EPiC Series in Computing

Volume 74, 2020, Pages 49–75

ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)

ARCH-COMP20 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics

Luca Geretti[1], Julien Alexandre dit Sandretto[2], Matthias Althoff[3], Luis Benet[4], Alexandre Chapoutot[2], Xin Chen[5], Pieter Collins[6], Marcelo Forets[7], Daniel Freire[7], Fabian Immler[8], Niklas Kochdumper[3], David P. Sanders[9], and Christian Schilling[10]

[1] Department of Computer Science, University of Verona, Verona, Italy
luca.geretti@univr.it
[2] ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France
julien.alexandre-dit-sandretto@ensta-paris.fr,alexandre.chapoutot@ensta-paris.fr
[3] Technische Universität München, Munich, Germany
althoff@in.tum.de,niklas.kochdumper@tum.de
[4] Instituto de Ciencias Físicas, Universidad Nacional Autónoma de México (UNAM), México
benet@icf.unam.mx
[5] University of Dayton, Dayton, OH, United States
xchen4@udayton.edu
[6] Department of Data Science and Knowledge Engineering, Maastricht University, Maastricht, The Netherlands
pieter.collins@maastrichtuniversity.nl
[7] Universidad de la República, Montevideo, Uruguay
mforets@gmail.com,dfreire@fisica.edu.uy
[8] Computer Science Department, Carnegie Mellon University, United States
fimmler@cs.cmu.edu
[9] Departamento de Física, Facultad de Ciencias,
Universidad Nacional Autónoma de México (UNAM), Mexico City, México
& Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, United States
dpsanders@ciencias.unam.mx
[10] IST Austria, Klosterneuburg, Austria
christian.schilling@ist.ac.at

Abstract

We present the results of a friendly competition for formal verification of continuous and hybrid systems with nonlinear continuous dynamics. The friendly competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2020. This year, 6 tools Ariadne, CORA, DynIbex, Flow*, Isabelle/HOL, and JuliaReach (in alphabetic order) participated. These tools are applied to solve reachability analysis problems on six benchmark problems, two of them featuring hybrid dynamics. We do not rank the tools based on the results, but show the current status and discover the potential advantages of different tools.

1 Introduction

Disclaimer The presented report of the ARCH friendly competition for *continuous and hybrid systems with nonlinear dynamics* aims at providing a landscape of the current capabilities of verification tools. We would like to stress that each tool has unique strengths—though not all of their features can be highlighted within a single report. To reach a consensus in what benchmarks are used, some compromises had to be made so that some tools may benefit more from the presented choice than others. The obtained results have been verified by an independent repeatability evaluation. To establish further trustworthiness of the results, the code with which the results have been obtained is publicly available as Docker [18] containers at gitlab.com/goranf/ARCH-COMP.

In this report, we summarize the results of the fourth ARCH friendly competition on the reachability analysis of continuous and hybrid systems with nonlinear dynamics. Given a system defined by a nonlinear Ordinary differential equation (ODE) $\dot{\vec{x}} = f(\vec{x}, t)$ along with an initial condition $\vec{x} \in X_0$, we apply the participating tools to prove properties of the state reachable set in a bounded time horizon. The techniques for solving such a problem are usually very sensitive to not only the nonlinearity of the dynamics but also the size of the initial set. This is also one of the main reasons why most of the tools require quite a lot of computational parameters.

In this report, 6 tools, namely Ariadne, CORA, DynIbex, Flow*, Isabelle/HOL, and JuliaReach participated in solving problems defined on four continuous and two hybrid benchmarks. The continuous benchmarks are the Production-Destruction system, the Coupled Van der Pol oscillator, the Laub-Loomis model, and a controlled Quadrotor model. The hybrid benchmarks model a Lotka-Volterra system with a Tangential Crossing, and a Space Rendezvous system.

The benchmarks are selected based on discussions of the tool authors, with a preference on keeping a significant set of the benchmarks from the previous year.

2 Participating Tools

Ariadne. (Luca Geretti, Pieter Collins) *Ariadne* [25, 16] is a library based on *Computable Analysis* [41] that uses a rigorous numerical approach to all its algebraic, geometric and logical operations. In particular, it performs numerical rounding control of all external and internal operations, in order to enforce conservative interpretation of input specification and guarantee formal correctness of the computed output. It focuses on nonlinear systems, both continuous and hybrid, supporting differential and algebraic relations. It is written in C++ with growing support for a Python interface. The official site for Ariadne is `http://www.ariadne-cps.org`.

CORA. (Matthias Althoff, Niklas Kochdumper) The tool *COntinuous Reachability Analyzer* (CORA) [6, 7] realizes techniques for reachability analysis with a special focus on developing scalable solutions for verifying hybrid systems with nonlinear continuous dynamics and/or nonlinear differential-algebraic equations. A further focus is on considering uncertain parameters and system inputs. Due to the modular design of CORA, much functionality can be used for other purposes that require resource-efficient representations of multi-dimensional sets and operations on them. CORA is implemented as an object-oriented MATLAB code. The modular design of CORA makes it possible to use the capabilities of the various set representations for other purposes besides reachability analysis. While CORA uses verified algorithms, it does not consider rounding errors since the main focus of the toolbox is the fast prototyping of new reachability algorithms and concepts, and for this purpose the effect of rounding errors is usually negligible. CORA is available at cora.in.tum.de.

DynIbex. (Alexandre Chapoutot, Julien Alexandre dit Sandretto) A library merging interval constraint satisfaction problem algorithms and guaranteed numerical integration methods based on Runge-Kutta numerical schemes implemented with affine arithmetic. This library is able to solve ordinary differential equations [2] and algebraic differential equations of index 1 [3], combined with numerical constraints on state variables and reachable tubes. It produces sound results taking into account round-off errors in floating-point computations and truncation errors generated by numerical integration methods [36]. Moreover, constraint satisfaction problem algorithms offer a convenient approach to check properties on reachable tubes as explained in [4]. This library implements in a very generic way validated numerical integration methods based on Runge-Kutta methods without many optimizations. Indeed, the computation of the local truncation error, for each method, depends only on the coefficients of Runge-Kutta methods and their order. DynIbex is freely available at `http://perso.ensta-paristech.fr/~chapoutot/dynibex/`. Figures have been produced with VIBes library [27] which is available at `http://enstabretagnerobotics.github.io/VIBES/`.

Flow*. (Xin Chen) The tool Flow* [23, 21] uses an adapted Taylor Model (TM) integration method to compute reachable set overapproximations for nonlinear continuous and hybrid systems. Similar to the original method proposed in [17], an ODE solution, i.e., a function over the initial set as well as the time variable, over a bounded time interval is overapproximated by a TM in Flow*, and it therefore forms an overapproximation of the reachable set there. We also call this TM a TM flowpipe. This year, we evaluate the performance of a new version of the tool which has the most of the original features implemented and greatly improved. All of the experiments are done based on the C++ API of the new version. The flowpipe/guard intersections [22] are better handled in the new version by an improved domain contraction method, and users are more free to use their own aggregation programs to merge the flowpipes. As in the previous versions, the result produced by Flow* is guaranteed, all of the roundoff errors, simplification errors (w.r.t. the cutoff threshold), range overapproximation errors are taken into account during the flowpipe computation and verification.

Isabelle/HOL-ODE-Numerics. (Fabian Immler) HOL-ODE-Numerics [30, 31] is a collection of rigorous numerical algorithms for continuous systems. It is based on Runge-Kutta methods implemented with affine arithmetic. The distinctive feature is that all algorithms are formally verified in the interactive theorem prover Isabelle/HOL: everything from single roundoff errors to the global approximation scheme is proved correct with respect to a formalization of ODEs in Isabelle/HOL. The resulting code is therefore highly trustworthy. It does, however, not feature many optimizations or the most sophisticated algorithms. We therefore do not expect competitive performance figures. Nevertheless, the tool should exhibit reasonable performance: it should scale (modulo possibly large constant factors) like "regular" tools implementing similar algorithms. Isabelle/HOL is available at `https://isabelle.in.tum.de`, HOL-ODE-Numerics is part of the Archive of Formal Proofs `http://isa-afp.org/entries/Ordinary_Differential_Equations.shtml`.

JuliaReach. (Marcelo Forets, Christian Schilling, David P. Sanders, Luis Benet, Daniel Freire) JuliaReach [19] is an open-source software suite for reachability computations of dynamical systems, written in the Julia language and available at `http://github.com/JuliaReach`. Linear, nonlinear, and hybrid problems are modeled and solved using the library ReachabilityAnalysis.jl, which can be used interactively, for example in Jupyter notebooks. Our implementation of Taylor models [15] integrates the TaylorSeries.jl [12, 13] package, TaylorIntegration.jl [37] and the IntervalArithmetic.jl [14] package for interval methods. The latter is a self-contained implementation of interval arithmetic in pure Julia, which is competitive, in terms of performance, with C++ interval packages, and yet has interactive usability similar to the MATLAB package INTLAB. There are several different implementations of directed rounding available in IntervalArithmetic.jl [14]. By default *error-free arithmetic* is used for the basic arithmetic operations, to avoid the overhead of changing the processor's rounding mode; this effectively makes the package thread-safe. The error-free operations are implemented in the RoundingEmulator.jl package. Correctly-rounded elementary functions such as sin and cos use the CRlibm.jl wrapper of the CRlibm library [26]. The algorithm `TMJets` applied in this report computes a non-validated integration using a Taylor model of order n_T. The coefficients of that series are polynomials of order n_Q in the variables that denote the small variations of the initial conditions. We obtain a time step from the last two coefficients of this time series. In order to validate the integration step, we compute a second integration using intervals as coefficients of the polynomials in time, and we obtain a bound for the integration using a Lagrange-like remainder. The remainder is used to check the contraction of a Picard iteration. If the combination of the time step and the remainder do not satisfy the contraction, we iteratively enlarge the remainder or possibly shrink the time step. Finally, we evaluate the initial Taylor series with the valid remainder at the time step for which the contraction has been proved, which is also evaluated in the initial set to yield an over-approximation. The approach is (numerically) sound due to rigorous interval bounds in the Taylor approximation. Discrete transitions for hybrid systems and Taylor model approximations are handled using our set-computations library LazySets.jl [38].

