EPiC Series in Computing
Volume 61, 2019, Pages 41–61
ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems

ARCH-COMP19 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics

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

[1] Computer Science Department, Carnegie Mellon University, United States
fimmler@cs.cmu.edu
[2] Technische Universität München, Munich, Germany
althoff@in.tum.de,niklas.kochdumper@tum.de
[3] Instituto de Ciencias Físicas, Universidad Nacional Autónoma de México (UNAM), México
benet@icf.unam.mx
[4] ENSTA ParisTech, Palaiseau, France
alexandre.chapoutot@ensta-paristech.fr
[5] University of Dayton, Dayton, OH, United States
xchen4@udayton.edu
[6] Universidad de la República, Uruguay
mforets@gmail.com
[7] University of Verona, Verona, Italy
luca.geretti@univr.it
[8] Departamento de Física, Facultad de Ciencias Físicas, Universidad Nacional Autónoma de México (UNAM), México
dpsanders@ciencias.unam.mx
[9] 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 2019. In this year, 6 tools Ariadne, CORA, DynIbex, Flow*, Isabelle/HOL, and JuliaReach (in alphabetic order) participated. They are applied to solve reachability analysis problems on four benchmark problems, one of them with 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—not all of the specificities 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 [14] containers at gitlab.com/goranf/ARCH-COMP.

In this report, we summarize the results of the second 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 $\vec{x} \in X_0$ as well as an unsafe set U, we apply the participating tools to prove that there is no state reachable contained in U over 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 Ariadne, CORA, DynIbex, Flow*, Isabelle/HOL, and JuliaReach participate in solving the safety problems defined on three continuous and one hybrid benchmark. The continuous benchmarks are the Van der Pol oscillator, the Laub-Loomis model, and a controlled quadrotor model. The hybrid benchmark models a space rendezvous.

The benchmarks are selected based on the discussions of the tool authors. Since the experimental results are produced on different platforms, we provide Section A for the hardware details.

2 Participating Tools

Ariadne. (Luca Geretti) *Ariadne* [21, 12] is a library based on *Computable Analysis* [31] that uses a rigorous numerical approach to all its algebraic, geometric and logical operations. In particular, it forces conservative rounding of all external and internal operations, in order to guarantee formal correctness of its results. 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`, while the version used for this competition is the development one available at `https://github.com/ariadne-cps/ariadne`.

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 multidimensional 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. CORA is available at http://www6.in.tum.de/Main/SoftwareCORA.

DynIbex. (Alexandre Chapoutot) It is 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 [1] and algebraic differential equations of index 1 [2], combined with numerical constraints on state variables and reachable tubes. It produces sound results taking into account round-off error in floating-point computations and truncation errors generated by numerical integration methods [28]. Moreover, constraint satisfaction problem algorithms offer a convenient approach to check properties on reachable tubes as explained in [3]. 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 methods, 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 [22] which is available at http://enstabretagnerobotics.github.io/VIBES/.

Flow*. (Xin Chen) The tool Flow* [19, 17] 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 [13], 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. For the discrete jumps of hybrid systems, Flow* uses the techniques of domain contraction and range overapproximation to compute flowpipe/guard intersections [18], and then aggregates them by a box or parallelotope. Besides, in order to reduce the accumulation of overestimation during an integration job, the tool can symbolically represent the remainders of the previous N flowpipes for some $N > 0$ (see [20]). In order to produce guaranteed results, the tool includes all roundoff errors in the TM remainders during reachability computation. This year, we entirely updated the module to handle continuous dynamics in Flow*, and all of the continuous benchmarks will be tested on it.

Isabelle/HOL-ODE-Numerics. (Fabian Immler) HOL-ODE-Numerics [24, 25] 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) JuliaReach [15] is a toolbox for reachability computations of dynamical systems, available at `http://juliareach.org`. For nonlinear reachability we combine functionality from TaylorModels.jl [11], TaylorSeries.jl [10] and TaylorIntegration.jl [29]. First, we compute 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 box of deviations from the central initial condition, to yield an over-approximation. The approach is (numerically) sound due to rigorous interval bounds in the Taylor approximation.

