HAL
open science

Non-linear Continuous Systems for Safety Verification (Benchmark Proposal)

Andrew Sogokon, Khalil Ghorbal, Taylor T. Johnson

To cite this version:

Andrew Sogokon, Khalil Ghorbal, Taylor T. Johnson. Non-linear Continuous Systems for Safety Verification (Benchmark Proposal). ARCH@CPSWeek 2016 - 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems, Apr 2016, Vienna, Austria. pp.42-51. hal-01660900

HAL Id: hal-01660900

https://hal.science/hal-01660900v1

Submitted on 11 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Non-linear Continuous Systems for Safety Verification (Benchmark Proposal)

Andrew Sogokon[1], Khalil Ghorbal[2], and Taylor T. Johnson[3]

[1] University of Texas at Arlington, Arlington, TX, USA
andrew.sogokon@uta.edu
[2] INRIA, Rennes, Brittany, France
khalil.ghorbal@inria.fr
[3] University of Texas at Arlington, Arlington, TX, USA
taylor.johnson@gmail.com

Abstract

Safety verification of hybrid dynamical systems relies crucially on the ability to reason about reachable sets of continuous systems whose evolution is governed by a system of ordinary differential equations (ODEs). Verification tools are often restricted to handling a particular class of continuous systems, such as e.g. differential equations with constant right-hand sides, or systems of affine ODEs. More recently, verification tools capable of working with *non-linear* differential equations have been developed. The behavior of non-linear systems is known to be in general extremely difficult to analyze because solutions are rarely available in closed-form. In order to assess the practical utility of the various verification tools working with non-linear ODEs it is very useful to maintain a set of verification problems. Similar efforts have been successful in other communities, such as automated theorem proving, SAT solving and numerical analysis, and have accelerated improvements in the tools and their underlying algorithms. We present a set of 65 safety verification problems featuring non-linear polynomial ODEs and for which we have proofs of safety. We discuss the various issues associated with benchmarking the currently available verification tools using these problems.

1 Introduction

For verifying safety properties of hybrid systems, it is crucial to have the means of reasoning about safety properties of purely continuous systems that determine state evolution inside the *operating modes*.

In computer science, emphasis has traditionally been placed on working with hybrid systems in which the continuous modes are governed by relatively simple ODEs. For instance, safety verification of systems with ODEs possessing constant right-hand sides and right-hand sides bounded within real intervals is aided by the fact that reachable sets of such continuous systems can be computed exactly. Progress has been made on verifying safety in systems with linear and affine continuous dynamics (with tools such as PHAVer [13] and SpaceEx [14]). This is a much more difficult problem, since reachable sets of linear ODEs cannot in general be phrased in a decidable theory, which is only known to be possible for some special classes of systems [22, 16, 18].

It is a well-known fact that non-linear ODEs can exhibit behaviour that is impossible under affine or linear dynamics [19]. Their expressive power allows for modelling very rich dynamic phenomena, but comes at the price of making the reachability analysis much more difficult. A major obstacle is the fact that solutions to non-linear ODEs cannot in general be obtained as closed-form expressions, i.e. finite expressions in terms of polynomials and elementary functions such as $\exp, \sin, \cos, \ln$, etc. Hybrid systems with non-linear ODEs are not at all uncommon in

control theory; this is especially true of the class of *piecewise-smooth systems* (sometimes called *variable structure systems*), which are used in the design of *sliding mode controllers* [10].

A number of tools and approaches have been developed that enable safety verification of non-linear systems (e.g. [7, 21, 36, 31, 30, 29, 24, 17, 15, 37]). The methods currently in existence differ in a number of aspects; for instance, the level of automation they provide, the generality of system and inputs specifications, etc. These important (and at times subtle) differences make the tools difficult to compare objectively. One approach to address the issue could be to push for a consensus in the community about a useful and fairly general class of systems of interest that we should all work on. However, any such enterprise would be necessarily artificial for the time being as there is no generally agreed-upon classification of differential equations. In this work, we rather advocate a pragmatic approach: that of creating a database of benchmarks that can be used for a comprehensive assessment of the existing and future verification tools. The hope would be to steer the research towards working with a growing set of examples that a variety of related communities care about. If such a set were available, a tool (or an approach) could easily be seen to be more powerful if it is able to handle (parse, verify, solve, etc.) a larger proportion of those examples. Determining which verification tool is "better" cannot be entirely objective as it would further need to take into account the tool's running time performance, memory requirements, level of automation, etc. However, we believe that the problem of comparing verification tools can, at least in part, be addressed by collecting verification benchmarks and converting them to a single standardized input format. While this effort is only a first step towards a more ambitious goal, we feel it is important to initiate the process of gathering interesting verification problems and making them available to the community.

