A method for invariant generation for polynomial continuous systems

Andrew Sogokon [1] Khalil Ghorbal [2] **Paul B. Jackson** [1]
André Platzer [2]

[1]Laboratory for Foundations of Computer Science, University of Edinburgh

[2]Computer Science Department, Carnegie Mellon University

VMCAI'16, St. Petersburg, Florida
18th January, 2018

Continuous systems

Continuous systems describe the continuous state evolution inside operating modes in *hybrid systems* (HS).

They are given by systems of *ordinary differential equations* (ODEs) defined on $\mathbb{R}^n$ and evolving under constraints, i.e.

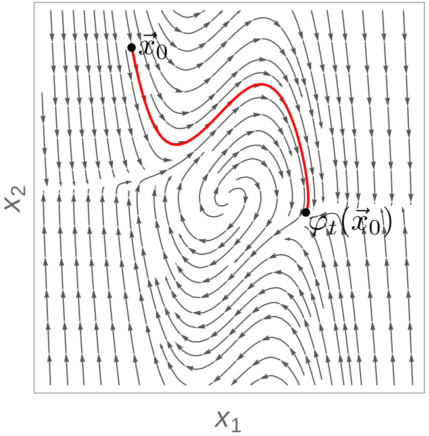
$$\begin{aligned} \dot{x}_1 &= f_1(\vec{x}), \\ \dot{x}_2 &= f_2(\vec{x}), \\ &\vdots \\ \dot{x}_n &= f_n(\vec{x}), \\ \vec{x} \in H &\subseteq \mathbb{R}^n. \end{aligned}$$

We write this more concisely in vector form as $\dot{\vec{x}} = f(\vec{x}) \mathbin{\&} H$.

When the system is initialized in a state $\vec{x}_0 \in H$, the (unique) *solution* gives the state of the system at time $t \in \mathbb{R}$, which is written as $\varphi_t(\vec{x}_0) \in \mathbb{R}^n$.

Continuous systems (example)

Consider the Van der Pol system.

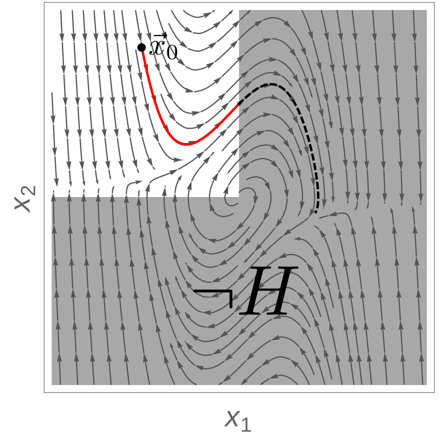


$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = \left(1 - x_1^2\right) x_2 - x_1,$$

Continuous systems (example)

Consider the Van der Pol system.

One may impose some evolution constraint, e.g. the fourth quadrant of $\mathbb{R}^2$.



$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = \left(1 - x_1^2\right) x_2 - x_1,$$
$$H \equiv x_1 \leq 0 \wedge x_2 \geq 0.$$

Safety assertions

Interested in verifying safety assertions of form

> *Starting from ψ, by continuously evolving inside H for any amount of time, the system remains within the safe region ϕ.*

where
$\psi,\ \phi$ are quantifier-free formulas of real arithmetic describing regions of $\mathbb{R}^n$
$\dot{\vec{x}} = f(\vec{x})\ \&\ H$ is a system of ODEs under some evolution constraint H.

In FOL:

$$\forall\, \vec{x} \in \psi.\ \forall\, t \geq 0.\ (\forall\, \tau \in [0, t].\ \varphi_\tau(\vec{x}) \in H) \rightarrow\ \varphi_t(\vec{x}) \in \phi$$

In differential dynamic logic ($d\mathcal{L}$):

$$\psi \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\ \phi$$

As a (continuous) Hoare triple:

$$\{\psi\}\ \dot{\vec{x}} = f(\vec{x})\ \&\ H\ \{\phi\}$$

Proof method for safety verification (continuous invariants)

Definition (Continuous invariant)