3 Benchmarks

For the 2020 edition of the competition we introduced two new benchmarks: the *production-destruction* system and the *Lotka-Volterra* system with tangential crossing. The former is a continuous system aimed at identifying the stability of integration schemes. The latter introduces nonlinear guards and in particular tangential crossings. In addition, we extended the van der Pol oscillator to two coupled oscillators, introduced nondeterministic crossing in the *space rendezvous* system and performed some input sensitivity analysis for the *quadrotor* system. The *Laub-Loomis* benchmark was not modified, in order to perform a direct comparison with results from the previous year (since the participants are the same). We also introduce a 4-letter code for each benchmark followed by the year in the YY format.

Compared with last year, the experimental results are produced on a common platform, using an Amazon EC2 G4 g4dn.4xlarge instance [1] with 16 Xeon vCPUs and 64 GiB RAM. Each tool supplied a repeatability package using Docker, with an agreed output format. Due to issues with the Isabelle/HOL package on the repeatability server, computation time values are obtained on a laptop machine instead; the values are then normalized by multiplying them by the speedup factor of the Amazon EC2 G4 machine with respect to the laptop machine, as calculated for each benchmark using Flow* as a reference tool. To distinguish those timing values, they are marked by a "*".

3.1 Production-destruction benchmark (PRDE20)

3.1.1 Model

A production-destruction system consists of an Ordinary Differential Equation and two constraints: positivity and conservativity. It means that the system states are quantities which are always positive and the sum of these quantities is constant. This particular family of systems is often used to test the stability of integration schemes.

As proposed by Michaelis-Menten theory [34], a model for three quantities can be defined as follows:

$$\begin{cases} \dot{x} = & \dfrac{-xy}{1+x} \\ \dot{y} = & \dfrac{xy}{1+x} - ay \\ \dot{z} = & ay \end{cases}$$

with $a = 0.3$ and the initial condition $x(0) = 9.98$, $y(0) = 0.01$ and $z(0) = 0.01$. In this model, x is the nutrients, y the phytoplankton and z the detritus. The constraints are $x(t)$, $y(t)$, $z(t)$ are positive and $x(t) + y(t) + z(t) = 10$ for all t.

3.1.2 Analysis

We are interested in computing the reachable tube until $t = 100$ and to verify (or embed) the constraints. Three setups are considered, depending on the source of bounded uncertainties:

I: $x(0) \in [9.5, 10.0]$, *i.e.*, uncertainty on the initial condition;

P: $a \in [0.296, 0.304]$, *i.e.*, time-invariant uncertainty on the parameter;

$I\&P$: $x(0) \in [9.7, 10.0]$ and $a \in [0.298, 0.302]$, *i.e.*, both uncertainties are mixed.

In terms of objectives, at $t = 100$, the constraints $10 \in x + y + z$ and $x, y, z \geq 0$ have to be verified.

Table 1: Results of PRDE20 in terms of computation time and volume of final enclosure.

	computation time in [s]		
tool	*I*	*P*	*I&P*
Ariadne	8.6	79	39
CORA	16	28	28
DynIbex	12	13	26
Flow*	4.1	9	5.2
Isabelle/HOL	11*	12*	26*
JuliaReach	1.5	3.9	3.0

	volume of final enclosure		
tool	*I*	*P*	*I&P*
Ariadne	$1.7e^{-13}$	$2.3e^{-17}$	$1.0e^{-13}$
CORA	$1.2e^{-21}$	$2.2e^{-23}$	$2.0e^{-23}$
DynIbex	$3.9e^{-17}$	$4.8e^{-17}$	$1.2e^{-17}$
Flow*	$8.0e^{-21}$	$1.4e^{-22}$	$4.8e^{-21}$
Isabelle/HOL	$3.31e^{-20}$	$7.29e^{-21}$	$2.60e^{-20}$
JuliaReach	$3.3e^{-20}$	$6.5e^{-21}$	$1.0e^{-20}$

3.1.3 Evaluation

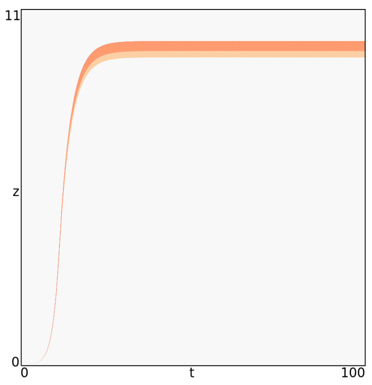
Two measures are provided for comparison: the volume of the box ($x \times y \times z$) enclosing the final state (at $t = 100$) and the total time of computation for evolution and verification. All of these results are obtained for the three setups. Plots of z (in the $[0, 11]$ range) w.r.t. time (in the $[0, 100]$ range) for the three cases are overlaid in a single figure.

3.1.4 Results

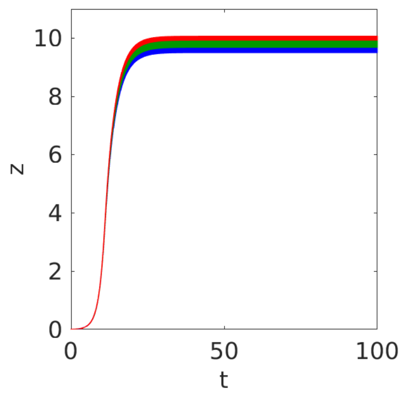
While the satisfaction of the specification seemed to be rather trivial for all tools, there is a wide degree of difference in the volume of the final enclosure. However from Fig. 1 it is apparent that similar errors are obtained for z hence the differences in volume are largely due to convergence quality with respect to x,y, i.e., towards zero. Given these results, improved metrics would distinguish z from x and y. Also, it seems that ranges in the three setups may overlap, where the overlapping is different based on the accuracy chosen.

Settings for Ariadne. For all three cases we use a TaylorPicardIntegrator with a maximum error of 10^{-6}, and use a maximum step size of 0.04.

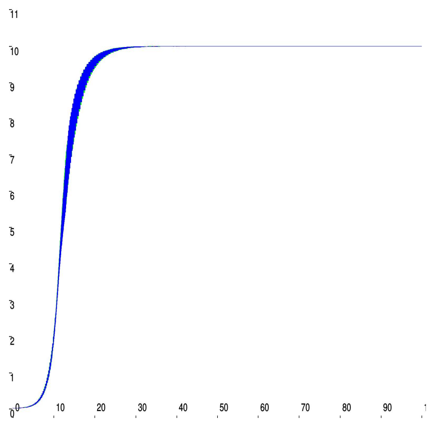
Settings for CORA. For CORA we apply the conservative linearization approach [9] and use zonotopes to represent the reachable set. We use a time step size of 1/17 for case I and 1/30 for case I and I&P. Furthermore, we use a maximum zonotope order of 20.



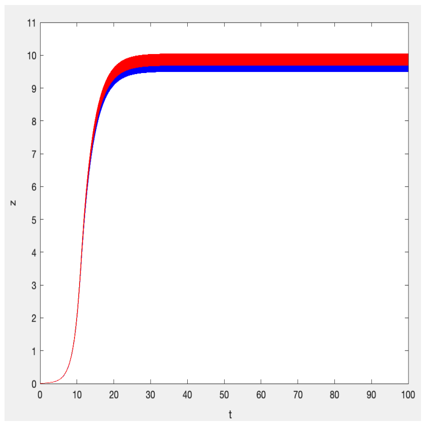
(a) Ariadne



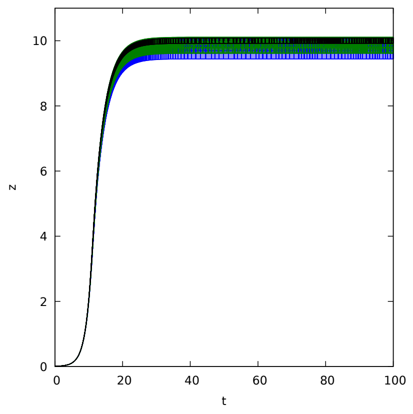
(b) CORA



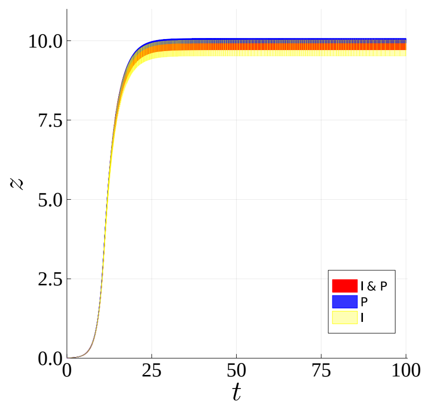
(c) DynIbex



(d) Flow*



(e) Isabelle/HOL



(f) JuliaReach

Figure 1: Reachable set overapproximations for PRDE20 in the I, P and $I\&P$ setups, for variable $z(t)$ and $t \in [0, 100]$.