Similar efforts have been successfully undertaken in fields such as automated theorem proving (e.g. the TPTP problem library [3]), SAT solving (where competitions, e.g. [1], have led to drastic improvements in the performance of SAT solvers in the last two decades) and numerical analysis [39], resulting in improved quality of the tools and their underlying algorithms.

Contributions

We (`I`) provide a set of 65 safety verification problems featuring non-linear systems, for all of which the safety property is known to hold. Further, we (`II`) discuss the current challenges in comparing verification tools working with non-linear continuous dynamics and (`III`) outline ideas for addressing some of these difficulties.

2 Benchmarks

We have collected a set of 65 safety verification problems featuring non-linear ODEs, which we have gathered from existing papers treating the problem of unbounded time safety verification [24, 9, 11, 37] and invariant generation for non-linear systems (e.g. [6]). The problems we have collected all share the property of having proofs of safety that were obtained using the methods presented in the pertinent papers (or having proofs that are immediate from the results described therein).

In general, in order to fully state a safety verification problem, one requires four pieces of information:

1. The system of ODEs, written using vector notation as $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, where $f : \mathbb{R}^n \to \mathbb{R}^n$.
2. The mode invariant, denoted $H \subseteq \mathbb{R}^n$, which defines the region where the system may evolve along the solution to the system of ODEs.

3. The set of initial states $X_0 \subseteq \mathbb{R}^n$.
4. The set of unsafe (or forbidden) states $X_u \subseteq \mathbb{R}^n$.

Remark Note that it is sufficient to consider autonomous ODEs, i.e. those in which the right-hand side does not depend explicitly on the independent time variable t, because one may always augment the system with $\dot{t} = 1$ and treat t as a state variable. Furthermore, in many cases it is also sufficient to only consider polynomial problems because it is often possible to re-cast safety verification problems with non-polynomial terms to problems only featuring polynomial functions (see e.g. [25, 28]).

The problem is to show that it is impossible for the system to evolve into a forbidden state $\boldsymbol{x}_u \in X_u$ from any initial state $\boldsymbol{x}_0 \in X_0$ by following the solution $\varphi_t(\boldsymbol{x}_0)$ to the system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ for any time while it remains within the evolution constraint H. Formally, this may be written down as

$$\forall t \geq 0.\ \forall \boldsymbol{x}_0 \in X_0.\ (\forall \tau \in [0, t].\ \varphi_\tau(\boldsymbol{x}_0) \in H) \rightarrow \varphi_t(\boldsymbol{x}_0) \notin X_u.$$

In bounded-time safety verification one is only interested in showing safety up to some finite time bound $T \geq 0$, i.e.

$$\forall t \in [0, T].\ \forall \boldsymbol{x}_0 \in X_0.\ (\forall \tau \in [0, t].\ \varphi_\tau(\boldsymbol{x}_0) \in H) \rightarrow \varphi_t(\boldsymbol{x}_0) \notin X_u.$$

Clearly, if the safety property holds for unbounded time, it is guaranteed for any finite time bound, but not conversely. Since all the problems we have gathered are non-linear and have proofs of unbounded-time safety, we may designate this class of problems `NONLIN-UNBOUND-TIME-SAFE` in order to distinguish it from other classes of problems that we may wish to add later on, such as e.g. provably safe linear systems, or provably unsafe systems, etc. In this section we will illustrate some of the safety verification problems featuring 2-dimensional ODEs. The full set of the 65 problems is available from http://verivital.com/hyst/benchmark-nonlinear/

Example 2.1 (Non-linear example [9])**.** Dai et al. in [9] studied safety verification using barrier certificates, illustrating their approach using the following system:

$$\dot{x} = 2x - xy,$$
$$\dot{y} = 2x^2 - y.$$

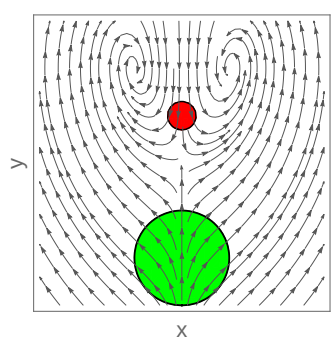

Figure 1: Non-linear system in the safety verification problem from [9].