Table 1: Results of the Van der Pol Oscillator. Details of the platforms are described in Section A.

	computation time in [s]		**platform**	
tool	$\mu = 1$	$\mu = 2$	**language**	**machine**
Ariadne	2.5	33	C++	$\mathrm{M}_{\mathrm{Ariadne}}$
CORA	2.3	2.8	MATLAB	$\mathrm{M}_{\mathrm{CORA}}$
DynIbex	16	653	C++	$\mathrm{M}_{\mathrm{DynIbex}}$
Flow*	0.4	8.7	C++	$\mathrm{M}_{\mathrm{Flow*}}$
Isabelle/HOL	1.4	2.0	SML	$\mathrm{M}_{\mathrm{Isabelle}}$
JuliaReach	0.7	5.9	Julia	$\mathrm{M}_{\mathrm{JuliaReach}}$

3 Benchmarks

3.1 Van der Pol Oscillator

3.1.1 Model

The Van der Pol oscillator was introduced by the Dutch physicist Balthasar van der Pol. It can be defined by the following ODE with 2 variables.

$$\begin{cases} \dot{x} & = & y \\ \dot{y} & = & \mu(1 - x^2)y - x \end{cases}$$

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

3.1.2 Specification

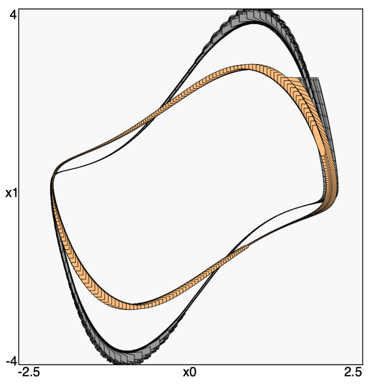
We consider $\mu = 1$ and $\mu = 2$.

$\mu = 1$: For $\mu = 1$ we set the initial condition $x(0) \in [1.25, 1.55]$, $y(0) \in [2.35, 2.45]$, which was used in [4]. The unsafe set is given by $y \geq 2.75$ for a time horizon of $[0, 7]$.

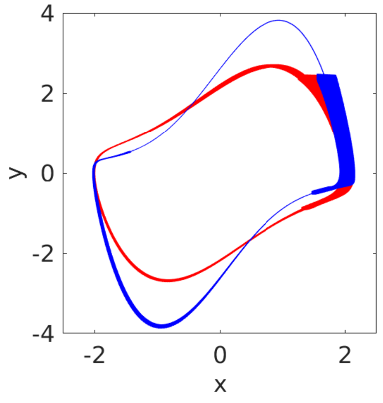
$\mu = 2$: For $\mu = 2$ we set the initial condition $x(0) \in [1.55, 1.85]$, $y(0) \in [2.35, 2.45]$, which is the same size as before, but on the limit cycle for $\mu = 2$. The unsafe set is given by $y \geq 4.0$ for a time horizon of $[0, 8]$.

3.1.3 Results

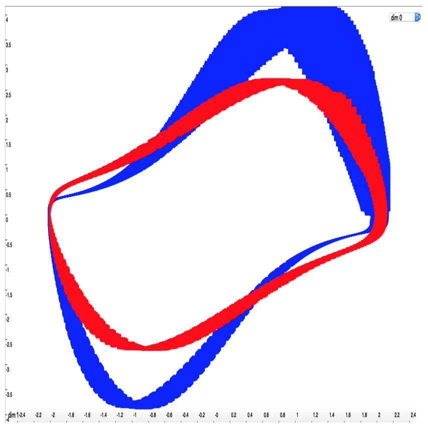
We observe that $\mu = 2$ is harder for all of the tools. Ariadne, DynIbex, Flow*, and JuliaReach deal with this difficulty by splitting the initial set (more). CORA and Isabelle/HOL introduce additional pseudo-invariants. The time costs of the participating tools on the Van der Pol oscillator benchmark are given in Table 1, and the plots of the overapproximation sets are presented in Figure 1. We also provide the computational settings of the tools as below.