A set $I \subseteq \mathbb{R}^n$ is a ***continuous invariant*** *for $\dot{\vec{x}} = f(\vec{x})$ & H if and only if*

$$\forall\, \vec{x} \in I.\, \forall\, t \geq 0.\, (\forall\, \tau \in [0, t].\, \varphi_\tau(\vec{x}) \in H) \rightarrow \varphi_t(\vec{x}) \in I.$$

Continuous invariants are used in safety verification

$$(\text{Safety})\,\frac{\vdash \psi \rightarrow I \qquad \vdash I \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\, I \qquad \vdash I \rightarrow \phi}{\vdash \psi \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\, \phi}$$

Caveat: φ_t is often impossible to obtain in closed-form. Instead, it is possible to work with the ODEs *directly*, i.e. only work with f instead of φ_t.

It is **decidable to check** whether the invariance assertion below is true

$$I \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\, I$$

if $I,\ H$ are semi-algebraic and f polynomial (Liu, Zhan, Zhao, EMSOFT'11).

Hints at why LZZ works

- An induction principle for R

$$\frac{\begin{array}{l}1.\ \forall t > 0.\ (\exists e \in (0 \ldots t].\ \forall t' \in (t - e \ldots t).\ P(t')) \rightarrow P(t)\\ 2.\ \forall t \geq 0.\ P(t) \rightarrow \exists e > 0.\ \forall t' \in (t \ldots t + e).\ P(t')\\ 3.\ P(0)\end{array}}{\forall t \geq 0.\ P(t)}$$

- Want to show $p \geq 0 \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\, p \geq 0$
  Sufficient to show that when $p = 0$ at time t, p is non-negative in some finite open interval after t.
- Consider Taylor series for p:

$$p(t) = p(0) + p'(0)t + \frac{1}{2}p''(0)t^2 + \ldots$$

  and look either for all $p^i(0)$ derivatives zero or first non-zero derivative positive.
- Only have to check finite many derivatives by an ascending chain condition argument

Invariant generation

All premises in the proof rule (Safety) are decidable if sets $\psi,\ \phi,\ H,\ I$ are semi-algebraic and f polynomial.

$$(\text{Safety})\,\frac{\vdash \psi \to I \qquad \vdash I \to [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\, I \qquad \vdash I \to \phi}{\vdash \psi \to [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\, \phi}$$

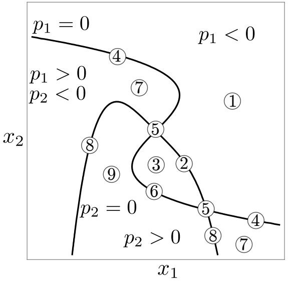
Problem: How can we find an appropriate continuous invariant I?

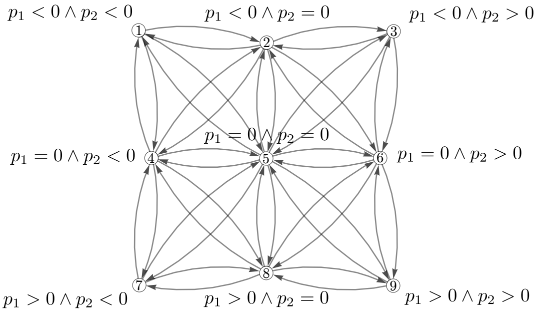
- Semi-algebraic continuous invariant **templates** require fresh variables (*not practical due to the complexity of real quantifier elimination*).
- Safety verification by computing **discrete abstractions** of continuous systems (implicitly) works with continuous invariants and does not require fresh variables . . .

Discrete abstraction

Given a continuous system and a set of polynomials A, e.g. $A = \{p_1, p_2\}$,

- partition the state space into regions where the polynomials do not change sign to obtain the **discrete states** (below, left), denoted S, and
- only connect **neighbouring** discrete states with discrete transitions to obtain a neighbouring transition relation $T_n \subset S \times S$ (below, right).





Exact discrete abstraction

Construct using polynomials $p_1, \dots p_m$

States $\mathbf{s}_i$ are *non-empty* intersections

$$H \cap p_1 \sim_1 0 \cap \cdots \cap p_m \sim_m 0$$

where $\sim_i \in \{<, =, >\}$ for $1 \leq i \leq m$.