The set of initial states is given by $x^2 + (y+2)^2 \leq 1$ and the set of unsafe states is $x^2 + (y-1)^2 \leq \frac{9}{100}$ (shown in green and red respectively in Fig. 1). The evolution constraint is taken to be the real plane $\mathbb{R}^2$.

Example 2.2 (FitzHugh-Nagumo system example [6])**.** Ben Sassi et al. [6] reported a method for generating polyhedral invariants for polynomial ODEs and applied it to the FitzHugh-Nagumo system:

$$\dot{x} = -\frac{x^3}{3} + x - y + \frac{7}{8},$$
$$\dot{y} = \frac{2}{25}\left(x - \frac{4y}{5} + \frac{7}{10}\right).$$

With the knowledge of the invariant, by considering initial states that lie inside the invariant,

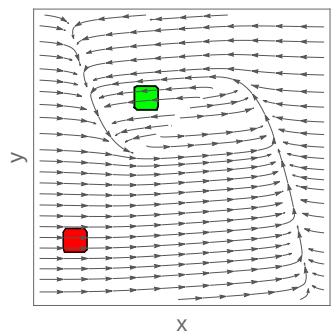

Figure 2: Safety verification in the FitzHugh-Nagumo system.

e.g. $-1 \leq x \leq -0.5 \ \wedge \ 1 \leq y \leq 1.5$ and letting $-2.5 \leq x \leq -2 \ \wedge \ -2 \leq y \leq -1.5$ represent the forbidden states, all of which lie entirely outside the invariant, one may conclude the safety property. Fig 2 shows the phase portrait along with the initial and the unsafe states (in green and red, respectively).

Example 2.3 ([37], ODE from [12], Ex. 10.15 (i))**.** In previous work [37], a non-linear ODE from a textbook on the qualitative theory of planar ODEs [12]

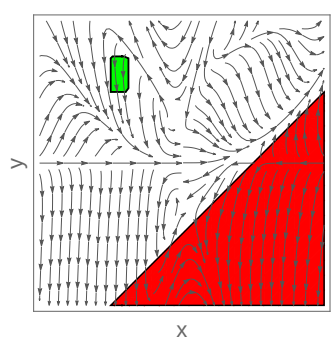

Figure 3: Safety verification problem from [37].

$$\dot{x} = -42x^7 + 68x^6y - 46x^5y + 258x^4y + 156x^3y + 50x^2y + 20xy^6 - 8y^7,$$
$$\dot{y} = y\left(1110x^6 - 220x^5y - 3182x^4y + 478x^3y^3 + 487x^2y^4 - 102xy^5 - 12y^6\right),$$

was used to create a safety verification problem where the initial states are given by $x > -1 \wedge x < -\frac{3}{4} \wedge y \leq \frac{3}{2} \wedge y \geq 1$ and the forbidden states satisfy the inequality $x > y + 1$ (shown respectively in green and red in Fig. 3).

2.1 Problem format

We have chosen to store our verification problems in a format used by the SpaceEx verification tool for hybrid systems [14]. While SpaceEx currently cannot work with non-linear differential equations, its input format is sufficiently simple and convenient. A given problem in this format is stored in two separate files

1. An `.xml` file storing the ODE $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and the mode invariant H of the system.
2. A `.cfg` file detailing the initial set X_0 and the set of forbidden states X_u.

For example, the verification problem described in Example 2.2, may be stored in the two files shown in Fig. 4 and Fig. 5.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<sspaceex xmlns="http://www-verimag.imag.fr/xml-namespaces/sspaceex" version="0.2"
    math="SpaceEx">
 <component id="fitzhugh_nagumo_ben_sassi_girard_2">
 <param name="x" type="real" local="false" d1="1" d2="1" dynamics="any"/>
 <param name="y" type="real" local="false" d1="1" d2="1" dynamics="any"/>
 <location id="1" name="p">
  <invariant>true</invariant>
  <flow>x'==7/8+x-x^3/3-y &amp; y'==(2*(7/10+x-(4*y)/5))/25</flow>
 </location>
 </component>
</sspaceex>
```

Figure 4: FitzHugh-Nagumo system dynamics, illustrated in Fig. 2.