Settings for DynIbex. Reachability analysis is carried out with an (absolute and relative) error tolerance of 10^{-8} for I and P, and 10^{-10} for I&P, using an implicit Gauss-Legendre method of order 4 and the constraint programming facilities.

Settings for Flow*. We use the same setting for I and I&P, such that the time stepsize is 0.02, TM order is 3, the cutoff threshold is 10^{-12}, and the remainder estimation is $[-0.1, 0.1]$ for all dimensions. For P, we only raise the TM order to 4.

Settings for Isabelle/HOL. Maximum Zonotope order is set to 18. Reachability analysis is carried out with an (absolute and relative) error tolerance of 2^{-16}. The initial set of I and I&P is split into two sets along the x coordinate.

Settings for JuliaReach. For all three cases we use $n_T = 7$, $n_Q = 1$, and an adaptive absolute tolerance with initial value 10^{-11} (resp 10^{-12}) is used for I and I&P (resp. P).

3.2 Coupled van der Pol benchmark (CVDP20)

3.2.1 Model

The original van der Pol oscillator was introduced by the Dutch physicist Balthasar van der Pol. For this benchmark we consider two coupled oscillators, as described in [10]. The system can be defined by the following ODE with 4 variables:

$$\begin{cases} \dot{x}_1 &= y_1 \\ \dot{y}_1 &= \mu(1 - x_1^2)y_1 - 2x_1 + x_2 \\ \dot{x}_2 &= y_2 \\ \dot{y}_2 &= \mu(1 - x_2^2)y_2 - 2x_2 + x_1 \end{cases} \tag{1}$$

The system has a stable limit cycle that becomes increasingly sharper for higher values of μ.

3.2.2 Analysis

We want to separately verify a safety specification for $\mu = 1$ and $\mu = 2$.

$\mu = 1$: we set the initial condition $x_{1,2}(0) \in [1.25, 1.55]$, $y_{1,2}(0) \in [2.35, 2.45]$. The unsafe set is given by $y_{1,2} \geq 2.75$ in a time horizon of $[0, 7]$. This is the same specification as the one used for the single oscillator in the 2019 competition.

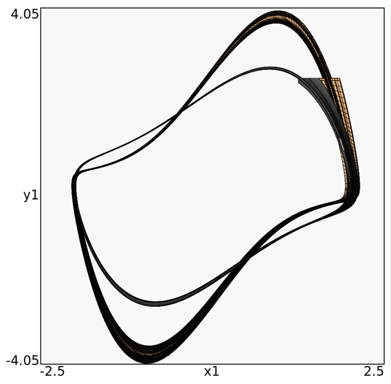
$\mu = 2$: we set the initial condition $x_{1,2}(0) \in [1.55, 1.85]$, $y_{1,2}(0) \in [2.35, 2.45]$, which is the same size as before, but on the limit cycle for $\mu = 2$. The unsafe set is given by $y_{1,2} \geq 4.05$ for a time horizon of $[0, 8]$. Note that in this case, the time horizon $T = 8.0$ is used because 7.0 is not sufficient for the oscillator to make a complete loop. The unsafe set has been slightly reduced compared with last year, due to the expected greater effort required for the coupled oscillators.

3.2.3 Evaluation

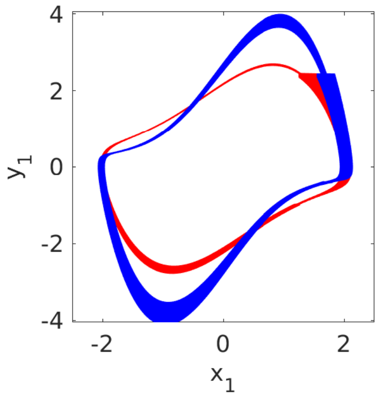
The computation time required to evolve the system and verify safety is provided for both values of μ. If the system can not be verified successfully, no value is provided.

3.2.4 Results

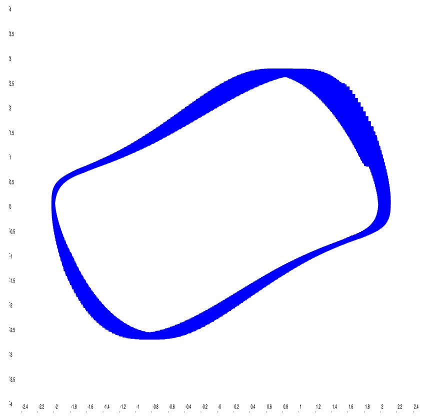
The computation results of the tools are given in Table 2. The increase in complexity for this benchmark resulted in a significant computational cost, for which Isabelle/HOL was unable to reach results in a reasonable time for both μ values, while DynIbex failed for $\mu = 2$. Ariadne and Flow* managed to handle the $\mu = 2$ case with splitting of the initial set, but still with a very large execution time; JuliaReach used splitting with a much more affordable time, while CORA succeeded without relying on any splitting strategy. The projection for the first oscillator is shown in Fig. 2, only for $\mu = 1$ in the case of DynIbex, and with no figures in the case of Flow* due to the excessively large number of flowpipes.

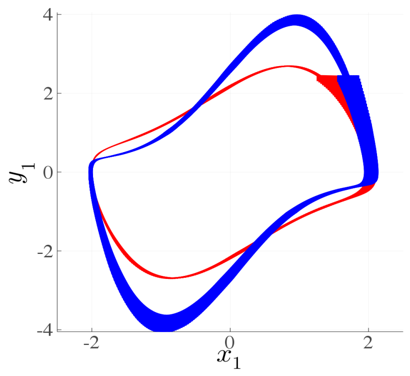


(a) Ariadne



(b) CORA


(c) DynIbex



(d) JuliaReach

Figure 2: Reachable set overapproximations for the first oscillator in CVDP20, $x_1 \in [-2.5, 2.5]$, $y_1 \in [-4.05, 4.05]$, overlaid for $\mu = 1$ and $\mu = 2$.

Table 2: Results of CVDP20 in terms of computation time.

	computation time in [s]	
tool	$\mu = 1$	$\mu = 2$
Ariadne	56	9838
CORA	8.0	122
DynIbex	517	–
Flow*	3.6	3725
Isabelle/HOL	–	–
JuliaReach	0.4	38

Settings for Ariadne. For $\mu = 1$, we use a TaylorPicardIntegrator with a maximum step size of 0.005. The maximum spacial error enforced for each step is 10^{-5}. The initial set is split once on x_1 and x_2, yielding 4 initial subsets. For $\mu = 2$ instead, the initial set is split into 256 subsets, using a maximum spacial error of 5×10^{-6} and a maximum step size of 0.02. It must be noted that the memory consumption for $\mu = 2$ is significant, in some cases making the operating system kill the process on a 16 GB machine during the latter part of the evolution.

Settings for CORA. We manually introduced artificial guard sets orthogonal to the flow of the system in order to shrink the reachable set so that the linearization error does not explode. For CORA we then applied the conservative linearization approach [9] using zonotopes with a zonotope order of 20 and a time step size of 0.01 for $\mu = 1$, and a zonotope order of 100 and a time step size of 0.001 for $\mu = 2$.

Settings for DynIbex. Maximum zonotope order is set to 100, reachability analysis is carried out with an (absolute and relative) error tolerance of 10^{-5} using an explicit Heun method of order 2. For $\mu = 1$ an automatic partition of the initial state is performed (259 boxes are necessary). For $\mu = 2$, the system cannot be verified in satisfying time.

Settings for Flow*. The tool uses a fixed stepsize 0.02, a fixed TM order 5, the cutoff threshold 10^{-7}, and the remainder estimation of $[-0.1, 0.1]$ to prove the safety for the benchmark of $\mu = 1$. For the more challenging case $\mu = 2$, we used a more automatic setting, that is, we let the tool adaptively select a stepsize between 0.0002 and 0.1 along with a fixed order 6, the initial set is uniformly divided into 4^4 pieces and each of them is analyzed independently. Although the time cost is large, the whole process is done automatically by the tool. Due to the large number of flowpipes, we don't show the figures.

Settings for Isabelle/HOL. None of the algorithms implemented in Isabelle/HOL can verify this problem without excessive subdivisions of the initial set.

Settings for JuliaReach. For both settings we used $n_Q = 1$ and $n_T = 7$, and an adaptive absolute tolerance. For $\mu = 2$ we split the set of initial states along the x_1 and x_2 directions into 64 boxes in total. In this case, the multi-threaded execution (4 threads) runs in 18s.

3.3 Laub-Loomis benchmark (LALO20)

3.3.1 Model

The Laub-Loomis model is presented in [35] for studying a class of enzymatic activities. The dynamics can be defined by the following ODE with 7 variables.

$$\begin{cases} \dot{x}_1 &= 1.4x_3 - 0.9x_1 \\ \dot{x}_2 &= 2.5x_5 - 1.5x_2 \\ \dot{x}_3 &= 0.6x_7 - 0.8x_2x_3 \\ \dot{x}_4 &= 2 - 1.3x_3x_4 \\ \dot{x}_5 &= 0.7x_1 - x_4x_5 \\ \dot{x}_6 &= 0.3x_1 - 3.1x_6 \\ \dot{x}_7 &= 1.8x_6 - 1.5x_2x_7 \end{cases}$$

The system is asymptotically stable and the equilibrium is the origin.