Add transition $\mathbf{s}_i \longrightarrow \mathbf{s}_j$ if and only if the system may evolve continuously from $\mathbf{s}_i$ into $\mathbf{s}_j$ without leaving $\mathbf{s}_i \cup \mathbf{s}_j$.

$$\exists \vec{x}.\ \mathbf{s}_i \land \langle\ \dot{\vec{x}} = f(\vec{x})\ \&\ (\mathbf{s}_i \lor \mathbf{s}_j)\ \rangle\ \mathbf{s}_j$$

Or equivalently:

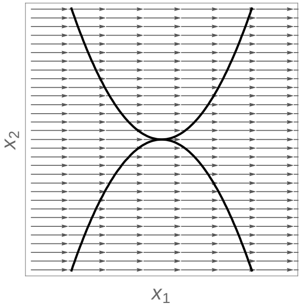
$$\neg(\mathbf{s}_i \to [\ \dot{\vec{x}} = f(\vec{x})\ \&\ (\mathbf{s}_i \lor \mathbf{s}_j)\ ]\ \mathbf{s}_i)$$

which is decidable by LZZ procedure.

Exact discrete abstraction (example)

Exact abstractions do not suffer from unsoundness and coarseness issues found in some other abstraction methods for non-linear systems.

For example, let the continuous system be given by $\dot{x}_1 = 1, \dot{x}_2 = 0$ and consider two polynomials for abstraction $A = \{x_1^2 + x_2, x_2 - x_1^2\}$.

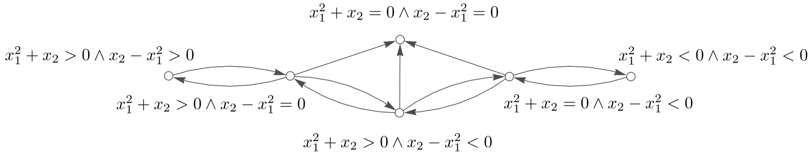


The intersecton of the two curves $x_1^2 + x_2 = 0 \wedge x_2 - x_1^2 = 0$ is precisely the point at the origin, $(0, 0)$. Clearly not an invariant set...

Laboratory for Foundations of Computer Science

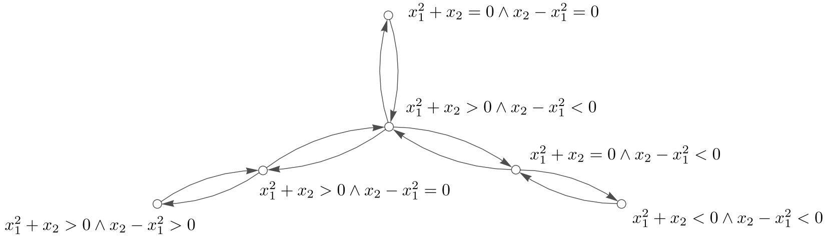
THE UNIVERSITY of EDINBURGH informatics

Exact discrete abstraction (example)



$x_1^2 + x_2 = 0 \wedge x_2 - x_1^2 = 0$
$x_1^2 + x_2 > 0 \wedge x_2 - x_1^2 > 0$
$x_1^2 + x_2 < 0 \wedge x_2 - x_1^2 < 0$
$x_1^2 + x_2 > 0 \wedge x_2 - x_1^2 = 0$
$x_1^2 + x_2 = 0 \wedge x_2 - x_1^2 < 0$
$x_1^2 + x_2 > 0 \wedge x_2 - x_1^2 < 0$

Figure: Abstraction $(S, T_\sim)$ using method by Tiwari, FMSD 2008.



$x_1^2 + x_2 = 0 \wedge x_2 - x_1^2 = 0$
$x_1^2 + x_2 > 0 \wedge x_2 - x_1^2 < 0$
$x_1^2 + x_2 = 0 \wedge x_2 - x_1^2 < 0$
$x_1^2 + x_2 > 0 \wedge x_2 - x_1^2 = 0$
$x_1^2 + x_2 > 0 \wedge x_2 - x_1^2 > 0$
$x_1^2 + x_2 < 0 \wedge x_2 - x_1^2 < 0$

Figure: Exact abstraction (S, T).