```
system = fitzhugh_nagumo_ben_sassi_girard_2
initially = "-1<=x & x<=-0.5 & 1<=y & y<=1.5"
forbidden = "-2.5<=x & x<=-2 & -2<=y & y<=-1.5"
output-variables = x,y
scenario = stc
directions = box
set-aggregation = "none"
sampling-time = 0.5
flowpipe-tolerance = 0.25
time-horizon = 9
iter-max = 4
output-format = GEN
verbosity = m
output-error = 0.001
rel-err = 1.0e-12
abs-err = 1.0e-15
```

Figure 5: SpaceEx configuration file specifying the initial and forbidden states.

3 Challenges

In using any significantly broad set of verification benchmarks, one faces a number of challenges if one wishes to use them to compare safety verification methods and tools. Firstly, in contrast to the world of SAT/SMT solving or automated theorem proving, verification of continuous systems has not matured to the point where the community has agreed upon an input standard that can be used to exchange problems (such as SMT-LIB [2] or TPTP [3]). Also, unlike with numerical analysis or simulation, general safety verification problems need not have point initial conditions, but rather a set of initial states that may be uncountably infinite, and not necessarily "nice" (e.g. may be disconnected, non-convex, unbounded, etc.). Below we outline some important challenges that stand in the way of benchmarking existing verification tools.

- Tools for bounded-time safety verification based on computing flowpipes enclosing reachable sets of non-linear ODEs, such as e.g. Flow*, are often limited in the nature of the initial and the forbidden sets of states. In particular, the underlying algorithms used in these tools require the set of initial states to be bounded (unlike in Example 2.3); ideally given by a hyper-rectangle (unlike Example 2.1). On the other hand, methods for automatic unbounded-time safety verification based on searching for appropriate continuous invariants (e.g. [29, 37]) are capable of working with much broader classes of initial and forbidden regions. For instance, semi-algebraic initial regions that are unbounded, non-convex, or whose description features a combination of conjunctions and disjunctions do not present a problem.

  Remark At the same time, tools based on flowpipe construction can sometimes give a sense of the "hardness" of the verification problem when they fail to prove safety up to some given time bound, whereas invariant-based verification tools typically do not provide useful insights into the nature or the difficulty of the problem when they fail.

- Tools that employ interval arithmetic often require bounds on the state variables of the system (e.g. HSolver [33, 34], dReach [21]), which technically renders them inapplicable to safety verification problems where the evolution constraint H is unbounded, e.g. given by $\mathbb{R}^n$.
- Certain tools (e.g. Flow*) cannot work with sets described by strict inequalities (such as the forbidden states in Example 2.3). While it would be sound to simply over-approximate the closure of such sets by relaxing the inequalities to be non-strict, this step currently needs to be performed manually by the user and (inevitably) affects the reachability analysis.
- The performance of tools often depends heavily on the user-specified options, such as e.g. the fixed/adaptive time steps used for the verified integration, error tolerances, etc. It is presently not apparent how one might automatically translate "good" settings from one verification tool to another, or indeed automatically arrive at good settings for a particular tool in the first place. Thus, some verification tools that are designed to be fully automatic rely crucially on the user choosing the right settings, which is typically difficult for a non-expert.
- Some unbounded-time verification methods (e.g. [31]) likewise require significant manual input from the user, such as e.g. selecting templates for polynomial functions. It is yet unclear how these methods can be meaningfully compared to methods that provide a greater level of automation.
- Uncertainty in the continuous dynamics is permitted by some verification tools (e.g. Flow*), but not others.

4 Outlook

Safety verification problems for non-linear systems are very useful for assessing the utility and efficiency of invariant generation methods (e.g. [35, 38, 6, 29, 26, 40, 17, 37]), as well as tools based on verified integration of ODEs (e.g [7, 27, 20, 21]). We are hopeful that maintaining and further populating the set of verification benchmarks will result in improvements to the existing capabilities offered by the tools for both bounded and unbounded-time safety verification. Improvements in invariant generation would also greatly benefit deductive verification tools for hybrid systems, such as theorem provers (e.g. [30, 15, 23]).