3.3.2 Analysis

The specification for the analysis is kept the same as last year, in order to better quantify any improvements to the participating tools.

The initial sets are defined according to the ones used in [40]. They are boxes centered at $x_1(0) = 1.2$, $x_2(0) = 1.05$, $x_3(0) = 1.5$, $x_4(0) = 2.4$, $x_5(0) = 1$, $x_6(0) = 0.1$, $x_7(0) = 0.45$. The range of the box in the ith dimension is defined by the interval $[x_i(0) - W, x_i(0) + W]$. The width W of the initial set is vital to the difficulty of the reachability analysis job. The larger the initial set the harder the reachability analysis.

We consider $W = 0.01$, $W = 0.05$, and $W = 0.1$. For $W = 0.01$ and $W = 0.05$ we consider the unsafe region defined by $x_4 \geq 4.5$, while for $W = 0.1$, the unsafe set is defined by $x_4 \geq 5$. The time horizon for all cases is $[0, 20]$.

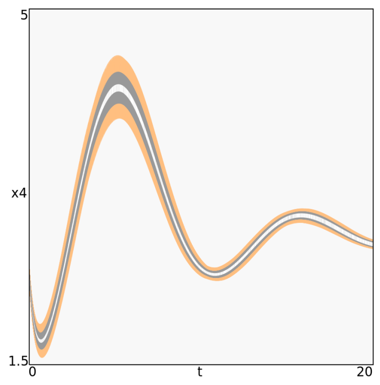
3.3.3 Evaluation

The final widths of x_4 along with the computation times are provided for all three cases. A figure is provided in the (t, x_4) axes, with $t \in [0, 20]$, $x_4 \in [1.5, 5]$, where the three plots are overlaid.

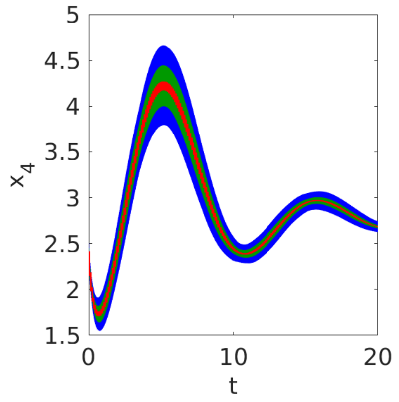
3.3.4 Results

The computation results of the tools are given in Table 3. It can be seen that enlarging the initial set size can greatly make the reachability analysis task harder. However, compared with the previous year, all tools improved the quality of their approximations, which is also apparent from Figure 3. The tool settings are given as below.

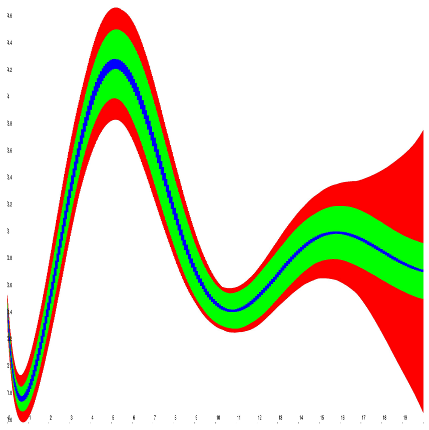
Settings for Ariadne. The maximum step size used is 0.2, with a TaylorPicardIntegrator with a maximum spacial error of 10^{-3} enforced for each step. For $W = 0.05$ and $W = 0.1$ a splitting strategy for the initial set is used; the strategy compares the radius of the set with a reference value, respectively 0.04 and 0.09, in order to split once for each dimension and yield a total of 128 initial subsets (the minimum amount using the current splitting strategy).



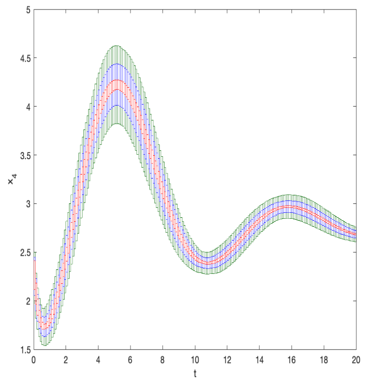
(a) Ariadne



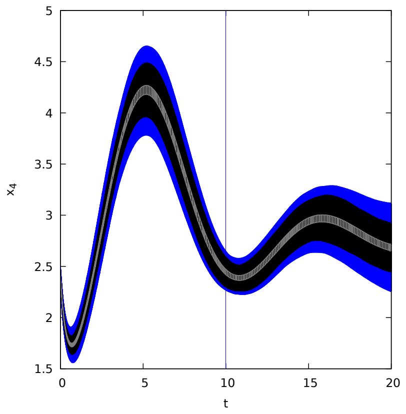
(b) CORA



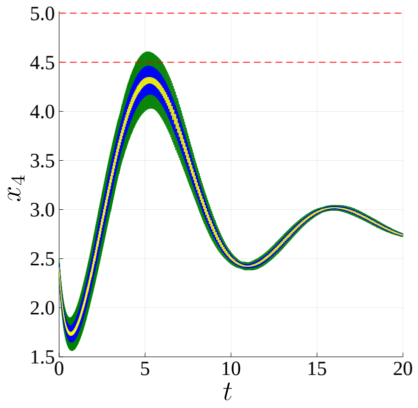
(c) DynIbex



(d) Flow*



(e) Isabelle/HOL



(f) JuliaReach

Figure 3: Reachable set overapproximations for LALO20 (overlayed plots for $W = 0.01$, $W = 0.05$, $W = 0.1$). $t \in [0, 20]$, $x_4 \in [1.5, 5]$.

Table 3: Results of LALO20 in terms of computation time and width of final enclosure.

	computation time in [s]		
tool	$W = 0.01$	$W = 0.05$	$W = 0.1$
Ariadne	4.8	664	1066
CORA	1.8	7.6	39
DynIbex	10	27	1535
Flow*	1.7	2.3	4.0
Isabelle/HOL	8.1*	13*	575*
JuliaReach	1.1	1.5	1.4

	width of x_4 in final enclosure		
tool	$W = 0.01$	$W = 0.05$	$W = 0.1$
Ariadne	0.037	0.058	0.093
CORA	0.01	0.04	0.12
DynIbex	0.01	0.40	2.07
Flow*	0.007	0.06	0.13
Isabelle/HOL	0.07	0.48	0.87
JuliaReach	0.004	0.017	0.033

Settings for CORA. Depending on the size of the initial set, different algorithms in CORA are applied. For the smaller initial sets $W = 0.01$ and $W = 0.05$, the faster but less accurate conservative linearization algorithm presented in [9] is executed. For the larger initial set $W = 0.1$, the more accurate conservative polynomialization algorithm from [5] is applied. CORA uses a step size of 0.1 for $W = 0.01$, a step size of 0.025 for $W = 0.05$, and a step size of 0.02 for $W = 0.1$. The maximum zonotope order for all initial sets is chosen as 200.

Settings for DynIbex. For $W = 0.01$ the maximum zonotope order is set to 50 and the reachability analysis is carried out with an (absolute and relative) error tolerance of 10^{-6} with an explicit Runge-Kutta method of order 3. For $W = 0.05$ the maximum zonotope order is set to 80 and the reachability analysis is carried out with an (absolute and relative) error tolerance of 10^{-7} with an explicit Runge-Kutta method of order 3. For $W = 0.01$ and $W = 0.05$ no splitting of the initial conditions is performed. For $W = 0.1$, the initial set is split 64 times. With parallelization, computation time is reduced to 249 seconds for this last experiment.

Settings for Flow*. The same setting in Flow* is used for all of the 3 tests: stepsize 0.1, TM order, $3 \sim 8$, cutoff threshold 10^{-6}, and the remainder estimation $[-0.001, 0.001]$. Besides, we turn on the symbolic remainder feature [24] such that we symbolically track the evolution of the flowpipe remainders for every 300 steps.

Settings for Isabelle/HOL. Maximum Zonotope order is set to 60. Reachability analysis is carried out with an (absolute and relative) error tolerance of 2^{-13}. For $W = 0.1$, we split the initial set along the x_1, x_2, x_3, x_4, x_5 coordinates, resulting in 32 initial sets that can be checked in parallel.

Settings for JuliaReach. We used an absolute tolerance of 10^{-11} for $W = 0.01$ and 10^{-12} for $W = 0.05$ and $W = 0.1$. In all cases, $n_Q = 1$, $n_T = 7$.

3.4 Quadrotor benchmark (QUAD20)

3.4.1 Model

We study the dynamics of a quadrotor as derived in [11, eq. (16) - (19)]. Let us first introduce the variables required to describe the model: the inertial (north) position x_1, the inertial (east) position x_2, the altitude x_3, the longitudinal velocity x_4, the lateral velocity x_5, the vertical velocity x_6, the roll angle x_7, the pitch angle x_8, the yaw angle x_9, the roll rate x_{10}, the pitch rate x_{11}, and the yaw rate x_{12}. We further require the following parameters: gravity constant $g = 9.81$ [m/s^2], radius of center mass $R = 0.1$ [m], distance of motors to center mass $l = 0.5$ [m], motor mass $M_{rotor} = 0.1$ [kg], center mass $M = 1$ [kg], and total mass $m = M + 4M_{rotor}$.