Invariant generation using exact abstraction

Given a continuous system $\dot{\vec{x}} = f(\vec{x})\ \&\ H$, an initial set ψ, and a set of polynomials A, want to compute the strongest invariant I in the abstraction generated by A.

Basic

- Establish all non-empty states $\mathbf{s}_i$.
- Construct full transition relation
- Initial states A_0 are all those with non-empty intersection with ψ
- Compute invariant I as all abstract states reachable from A_0.

LazyReach

- As Approach 1, but explore state space from A_0 lazily.

State space explosion problem

The main problem with using discrete abstraction for invariant generation is **scalability**. The number of discrete states and transitions grows *exponentially* with the number of polynomials $|A|$.

What can be done?

Given: continuous system $\dot{\vec{x}} = f(\vec{x})\ \&\ H$, an initial set ψ, and a set of polynomials A.

Challenge: refine the constraint H in a way that some polynomials can be *removed* from A *without making the abstraction any coarser*.

We address this problem by employing sound *proof rules* for safety verification.

Proof rules. Differential Weakening (Platzer, 2008)

The DW rule says that it is always sound to conclude system safety if the evolution constraint H contains no unsafe states.

$$(DW)\,\frac{H \to \phi}{\psi \to [\dot{\vec{x}} = f(\vec{x}) \,\&\, H]\, \phi}$$

Obvious, since in this case every reachable state is safe.

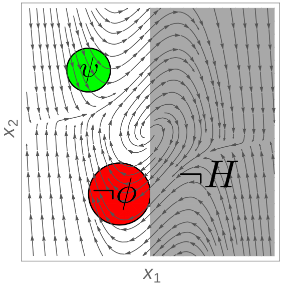
If this rule applies, then one may forget about computing an abstraction and simply *return the constraint H as the continuous invariant*.

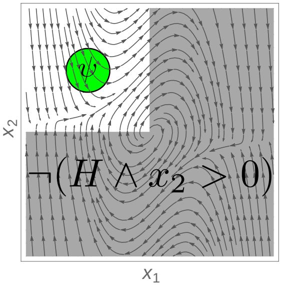
Proof rules. Differential Cut (Platzer, 2008)

The DC rule says that it is always sound to refine the evolution domain H by some continuous invariant F, *provided that it includes the initial set* ψ, i.e.

$$(DC)\,\frac{\psi \to [\dot{\vec{x}} = f(\vec{x})\ \&\ H]F \qquad \psi \to [\dot{\vec{x}} = f(\vec{x})\ \&\ H \wedge F]\,\phi}{\psi \to [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\,\phi}$$

For example, consider the Van der Pol system below where $H \equiv x_1 \geq 0$. Apply DC with $F \equiv x_2 > 0$ (below, right) and then prove safety using DW.





N.B. the polynomial x_2 is *sign-invariant* in the refined constraint. If $x_2 \in A$, then it can be removed because x_2 cannot partition the refined constraint.

Differential divide and conquer (DDC)

Differential divide and conquer (DDC) works to split the safety assertion into 3 independent safety assertions about smaller systems. DDC requires a polynomial p in order to partition the constraint H *and the initial set* ψ.

Proposition

The proof rule DDC given below (with five premises) is sound.

$$\begin{array}{c} p = 0 \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\, p = 0 \\ p = 0 \rightarrow [\dot{\vec{x}} = -f(\vec{x})\ \&\ H]\, p = 0 \end{array}$$

$$(DDC)\,\frac{\begin{array}{c} \psi \wedge p > 0 \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H \wedge p > 0]\ \phi \\ \psi \wedge p = 0 \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H \wedge p = 0]\ \phi \\ \psi \wedge p < 0 \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H \wedge p < 0]\ \phi \end{array}}{\psi \rightarrow [\dot{\vec{x}} = f(\vec{x})\ \&\ H]\ \phi}$$

The set of real zeros of p, i.e. $p = 0$ is required to be a continuous invariant in both positive and negative time. This means that there can be no continuous flow between the three regions $p > 0$, $p = 0$ and $p < 0$ within the system.