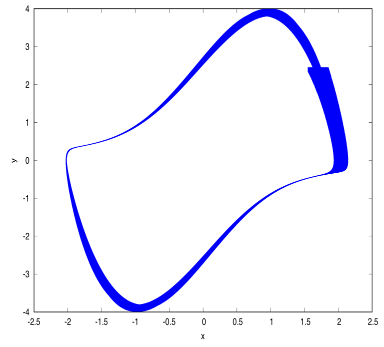
(a) Ariadne.



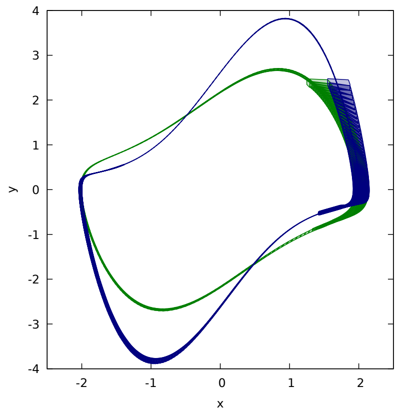
(b) CORA.



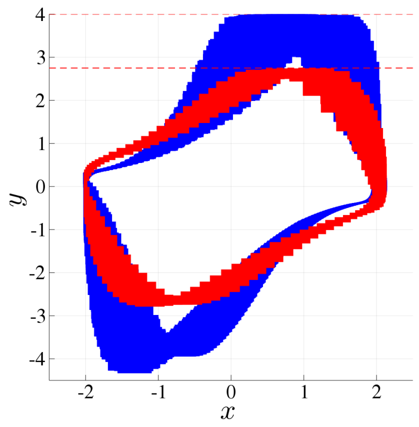
(c) DynIbex.



(d) Flow* ($\mu = 2$.



(e) Isabelle/HOL.



(f) JuliaReach.

Figure 1: Reachable set overapproximations for the Van der Pol oscillator. $x \in [-2.5, 2.5]$, $y \in [-4, 4]$.

Setting for Ariadne. For $\mu = 1$, the maximum step size used is 0.02, with a maximum temporal order of 5. The maximum spacial error enforced for each step is $2 \cdot 10^{-4}$. For $\mu = 2$, the maximum step size used is 0.04, a maximum temporal order of 5 and a maximum spacial error enforced for each step equal to $2 \cdot 10^{-4}$; in this case we also split the initial set into 16 subsets by enforcing a maximum enclosure radius of 0.3. While for $\mu = 2$ the plotting (in dark gray) appears to cross the $y = 4$ boundary, in fact it is overapproximated graphically and the reachable sets are proved to respect $y \leq 3.96$.

Setting for CORA. For $\mu = 1$, a pseudo invariant at $x = 1.5$ was introduced manually. For $\mu = 2$, two pseudo invariants at $x = 1.8$ and $x = -1.8$ were introduced manually. CORA uses the time step size 0.01 and the zonotope order 20 for both cases $\mu = 1$ and $\mu = 2$.

Setting for DynIbex. Maximum zonotope order is set to 20, reachability analysis is carried out with an (absolute and relative) error tolerance of 10^{-7} using an implicit midpoint Runge-Kutta method of order 2. For $\mu = 1$ a partition of the initial state in 16 sub-boxes is performed. For $\mu = 2$ a partition of the initial state in 256 sub-boxes has been considered.

Setting for Flow*. For the model with $\mu = 1$, Flow* uses the following setting: a fixed stepsize of 0.04, an adaptive order from 4 to 8, the cutoff threshold is set to be 10^{-6}, the remainder estimation is 10^{-5} in all dimensions, the remainder queue size is 100, and the precision is 100. All roundoff errors are included in the TM flowpipes. The test case for $\mu = 2$ is more challenging, we used a fixed order of 6 along with an adaptive stepsize from 0.001 to 0.04, the cutoff threshold is 10^{-8}, and we increase the remainder queue size to 1000. In order to prove the flowpipes are in the range of $y < 4$, we equally divide the initial set in the dimension of x to 6 pieces. Since Flow* does not provide the function to overlap grid plots, we show the flowpipes in the second test in Figure 1(d).

Setting for Isabelle/HOL. Maximum Zonotope order is set to 20, Reachability analysis is carried out with an (absolute and relative) error tolerance of 2^{-14}. For $\mu = 1$ and $\mu = 2$, a pseudo-invariant is added at $x = 1.5$. The case $\mu = 2$ is more demanding, it requires an additional pseudo-invariant at $x = -1.5, y >= 0$.