From the above parameters we can compute the moments of inertia as

$$\begin{aligned}
J_x =& \frac{2}{5}\, M\, R^2 + 2\, l^2\, M_{rotor},\\
J_y =& J_x,\\
J_z =& \frac{2}{5}\, M\, R^2 + 4\, l^2\, M_{rotor}.
\end{aligned}$$

Finally, we can write the set of ordinary differential equations for the quadrotor according to [11, eq. (16) - (19)]:

$$\begin{cases}
\dot{x}_1 &= \cos(x_8)\cos(x_9)x_4 + \Big(\sin(x_7)\sin(x_8)\cos(x_9) - \cos(x_7)\sin(x_9)\Big)x_5\\
&\quad +\Big(\cos(x_7)\sin(x_8)\cos(x_9) + \sin(x_7)\sin(x_9)\Big)x_6\\
\dot{x}_2 &= \cos(x_8)\sin(x_9)x_4 + \Big(\sin(x_7)\sin(x_8)\sin(x_9) + \cos(x_7)\cos(x_9)\Big)x_5\\
&\quad +\Big(\cos(x_7)\sin(x_8)\sin(x_9) - \sin(x_7)\cos(x_9)\Big)x_6\\
\dot{x}_3 &= \sin(x_8)x_4 - \sin(x_7)\cos(x_8)x_5 - \cos(x_7)\cos(x_8)x_6\\
\dot{x}_4 &= x_{12}x_5 - x_{11}x_6 - g\sin(x_8)\\
\dot{x}_5 &= x_{10}x_6 - x_{12}x_4 + g\cos(x_8)\sin(x_7)\\
\dot{x}_6 &= x_{11}x_4 - x_{10}x_5 + g\cos(x_8)\cos(x_7) - \frac{F}{m}\\
\dot{x}_7 &= x_{10} + \sin(x_7)\tan(x_8)x_{11} + \cos(x_7)\tan(x_8)x_{12}\\
\dot{x}_8 &= \cos(x_7)x_{11} - \sin(x_7)x_{12}\\
\dot{x}_9 &= \frac{\sin(x_7)}{\cos(x_8)}x_{11} + \frac{\cos(x_7)}{\cos(x_8)}x_{12}\\
\dot{x}_{10} &= \frac{J_y - J_z}{J_x}x_{11}x_{12} + \frac{1}{J_x}\tau_\phi\\
\dot{x}_{11} &= \frac{J_z - J_x}{J_y}x_{10}x_{12} + \frac{1}{J_y}\tau_\theta\\
\dot{x}_{12} &= \frac{J_x - J_y}{J_z}x_{10}x_{11} + \frac{1}{J_z}\tau_\psi
\end{cases}$$

To check interesting control specifications, we stabilize the quadrotor using simple PD controllers for height, roll, and pitch. The inputs to the controller are the desired values for height, roll, and pitch u_1, u_2, and u_3, respectively. The equations of the controllers are

$$\begin{aligned}
F &= m\,g - 10(x_3 - u_1) + 3x_6 && \text{(height control)},\\
\tau_\phi &= -(x_7 - u_2) - x_{10} && \text{(roll control)},\\
\tau_\theta &= -(x_8 - u_3) - x_{11} && \text{(pitch control)}.
\end{aligned}$$

We leave the heading uncontrolled so that we set $\tau_\psi = 0$.

3.4.2 Analysis

The task is to change the height from 0 [m] to 1 [m] within 5 [s]. A goal region $[0.98, 1.02]$ of the height x_3 has to be reached within 5 [s] and the height has to stay below 1.4 for all times. After 1 [s] the height should stay above 0.9 [m]. The initial value for the position and velocities (i.e., from x_1 to x_6) is uncertain and given by $[-\Delta, \Delta]$ [m], with $\Delta = 0.4$. All other variables are initialized to 0. This preliminary analysis must be followed by a corresponding evolution for $\Delta = 0.1$ and $\Delta = 0.8$ while keeping all the settings the same. No goals are specified for these cases: the objective instead is to understand the scalability of each tool with fixed settings.

3.4.3 Evaluation

Computation times for evolution and analysis are provided for all three cases. A figure is provided in the (t, x_3) axes, with $t \in [0, 5]$ and $x_3 \in [-0.8, 1.5]$, where the three plots are overlaid.

Table 4: Results of QUAD20 in terms of computation time.

	computation time in [s]		
tool	$\Delta = 0.1$	$\Delta = 0.4$	$\Delta = 0.8$
Ariadne	4.4	4.4	4.5
CORA	5.4	5.4	5.8
DynIbex	878	862	875
Flow*	5.2	5.3	5.2
Isabelle/HOL	42*	36*	37*
JuliaReach	2.6	1.9	1.9

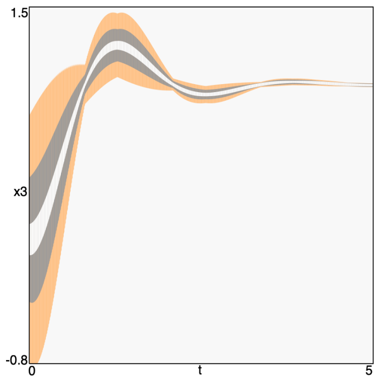
3.4.4 Results

The results of the reachability computation for the quadrotor model are given in Figure 4 and Table 4. All tools were able to produce results for the three setups, in particular the analysis with varying initial set sizes shows that the system is almost insensible to the initial set.

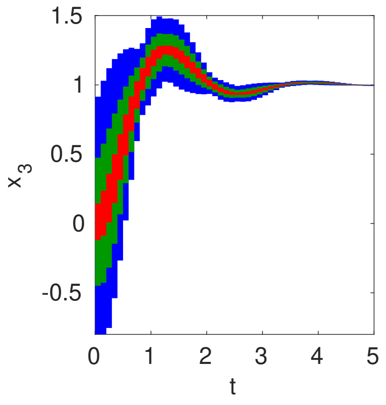
Settings for Ariadne. The maximum step size is 0.01. A TaylorPicardIntegrator is used with a maximum spacial error enforced for each step equal to 10^{-2}.

Settings for CORA. For CORA we apply the conservative linearization approach [9] with a time step size of 0.1 and a maximum zonotope order of 50.

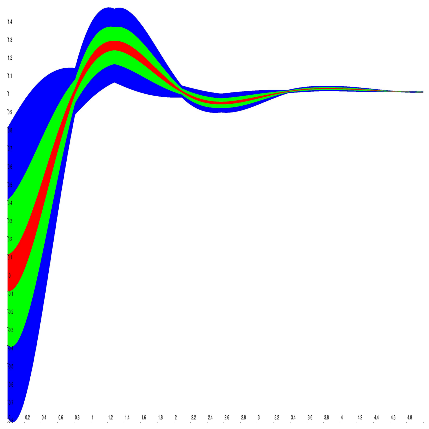
Settings for DynIbex. Maximum zonotope order is set to 60, reachability analysis is carried out with an (absolute and relative) error tolerance of 10^{-8} using an explicit Runge-Kutta method of order 2 (Heun's method). No splitting of the initial state has been performed. With $\Delta = 0.8$, the limit 1.4 is violated.



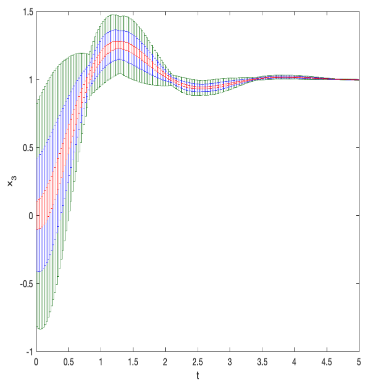
(a) Ariadne



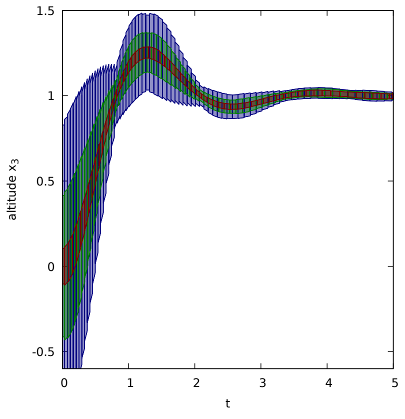
(b) CORA



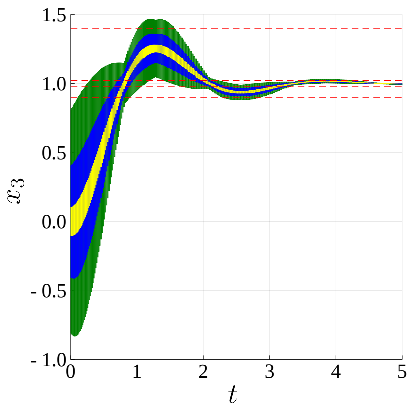
(c) DynIbex



(d) Flow*



(e) Isabelle/HOL



(f) JuliaReach

Figure 4: Reachable set overapproximations for QUAD20 on the $t - x_3$ axes.

Settings for Flow*. The tool uses the same setting for all of the 3 tests. The stepsize is fixed at 0.025, and the TM order is selected by the tool from the range of 3 to 8 according to the remainder estimation $[-0.001, 0.001]$, and the cutoff threshold is 10^{-6}. There is no need to turn on the symbolic remainder feature.