Invariant generation algorithms DWC and DWCL

DWC

1. Apply DW, DC and DDC exhaustively.
2. This results in two collections C_1 and C_2 of non-overlapping subsets of the original H, where:
   - each $H' \in C_1$ is from a DW application and satisfies $H' \subseteq \phi$,
   - each member of C_2 is from a DC or DDC application. Nothing further known about them.
3. Return formula for $C_1 \cup C_2$

DWCL

- As DWC, but in step 3, further cut down each element of C_1 by applying *LazyReach* using polynomials not removed by DC or DDC rules.

Invariant generation example (DWCL)

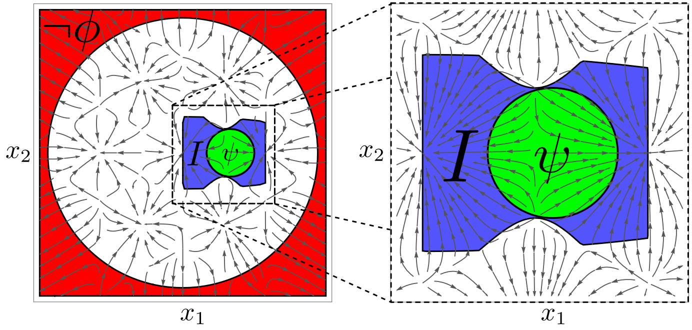
$$\begin{aligned}
\dot{x}_1 &= 2x_1\left(x_1^2-3\right)\left(4x_1^2-3\right)\left(x_1^2+21x_2^2-12\right),\\
\dot{x}_2 &= x_2\Big(35x_1^6+105x_2^2x_1^4-315x_1^4-63x_2^4x_1^2+378x_1^2+27x_2^6-189x_2^4+378x_2^2-216\Big),\\
H &= \mathbb{R}^2,\\
\psi &\equiv (x_1-1)^2+x_2^2<\frac{1}{4},\\
\phi &\equiv x_1^2+x_2^2<8.
\end{aligned}$$

The algorithm DWCL generates the following continuous invariant in under 2 minutes (using 7 polynomials from the post-condition and the factors of the RHS of the ODEs) :

$$\begin{aligned}
&\Bigg(\bigg(35x_1^6+105\left(x_2^2-3\right)x_1^4+27\left(x_2^6-7x_2^4+14x_2^2-8\right)<63x_1^2\left(x_2^4-6\right)\vee x_2=0\bigg)\\
&\wedge 4x_1^2=3\wedge x_1>0\bigg)\vee\bigg(x_2=0\wedge\left(0<x_1<\frac{\sqrt{3}}{2}\vee\frac{\sqrt{3}}{2}<x_1<\sqrt{3}\right)\bigg)\\
&\vee\bigg(35x_1^6+105\left(x_2^2-3\right)x_1^4+27\left(x_2^6-7x_2^4+14x_2^2-8\right)<63x_1^2\left(x_2^4-6\right)\\
&\wedge x_1^2+21x_2^2<12\wedge\left(0<x_1<\frac{\sqrt{3}}{2}\vee\left(2x_1>\sqrt{3}\wedge x_1^2<3\wedge x_2\neq 0\right)\right)\bigg)\Bigg).
\end{aligned}$$

Invariant generation example (DWCL)

$$\dot{x}_1 = 2x_1\left(x_1^2-3\right)\left(4x_1^2-3\right)\left(x_1^2+21x_2^2-12\right),$$
$$\dot{x}_2 = x_2\Big(35x_1^6+105x_2^2x_1^4-315x_1^4-63x_2^4x_1^2+378x_1^2+27x_2^6-189x_2^4+378x_2^2-216\Big),$$
$$H = \mathbb{R}^2,$$
$$\psi \equiv (x_1-1)^2 + x_2^2 < \frac{1}{4},$$
$$\phi \equiv x_1^2 + x_2^2 < 8.$$



Benchmarks

We have collected a set of 100 safety verification problems featuring mainly non-linear polynomial continuous systems. Most of the problems are planar or 3-dimensionsional, but a few problems have up to 5 variables.

