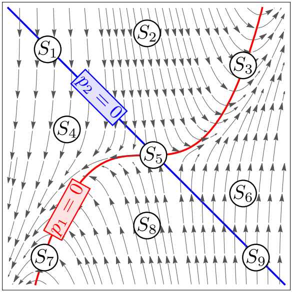
Sound integration architecture

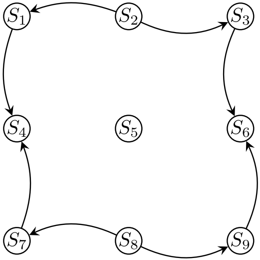


Discrete abstraction

Partition $\mathbb{R}^n$ into discrete states $S_1, \ldots, S_k$ defined by some predicates.

Compute the discrete transition relation.



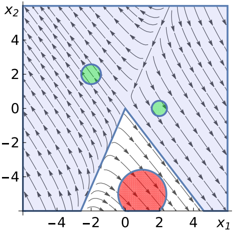


Qualitative analysis

In essence: discrete abstraction using information in the problem.

Some sources of predicates:

- right-hand sides of ODEs, their factors, etc.
- functions defining the pre/postcondition
- physically meaningful quantities (e.g. divergence of the vector field)



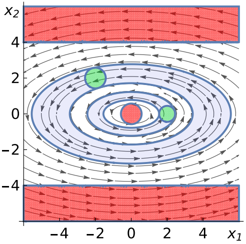
First integrals

and Darboux polynomials

Conserved quantities in the continuous system.

Functions p such that $p' = 0$ (i.e. the rate of change of p w.r.t. f is 0).

Searching for **polynomial** first integrals (of bounded degree) can be done using linear algebra.



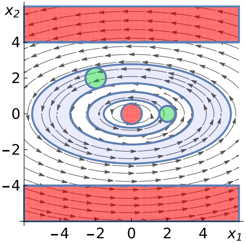
First integrals

and Darboux polynomials

Conserved quantities in the continuous system.

Functions p such that $p' = 0$ (i.e. the rate of change of p w.r.t. f is 0).

Searching for **polynomial** first integrals (of bounded degree) can be done using linear algebra.

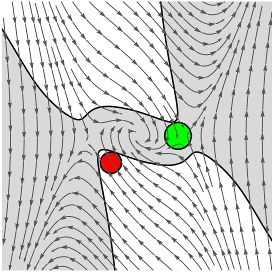


Darboux polynomials: $p' = \alpha p$, where α is a polynomial.

Barrier certificates

Main idea: find a continuous invariant $p \leq 0$ using

- **differential inequalities**, e.g. $p' \leq 0$, $p' \leq \lambda p$ ($\lambda \in \mathbb{R}$), and
- **sum-of-squares decomposition** (via semidefinite programming).

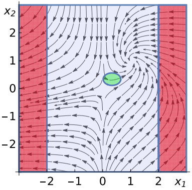


First described by Prajna and Jadbabaie (HSCC 2004).
Generalizes to **vector barrier certificates** (our work, FM 2018).

Differential saturation

A strategy for combining invariant generation methods.

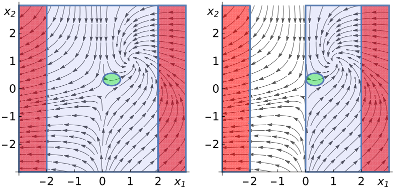
Iteratively refine the invariant using available methods.



Differential saturation

A strategy for combining invariant generation methods.

Iteratively refine the invariant using available methods.

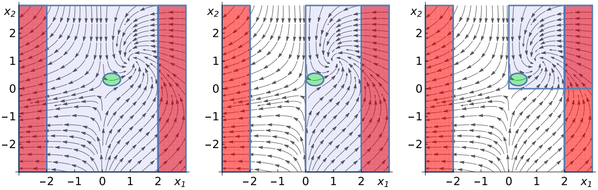


- Refinement 1 (using a Darboux polynomial)

Differential saturation

A strategy for combining invariant generation methods.

Iteratively refine the invariant using available methods.



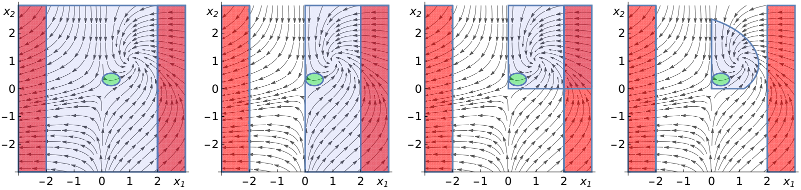
- Refinement 1 (using a Darboux polynomial)
- Refinement 2 (using Qualitative analysis)

Differential saturation

Carnegie Mellon University Southampton

A strategy for combining invariant generation methods.

Iteratively refine the invariant using available methods.

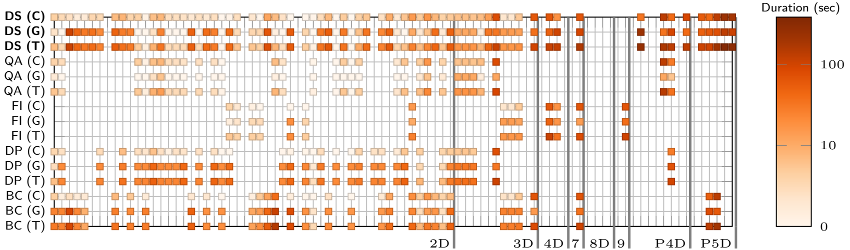


- Refinement 1 (using a Darboux polynomial)
- Refinement 2 (using Qualitative analysis)
- Refinement 3 (using a barrier certificate)

Some results

Non-linear systems

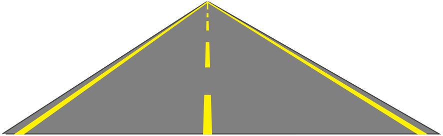
- 90 benchmark safety verification problems from the literature.
- 71 problem could be solved by the combined strategy.



Non-linear problems (dimension: 2D-9D, followed by 4D and 5D product systems)

- A few problems were **only** solved by the combined strategy (no individual method succeeded by itself).

Conclusion & future outlook



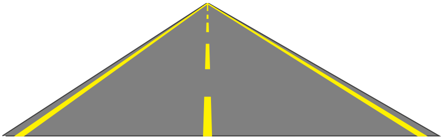
The results we observe are thus far very encouraging.

- Many more invariant generation methods to implement.
- Generation strategies that work solely in tractable theories.
- Larger corpus of continuous verification problems needed.

Goal: to make hybrid systems theorem proving more or less **automatic**.

Conclusion & future outlook

Carnegie Mellon University UNIVERSITY OF Southampton



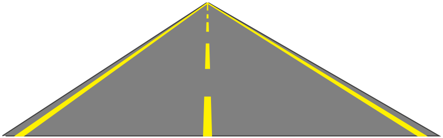
The results we observe are thus far very encouraging.

- Many more invariant generation methods to implement.
- Generation strategies that work solely in tractable theories.
- Larger corpus of continuous verification problems needed.

Goal: to make hybrid systems theorem proving more or less **automatic**.

The next ~~30~~ 10 years?

Conclusion & future outlook



The results we observe are thus far very encouraging.

- Many more invariant generation methods to implement.
- Generation strategies that work solely in tractable theories.
- Larger corpus of continuous verification problems needed.

Goal: to make hybrid systems theorem proving more or less **automatic**.

The next ~~30~~ 10 years?

http://pegasus.keymaeraX.org