Settings for Isabelle/HOL. Maximum Zonotope order is set to 25. Reachability analysis is carried out with an (absolute and relative) error tolerance of 2^{-10}. $\Delta = 0.8$ violates the limit 1.4.

Settings for JuliaReach. We used $n_Q = 1$, $n_T = 5$ and an absolute tolerance of 10^{-7} for all cases.

3.5 Lotka–Volterra with tangential crossings benchmark (LOVO20)

3.5.1 Model

The benchmark described below refers to the Lotka-Volterra equations, or predator-prey equations, which are well-known in the literature.

The system is defined as follows:

$$\begin{cases} \dot{x} = 3x - 3xy \\ \dot{y} = xy - y \end{cases} \tag{2}$$

which produces cyclic trajectories around the equilibrium point $(1, 1)$ dependent on the initial state.

We are interested to see how this nonlinear dynamics plays with a nonlinear guard, whose boundary is:

$$\sqrt{(x-1)^2 + (y-1)^2} = 0.15 \tag{3}$$

which is a circle of radius 0.15 around the equilibrium.

By choosing an initial state $I = (1.3, 1.0)$ the cycle has a period of approximately 3.64 time units. The trajectory of the Lotka–Volterra system trajectory is almost tangent (externally) to the guard circle. Hence, small variations in the width of the initial set would put the trajectory partially within the guard. Consequently we are interested in evaluating the time spent within the circle by introducing a continuous counter variable called *cnt*.

The corresponding hybrid automaton is used to model the system:

- Continuous variables: x, y and *cnt*;
- Locations: *outside* and *inside*;
- Dynamics: those from Eq. 2 for x, y in both locations, and

$$\dot{cnt} = \begin{cases} 0, \text{ in } \textit{outside} \\ 1, \text{ in } \textit{inside} \end{cases} \tag{4}$$

- Guards:

$$\begin{cases} (x - Q_x)^2 + (y - Q_y)^2 \leq R^2 \text{ from } \textit{outside} \text{ to } \textit{inside} \\ (x - Q_x)^2 + (y - Q_y)^2 \geq R^2 \text{ from } \textit{inside} \text{ to } \textit{outside} \end{cases} \tag{5}$$

- Invariants: the complement of the corresponding guards (i.e., transitions are urgent);
- Resets: none, i.e., the identity for both transitions.

3.5.2 Analysis

We want to start the system from $I = (1.3 \pm \epsilon, 1.0)$, with $\epsilon = 0.008$, and evolve it for $T = 3.64$ time units. Since the original system was close to tangency, by enlarging the initial set we expect to produce different sequences of discrete events due to the distinction between crossing and not crossing, and possibly by distinguishing the crossing sets based on the different crossing times.

The following two properties must be verified:

1. At least one final set must not cross the guard, and at least one final set must cross it twice by entering and exiting the reference circle;
2. $cnt < 0.2$ holds for all final sets.

Table 5: Results of LOVO20 in terms of computation time, area and cnt_{max}.

tool	**computation time in** [s]	**area**	cnt_{max}
Ariadne	3.9	0.052	0.18
CORA	12	0.0025	0.2
DynIbex	10	$7.6e^{-5}$	0.132
Flow*	–	–	–
Isabelle/HOL	–	–	–
JuliaReach	7.1	0.0022	0.182

3.5.3 Evaluation

In terms of metrics, it is required to supply the following:

1. The execution time for computing the reachable set and checking the properties;
2. The area $x \times y$ of the box hull enclosing all the final sets;
3. The maximum of the *cnt* value obtained from all the final sets.

In addition, a figure showing the reachable set along with the circular guard shall be provided. The axes are $[0.6, 1.4] \times [0.6, 1.4]$.

3.5.4 Results

This benchmark produced polarizing results: those tools that could handle it easily with good performance, and those that were not able to handle the hybrid nature of the problem. Table 5 gives the timing/quality results, while Fig. 5 shows the graphical output. A future improvement could be to focus on the section of the figure that displays the crossing and the subsequent trajectories.

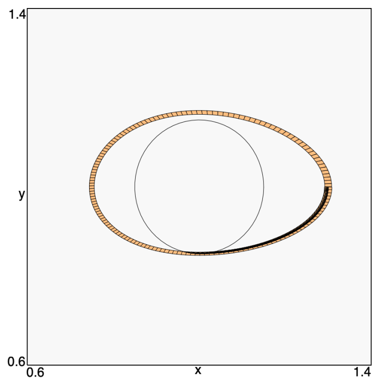
Settings for Ariadne. A TaylorPicardIntegrator is used with a maximum spacial error of $1e-5$. The maximum step size is 0.09.

Settings for CORA. We use the approach in [33] to calculate the intersections with the non-linear guard set. For continuous reachability we apply the conservative linearization approach [9] with time step size of 0.005 and a zonotope order of 20 for all modes.

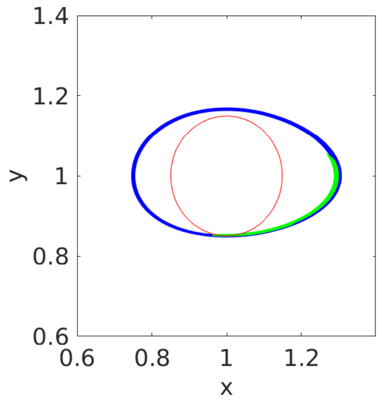
Settings for DynIbex. The library DynIbex does not support hybrid systems natively. However, based on constraint programming, event detection can be implemented and hybrid systems can be simulated. Reachability analysis is carried out with an error tolerance of 10^{-12} using an explicit Runge-Kutta method of order 4 (RK4 method). No splitting of the initial state has been performed.

Settings for Flow*. Since Flow* does not support urgent discrete transitions in hybrid systems, we skip the test on this benchmark.

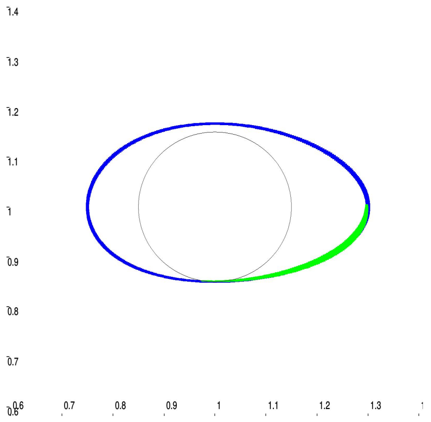
Settings for Isabelle/HOL. Isabelle/HOL does not support hybrid systems automatically.



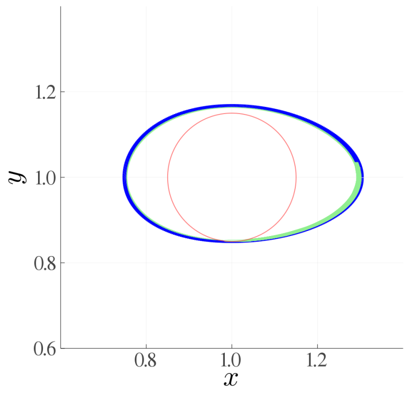
(a) Ariadne



(b) CORA



(c) DynIbex



(d) JuliaReach

Figure 5: Reachable set overapproximation for LOVO20, with $x, y \in [0.6, 1.4]$, where the circular guard is shown.

Settings for JuliaReach. The invariants and nonlinear guard are handled by suitable polygonal under- and over-approximations of the ball (resp. their complements). We used $n_T = 7$ and $n_Q = 1$ and split the initial set into 5 boxes.

3.6 Space rendezvous benchmark (SPRE20)

3.6.1 Model

Spacecraft rendezvous is a perfect use case for formal verification of hybrid systems with nonlinear dynamics since mission failure can cost lives and is extremely expensive. This benchmark is taken from [20]. A version of this benchmark with linearized dynamics is verified in the ARCH-COMP category *Continuous and Hybrid Systems with Linear Continuous Dynamics*. The nonlinear dynamic equations describe the two-dimensional, planar motion of the spacecraft on an orbital plane towards a space station:

$$\begin{cases} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{v_x} &= n^2 x + 2nv_y + \frac{\mu}{r^2} - \frac{\mu}{r_c^3}(r+x) + \frac{u_x}{m_c} \\ \dot{v_y} &= n^2 y - 2nv_x - \frac{\mu}{r_c^3} y + \frac{u_y}{m_c} \end{cases}$$

The model consists of position (relative to the target) x, y [m], time t [min], as well as horizontal and vertical velocity v_x, v_y [m / min]. The parameters are $\mu = 3.986 \times 10^{14} \times 60^2$ [m^3 / min^2], $r = 42164 \times 10^3$ [m], $m_c = 500$ [kg], $n = \sqrt{\frac{\mu}{r^3}}$ and $r_c = \sqrt{(r+x)^2 + y^2}$.