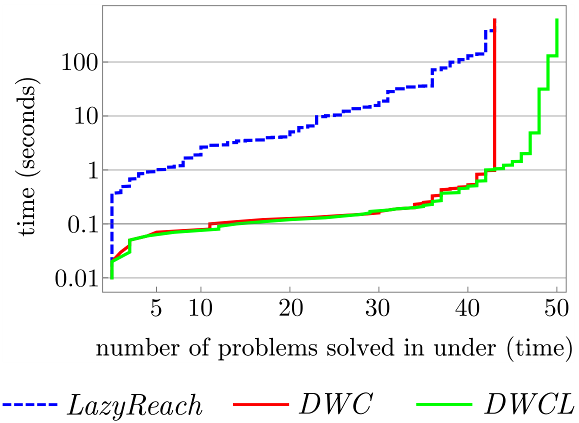
ODEs mainly originate from examples found in textbooks on dynamical systems, papers on the qualitative theory of ODEs and safety verification of hybrid systems.

We have compared the performance of our algorithms using 4 different sources of polynomials A for the abstraction:

1. Irreducible factors of the polynomials in the RHS of the ODEs and the post-condition ϕ,
2. As in (1), together with their derivatives (doubling the size of A),
3. As in (1), together with polynomials whose real roots define algebraic invariants for the system at hand (generated using the method in Ghorbal & Platzer, TACAS'14),
4. As in (2), together with polynomials that define algebraic invariants.

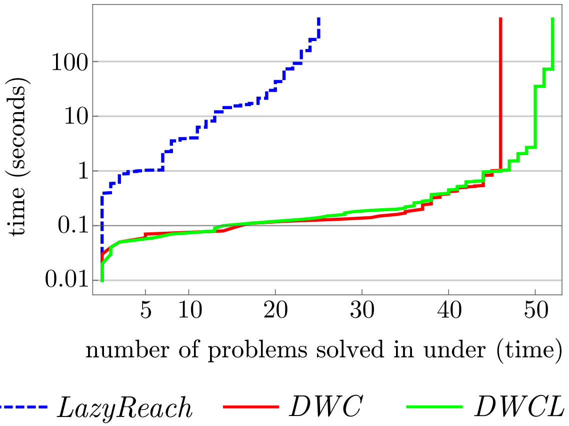
Benchmarks

Factors of the polynomials in the RHS of ODEs and the post-condition.



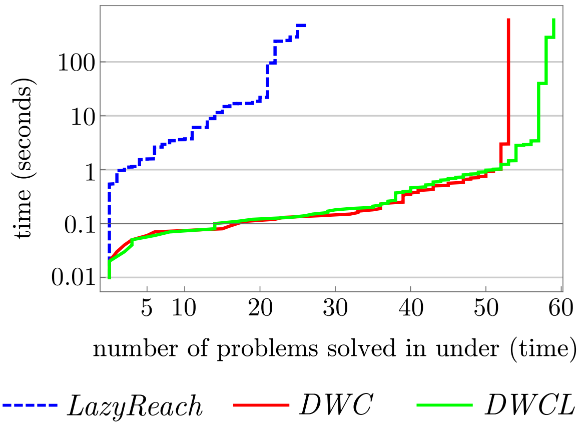
Benchmarks

Factors of the polynomials in the RHS of ODEs and the post-condition + their derivatives.



Benchmarks

Factors of the polynomials in the RHS of ODEs and the post-condition + algebraic invariants (Ghorbal & Platzer, TACAS'14) + their derivatives.



Observations & Challenges

Observations:

- 59 out of 100 safety verification problems could be solved.
- DWC and DWCL scale far better than simply extracting reachable sets from abstractions.
- Polynomials with invariant 0 sets (varieties) are often useful for abstraction. Zaki, Tahar & Bois, SMCAD'06 previously used Darboux invariants.

Challenges:

- Selecting "good" polynomials for abstraction is difficult.
- Real arithmetic is expensive.

Conclusion

Exact abstractions of polynomial continuous systems.

- Intuitively simple once one has a decision procedure for checking semi-algebraic continuous invariants.
- Remove unsoundness and coarseness issues found in earlier methods.

Invariant generation method for polynomial continuous systems.

- Capable of extracting reachable sets of exact abstractions.
- Scalability gains due to the use of sound proof rules.
- Highly desirable in a theorem proving environment.

Future work:

- Implementation in an theorem prover for hybrid systems
- Applications to hybrid system verification

End.