At least some of the challenges outlined in the previous section can potentially be addressed using HyST [5], a source transformation tool for hybrid systems that takes as input a hybrid system verification problem in the SpaceEx format and translates it into formats accepted by other verification tools. In addition to translating between the various problem formats, HyST is able to work with its internal representation of the verification problem through so-called *model transformation passes*, which can address issues that affect particular verification tools. For instance, currently HyST can add identity reset maps to transitions in hybrid automata, split transition guards with disjunctions, etc. A potentially interesting future transformation pass could be implemented in HyST to convert continuous systems with uncertainty into *hybrid* systems in which there is no uncertainty in the continuous dynamics, e.g. following the work of Ramdani et al. [32].

At present, HyST can translate problems into formats accepted by Flow*, dReach, HyCreate [4], HyComp [8] and SpaceEx. An interesting future direction would be to extend it to also work with invariant generation tools and add model transformation passes to soundly convert safety verification problems that currently cannot be processed by some of the verification tools into a form that is amenable to analysis.

In collecting safety verification benchmarks it is profitable to find a useful classification. One could separate verification problems for continuous systems into classes depending on certain features, such as:

- the type of continuous dynamics, e.g. constant/linear/non-linear,
- the dimensionality of the system (i.e. the number of state variables, $|\boldsymbol{x}|$),
- the type of safety verification (i.e. bounded versus unbounded time),
- the nature of the evolution constraint (e.g. bounded versus unbounded state space),
- the nature of the initial and forbidden set (bounded versus unbounded; if bounded, hyper-rectangles versus more general sets), and
- the nature of the verification problem itself (i.e. is the system safe or unsafe?).

Such a classification will certainly become important in the future as more verification problems are gathered and added to our collection. Our initial set of 65 problems (which we tentatively labelled `NONLIN-UNBOUND-TIME-SAFE`) belongs to one of the most general classes under this scheme, since it makes few assumptions about the nature of the verification problem. This generality makes it difficult to use the problems for benchmarking existing tools, but at the same time serves to bring out their current limitations.

Acknowledgements The authors would very much like to thank the anonymous reviewers for their careful reading, pertinent points of critique and valuable suggestions for improving this document. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the

Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

[1] The international SAT competitions web page. http://www.satcompetition.org/. Accessed: 2016-02-15.

[2] SMT-LIB the satisfiability modulo theories library. http://smtlib.cs.uiowa.edu/. Accessed: 2016-02-15.

[3] The TPTP problem library for automated theorem proving. http://www.cs.miami.edu/~tptp/. Accessed: 2016-02-15.

[4] Stanley Bak. HyCreate: A tool for overapproximating reachability of hybrid automata. http://stanleybak.com/projects/hycreate/hycreate.html. Accessed: 2016-02-15.

[5] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HYST: a source transformation and translation tool for hybrid automaton models. In *HSCC*, pages 128–133. ACM, 2015.

[6] M.A. Ben Sassi, A. Girard, and S. Sankaranarayanan. Iterative computation of polyhedral invariants sets for polynomial dynamical systems. In *CDC*, pages 6348–6353, Dec 2014.

[7] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *CAV*, pages 258–263, 2013.

[8] Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. HyComp: An SMT-based model checker for hybrid systems. In *TACAS*, pages 52–67, 2015.

[9] Liyun Dai, Ting Gan, Bican Xia, and Naijun Zhan. Barrier certificates revisited. *CoRR*, abs/1310.6481, 2013.