The hybrid nature of this benchmark originates from a switched controller. In particular, the modes are *approaching* ($x \in [-1000, -100]$ [m]), *rendezvous attempt* ($x \geq -100$ [m]), and *aborting*. A transition to mode *aborting* occurs nondeterministically at $t \in [120, 150]$ [min]. The linear feedback controllers for the different modes are defined as $\binom{u_x}{u_y} = K_1 \underline{x}$ for mode *approaching*, and $\binom{u_x}{u_y} = K_2 \underline{x}$ for mode *rendezvous attempt*, where $\underline{x} = \begin{pmatrix} x & y & v_x & v_y \end{pmatrix}^T$ is the vector of system states. The feedback matrices K_i were determined with an LQR-approach applied to the linearized system dynamics, which resulted in the following numerical values:

$$K_1 = \begin{pmatrix} -28.8287 & 0.1005 & -1449.9754 & 0.0046 \\ -0.087 & -33.2562 & 0.00462 & -1451.5013 \end{pmatrix}$$

$$K_2 = \begin{pmatrix} -288.0288 & 0.1312 & -9614.9898 & 0 \\ -0.1312 & -288 & 0 & -9614.9883 \end{pmatrix}$$

In the mode *aborting*, the system is uncontrolled $\binom{u_x}{u_y} = \binom{0}{0}$.

3.6.2 Analysis

The spacecraft starts from the initial set $x \in [-925, -875]$ [m], $y \in [-425, -375]$ [m], $v_x = 0$ [m/min] and $v_y = 0$ [m/min]. For the considered time horizon of $t \in [0, 200]$ [min], the following specifications have to be satisfied:

- **Line-of-sight:** In mode *rendezvous attempt*, the spacecraft has to stay inside line-of-sight cone $\mathcal{L} = \{\binom{x}{y} \mid (x \geq -100) \wedge (y \geq x\tan(30°)) \wedge (-y \geq x\tan(30°))\}$.
- **Collision avoidance:** In mode *aborting*, the spacecraft has to avoid a collision with the target, which is modeled as a box $\mathcal{B}$ with 0.2m edge length and the center placed at the origin.
- **Velocity constraint:** In mode *rendezvous attempt*, the absolute velocity has to stay below 3.3 [m/min]: $\sqrt{v_x^2 + v_y^2} \leq 3.3$ [m/min].

Remark on velocity constraint In the original benchmark [20], the constraint on the velocity was set to 0.05 m/s, but it can be shown (by a counterexample) that this constraint cannot be satisfied. We therefore use the relaxed constraint 0.055 [m/s] = 3.3 [m/min].

3.6.3 Evaluation

The computation time for evolution and verification is provided. A figure is shown in the (x, y) axes, with $x \in [-1000, 200]$ and $y \in [-450, 0]$.

Table 6: Results of SPRE20 in terms of computation time.

tool	**computation time in [s]**
Ariadne	–
CORA	29
DynIbex	143
Flow*	8.9
Isabelle/HOL	–
JuliaReach	20

3.6.4 Results

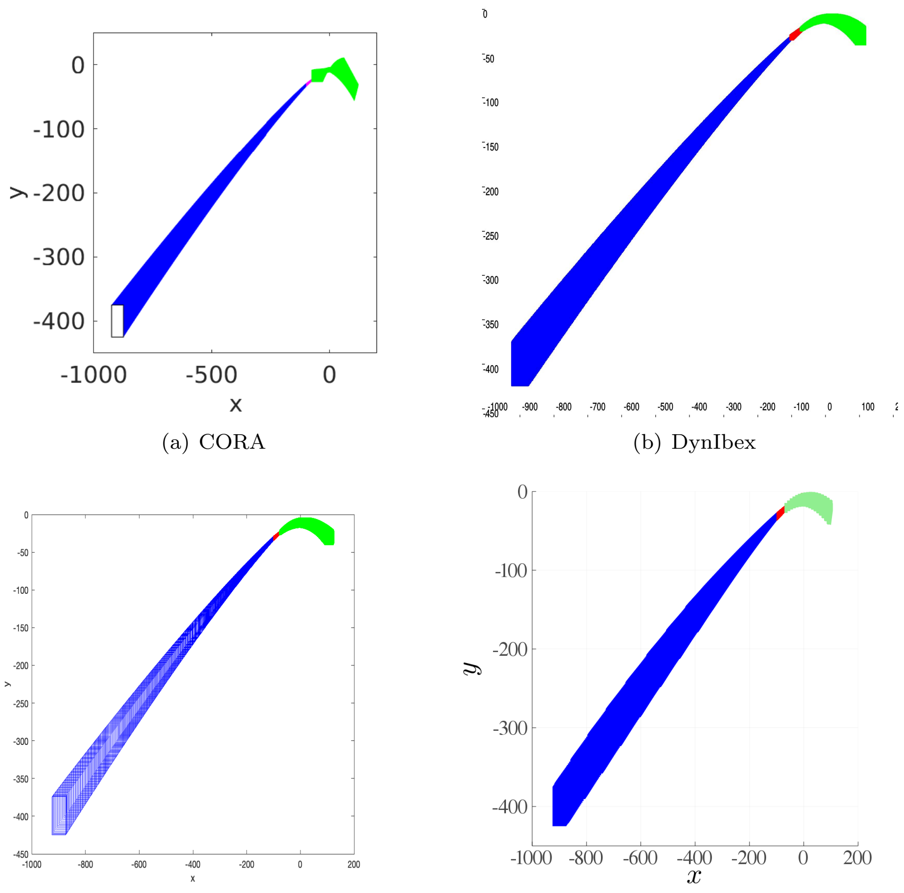
The results of the reachability computation for the spacecraft rendezvous model are given in Figure 6 and Table 6, with the tool settings below. The introduction of a permissive guard prevented completion for Ariadne: too many trajectories were generated and the absence of a recombination strategy proved an issue. Therefore this benchmark requires proper support of crossings in the presence of large sets, even if the crossing region is very simple from a geometrical viewpoint. The hybrid nature of the problem again was an obstacle for Isabelle/HOL.

Settings for Ariadne. Ariadne was not able to complete evolution, due to the extremely large number of trajectories produced from the nondeterministic guard: this is caused by the lack of a recombination strategy. The maximum step size used was 1.0, essentially meaning that we allowed the step size to vary widely along evolution: this choice turned out to be preferable in terms of execution time. The maximum temporal order was 4 and the maximum spacial error enforced for each step equal is 10^{-3}. A splitting strategy for the initial set was used; the strategy compare the radius of the set with a reference value of 12.0, in order to split the first two dimensions once and yield a total of 4 initial subsets.

Settings for CORA. CORA was run with a time step size of 0.2 [min] for the modes *approaching* and *aborting*, and with a time step size of 0.05 [min] for mode *rendezvous attempt*. The intersections with the guard sets are calculated with constrained zonotopes [39], and the intersection is then enclosed with a zonotope bundle [8]. In order to find suitable orthogonal directions for the enclosure *principal component analysis* is applied.

Settings for DynIbex. The library DynIbex does not support hybrid systems natively. However, based on constraint programming, event detection can be implemented and hybrid systems can be simulated. Maximum zonotope order is set to 10, reachability analysis is carried out with an error tolerance of 10^{-6} using an explicit Runge-Kutta method of order 3 (Kutta's method). No splitting of the initial state has been performed.

Settings for Flow*. The computation setting for Flow* is given as follows. For the mode *approaching*, the time stepsize is 0.2 and the TM order is 4. For the mode *rendezvous attempt*, we use a smaller time stepsize which is 0.02, and also a lower TM order which is 3. For the last mode, we use the stepsize 0.1, the TM order 3 and track the remainder symbolically every 200 steps. Besides, the cutoff threshold is 10^{-7} in all modes.



(a) CORA (b) DynIbex

(c) Flow* (d) JuliaReach

Figure 6: Reachable set of the spacecraft position in the x-y-plane for SPRE20. CORA, Isabelle/HOL, and DynIbex use different colors to encode different modes of the hybrid system.

Settings for Isabelle/HOL. Isabelle/HOL does not support hybrid systems automatically.

Settings for JuliaReach. We used $n_Q = 1$ and $n_T = 5$. The transition to the aborting mode is handled by clustering the flowpipe into 29 boxes.

4 Conclusion and Outlook

This year, the competition confirmed the same six participants from 2019. This gave us the opportunity to raise the bar of the benchmarks, by introducing two new systems (one in the continuous space, the other in the hybrid space) and by increasing the difficulty of three of the existing systems.

The new benchmarks were the *production-destruction* system (PRDE20) and the *Lotka-Volterra* system with tangential crossing (LOVO20). The first one was handled easily by all tools, with significantly different quality results. For the next year, we aim at making results better comparable, possibly by setting a target volume to be reached. LOVO20 turned out to be problematic where handling tangential crossings was not supported in the tool; tools that handled it were able to reach completion in a short time. We think that the benchmark is interesting, but the complexity could be increased next year. Again, a target area could also be considered as an objective.

Regarding the variation of the existing benchmarks, the *coupled Van der Pol* oscillator (CVDP20) turned out to be a challenge for most competitors, in some cases preventing the satisfaction of the specification. This was an interesting experiment and we shall iterate on it next year, to evaluate improvements of tools. The *quadrotor* system (QUAD20), on the other hand, did not show any interesting property after the extension to different inputs. Since the benchmark is easily handled by all tools, we might consider its exclusion from next year's competition: while it is high-dimensional and potentially highly nonlinear, it does not particularly exploit such nonlinearity. The steady-state behavior at the end of the evolution was a good benchmark for the stability of the integration scheme, but it has been supplanted by the Production-Destruction benchmark this year.

The extension of the Space Rendezvous benchmark to a non-urgent transition proved to be unmanageable by tools that did not perform a set recombination strategy, but was reasonably handled otherwise. The benchmark therefore remains an interesting challenge peculiar to the hybrid domain.

Finally, the Laub-Loomis benchmark was kept the same as last year due to its difficulty. Tools showed improvements, being all able to return results with a satisfactory quality. For the next year, an extension is therefore envisioned.

We care to mention that, triggered by the participation in this competition, individual tools made progress:

- A new set representation called *sparse polynomial zonotopes* [32] was implemented in CORA which enables the faster and more accurate computation of the reachable set for nonlinear systems. Furthermore, CORA now supports hybrid systems with nonlinear guard sets [33].

- Most of the fundamental data structures and functions in Flow* are re-implemented in a more efficient way and exposed as a C++ API. The purpose of the re-implementation is not only the improvement of the code organization and efficiency, but also the ease of extension. In the near future, the tool will be extended in the following two directions: (a) as a core component in a verification tool for nonlinear neural network controlled systems. The effectiveness of Flow* in the verification of AI systems has already been shown in [28, 29]. (b) easier to be tailored to build into embedded systems. We have successfully run Flow* on a mobile device to control a self-driving robot.

- JuliaReach has a new user API, `ReachabilityAnalysis.jl`, with many improved features such as more performant code, a simplified interface, and new algorithms for continuous and hybrid systems. Moreover, we have incorporated new convex approximation methods for Taylor models for enhanced precision. We also added support for multi-threaded parallelism.

Summarizing, the new benchmarks and extensions were interesting by way of touching critical aspects of continuous/hybrid evolution that were not considered in the previous years. We believe that a benchmark suite with representative problems is of the utmost importance, in order to stimulate meaningful progress of all the participating tools. Consequently, for the next year we aim at refining the existing suite to advance in this direction.

5 Acknowledgments

Luca Geretti acknowledges the support from MIUR, Project ”Italian Outstanding Departments, 2018-2022” and by INDAM, GNCS 2020, ”Strategic Reasoning and Automated Synthesis of Multi-Agent Systems”.

Luis Benet acknowledges support from PAPIIT grant IG-100819.

Xin Chen acknowledges the support from Air Force Research Laboratory under the contract FA8650-16-C-2642, Thrust 3.

Fabian Immler: This material is based upon work supported by the Air Force Office of Scientific Research under grant number FA9550-18-1-0120. Any opinions, finding, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force.

David Sanders acknowledges support from PAPIIT grant IN-117117.

Christian Schilling acknowledges support in part by the Austrian Science Fund (FWF) under grant Z211-N23 (Wittgenstein Award) and the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411.

Matthias Althoff acknowledges support by the European Commission project justITSELF under grant number 817629.

Julien Alexandre dit Sandretto: this work was supported by the “Chair Complex Systems Engineering - Ecole polytechnique, THALES, DGA, FX, Dassault Aviation, DCNS Research, ENSTA Paris, Télécom Paris, and Fondation ParisTech” and partially supported by DGA AID.

References

[1] *”Amazon EC2 G4 Instance”*, (accessed July 10, 2020). https://aws.amazon.com/it/blogs/aws/now-available-ec2-instances-g4-with-nvidia-t4-tensor-core-gpus/.

[2] Julien Alexandre dit Sandretto and Alexandre Chapoutot. Validated Explicit and Implicit Runge-Kutta Methods. *Reliable Computing electronic edition*, 22, 2016.

[3] Julien Alexandre dit Sandretto and Alexandre Chapoutot. Validated Simulation of Differential Algebraic Equations with Runge-Kutta Methods. *Reliable Computing electronic edition*, 22, 2016.

[4] Julien Alexandre Dit Sandretto, Alexandre Chapoutot, and Olivier Mullier. Constraint-Based Framework for Reasoning with Differential Equations. In Çetin Kaya Koç, editor, *Cyber-Physical Systems Security*, pages 23–41. Springer International Publishing, December 2018.

[5] M. Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Hybrid Systems: Computation and Control*, pages 173–182, 2013.

[6] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[7] M. Althoff and D. Grebenyuk. Implementation of interval arithmetic in CORA 2016. In *Proc. of the 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 91–105, 2016.

[8] M. Althoff and B. H. Krogh. Zonotope bundles for the efficient computation of reachable sets. In *Proc. of the 50th IEEE Conference on Decision and Control*, pages 6814–6821, 2011.

[9] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proc. of the 47th IEEE Conference on Decision and Control*, pages 4042–4048, 2008.

[10] Miguel Angel Barron. Stability of a ring of coupled van der pol oscillators with non-uniform distribution of the coupling parameter. In *Journal of applied research and technology 14.1*, pages 62–66, 2016.

[11] R. Beard. Quadrotor dynamics and control rev 0.1. Technical report, Brigham Young University, 2008.

[12] Luis Benet and David P. Sanders. TaylorSeries.jl: Taylor expansions in one and several variables in julia. *Journal of Open Source Software*, 4(36):1043, April 2019.

[13] Luis Benet and David P. Sanders. JuliaDiff/TaylorSeries.jl. `https://github.com/JuliaDiff/TaylorSeries.jl`, March 2020.

[14] Luis Benet and David P. Sanders. JuliaIntervals/IntervalArithmetic.jl. `https://github.com/JuliaIntervals/IntervalArithmetic.jl`, March 2020.

[15] Luis Benet and David P. Sanders. JuliaIntervals/TaylorModels.jl. `https://github.com/JuliaIntervals/TaylorModels.jl`, March 2020.

[16] L. Benvenuti, D. Bresolin, P. Collins, A. Ferrari, L. Geretti, and T. Villa. Assume-guarantee verification of nonlinear hybrid systems with Ariadne. *Int. J. Robust. Nonlinear Control*, 24(4):699–724, 2014.

[17] M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4:361–369, 1998.

[18] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[19] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling. JuliaReach: a toolbox for set-based reachability. In *HSCC*, 2019.

[20] N. Chan and S. Mitra. Verifying safety of an autonomous spacecraft rendezvous mission. In *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems, collocated with Cyber-Physical Systems Week (CPSWeek) on April 17, 2017 in Pittsburgh, PA, USA*, pages 20–32, 2017.

[21] X. Chen. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models*. PhD thesis, RWTH Aachen University, 2015.

[22] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Proc. of RTSS'12*, pages 183–192. IEEE Computer Society, 2012.

[23] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of CAV'13*, volume 8044 of *LNCS*, pages 258–263. Springer, 2013.

[24] X. Chen and S. Sankaranarayanan. Decomposed reachability analysis for nonlinear systems. In *Proc. of RTSS'16*, pages 13–24. IEEE Computer Society, 2016.

[25] P. Collins, D. Bresolin, L. Geretti, and T. Villa. Computing the evolution of hybrid systems using rigorous function calculus. In *Proc. of the 4th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS12)*, pages 284–290, Eindhoven, The Netherlands, June 2012.

[26] Catherine Daramy-Loirat, Florent De Dinechin, David Defour, Matthieu Gallet, Nicolas Gast, and Christoph Lauter. CR-LIBM: A library of correctly rounded elementary functions in double-precision. Technical report, ENS de Lyon, 2010.

[27] Vincent Drevelle and Jeremy Nicola. Vibes: A visualizer for intervals and boxes. *Mathematics in Computer Science*, 8(3):563–572, Sep 2014.

[28] S. Dutta, X. Chen, and S. Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019*, pages 157–168. ACM, 2019.

[29] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Trans. Embedded Comput. Syst.*, 18(5s):106:1–106:22, 2019.

[30] F. Immler. Verified reachability analysis of continuous systems. In *Proc. of TACAS'15*, volume 9035 of *LNCS*, pages 37–51. Springer, 2015.

[31] F. Immler and J. Hölzl. Ordinary differential equations. *Archive of Formal Proofs*, July 2018. `http://isa-afp.org/entries/Ordinary_Differential_Equations.shtml`, Formal proof development.

[32] N. Kochdumper and M. Althoff. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *arXiv preprint arXiv:1901.01780*, 2019.

[33] N. Kochdumper and M. Althoff. Reachability analysis for hybrid systems with nonlinear guard sets. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020.

[34] Stefan Kopecz and Andreas Meister. On order conditions for modified patankar–runge–kutta schemes. *Applied Numerical Mathematics*, 123:159–179, 2018.

[35] M. T. Laub and W. F. Loomis. A molecular network that produces spontaneous oscillations in excitable cells of dictyostelium. *Molecular Biology of the Cell*, 9:3521–3532, 1998.

[36] Olivier Mullier, Alexandre Chapoutot, and Julien Alexandre dit Sandretto. Validated computation of the local truncation error of runge–kutta methods with automatic differentiation. *Optimization Methods and Software*, 33(4-6):718–728, 2018.

[37] Jorge A. Pérez-Hernández and Luis Benet. PerezHz/TaylorIntegration.jl. `https://github.com/PerezHz/TaylorIntegration.jl`, February 2020.

[38] Christian Schilling and Marcelo Forets. JuliaReach/LazySets.jl: v1.36.0. `https://github.com/JuliaReach/LazySets.jl`, March 2020. Accessed: 2020-03-05.

[39] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69:126–136, 2016.

[40] R. Testylier and T. Dang. Nltoolbox: A library for reachability computation of nonlinear dynamical systems. In *Proc. of ATVA'13*, volume 8172 of *LNCS*, pages 469–473. Springer, 2013.

[41] K. Weihrauch. *Computable analysis.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2000.