[10] Raymond A. DeCarlo, Stanisław H. Żak, and Gregory P. Matthews. Variable structure control of nonlinear multivariable systems: a tutorial. *Proceedings of the IEEE*, 76(3):212–232, 1988.

[11] A Djaballah, A. Chapoutot, M. Kieffer, and O Bouissou. Construction of Parametric Barrier Functions for Dynamical Systems using Interval Analysis. *ArXiv e-prints*, June 2015.

[12] Freddy Dumortier, Jaume Llibre, and Joan C. Artés. *Qualitative Theory of Planar Differential Systems*. Springer, 2006.

[13] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *HSCC*, volume 3414, pages 258–273. Springer, 2005.

[14] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*, volume 6806 of *LNCS*. Springer, 2011.

[15] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In *CADE*, volume 9195 of *LNCS*. Springer, 2015.

[16] Ting Gan, Mingshuai Chen, Liyun Dai, Bican Xia, and Naijun Zhan. Decidability of the reachability for a family of linear vector fields. In *ATVA*, pages 482–499, 2015.

[17] Khalil Ghorbal and André Platzer. Characterizing algebraic invariants by differential radical invariants. In *TACAS*, volume 8413, pages 279–294. Springer, 2014.

[18] Emmanuel Hainry. Reachability in linear dynamical systems. In *Logic and Theory of Algorithms, CiE 2008*, volume 5028 of *LNCS*, pages 241–250, 2008.

[19] Jack K. Hale and Joseph P. LaSalle. Differential equations: Linearity vs. nonlinearity. *SIAM Review*, 5(3):249–272, July 1963.

[20] Fabian Immler. Verified reachability analysis of continuous systems. In *TACAS*, pages 37–51, 2015.

[21] Soonho Kong, Sicun Gao, Wei Chen, and Edmund M. Clarke. dreach: δ-reachability analysis for hybrid systems. In *TACAS*, pages 200–205. Springer, 2015.

[22] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *J. Symb. Comput.*, 32(3):231–253, 2001.

[23] Jiang Liu, Jidong Lv, Zhao Quan, Naijun Zhan, Hengjun Zhao, Chaochen Zhou, and Liang Zou. A calculus for hybrid CSP. In *Programming Languages and Systems*, volume 6461 of *LNCS*, pages 1–15. Springer, 2010.

[24] Jiang Liu, Naijun Zhan, and Hengjun Zhao. Computing semi-algebraic invariants for polynomial dynamical systems. In *EMSOFT*, pages 97–106, New York, NY, USA, 2011. ACM.

[25] Jiang Liu, Naijun Zhan, Hengjun Zhao, and Liang Zou. Abstraction of elementary hybrid systems by variable transformation. In *FM*, pages 360–377, 2015.

[26] Nadir Matringe, Arnaldo Vieira Moura, and Rachid Rebiha. Generating invariants for non-linear hybrid systems by linear algebraic methods. In *SAS*, volume 6337 of *LNCS*, pages 373–389. Springer, 2011.

[27] Nedialko S. Nedialkov. Interval Tools for ODEs and DAEs. In *SCAN*, pages 4–4, Sept 2006.

[28] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput.*, 20(1):309–352, 2010.

[29] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In *CAV*, pages 176–189, 2008.

[30] André Platzer and Jan-David Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In *IJCAR*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.

[31] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *HSCC*, volume 2993 of *LNCS*, pages 477–492. Springer, 2004.

[32] N. Ramdani, N. Meslem, and Y. Candau. A hybrid bounding method for computing an over-approximation for the reachable set of uncertain nonlinear systems. *Automatic Control, IEEE Transactions on*, 54(10):2352–2364, Oct 2009.

[33] Stefan Ratschan and Zhikun She. HSolver. http://hsolver.sourceforge.net/, 2004. Accessed: 2016-02-15.

[34] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(1):8, 2007.

[35] Sriram Sankaranarayanan. Automatic invariant generation for hybrid systems using ideal fixed points. In *HSCC*, pages 221–230, New York, NY, USA, 2010. ACM.

[36] B. I. Silva, Keith Richeson, Bruce Krogh, and Alongkrit Chutinan. Modeling and verifying hybrid dynamic systems using CheckMate. In *ADPM*, 2000.

[37] Andrew Sogokon, Khalil Ghorbal, Paul B. Jackson, and André Platzer. A method for invariant generation for polynomial continuous systems. In *VMCAI*, volume 9583 of *LNCS*. Springer, 2016.

[38] Ashish Tiwari. Generating box invariants. In *HSCC*, volume 4981 of *LNCS*, pages 658–661. Springer, 2008.

[39] Hoang-Dung Tran, Luan Viet Nguyen, and Taylor T. Johnson. Benchmark: A nonlinear reachability analysis test set from numerical analysis. In *Applied Verification for Continuous and Hybrid Systems Workshop*, Seattle, Washington, April 2015.

[40] Hengjun Zhao, Naijun Zhan, and Deepak Kapur. Synthesizing switching controllers for hybrid systems by generating invariants. In *Theories of Programming and Formal Methods*, pages 354–373, 2013.