Setting for JuliaReach. For $\mu = 1$, we used $n_Q = 2$, $n_T = 10$, and an absolute tolerance of 10^{-10}, which leads to an average step size ~ 0.06. For $\mu = 2$, in order to stay in the safe region, we split the set of initial states along the x-direction and used $n_Q = 1$, $n_T = 10$, and an absolute tolerance of 10^{-10}.

3.2 Laub-Loomis Benchmark

3.2.1 Model

The Laub-Loomis model is presented in [27] 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.

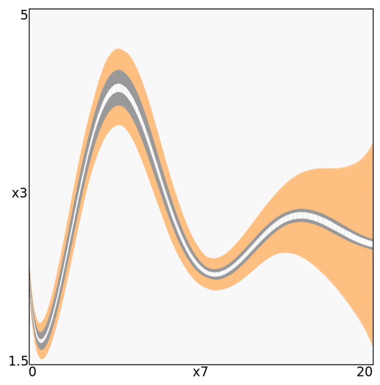
3.2.2 Specification

The initial sets are defined according to the one used in [30]. 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.

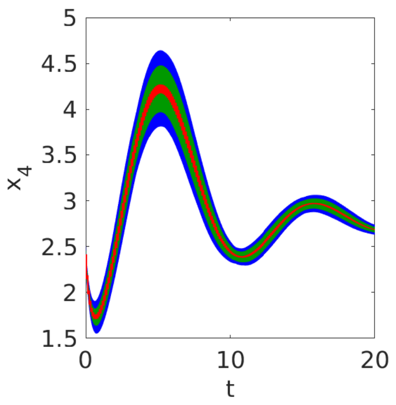
We consider $W = 0.01$, $W = 0.05$, and $W = 0.1$. For $W = 0.01$ and $W = 0.05$ we consider the unsafe region defined by $x_4 \geq 4.5$, while for $W = 0.1$, the unsafe set is defined by $x_4 \geq 5$. The time horizon for all cases is $[0, 20]$.

3.2.3 Results

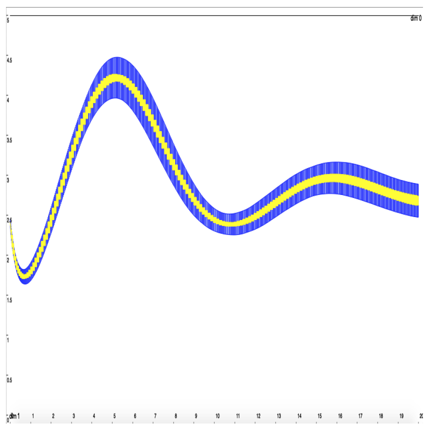
The computation results of the tools are given in Table 2. It can be seen that enlarging the initial set size can greatly make the reachability analysis task harder. The tool settings are given as below. Since the safety condition is only related to the variable x_4, we present the plots of projections of the overapproximations in the t-x_4 plane such that t is the time variable. Figure 2 shows the results.

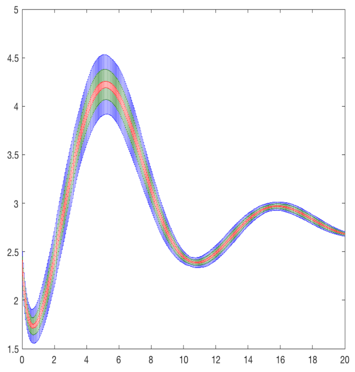


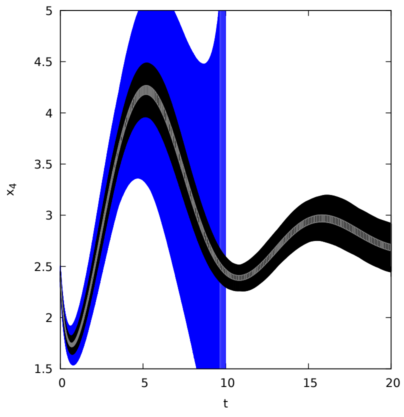
(a) Ariadne



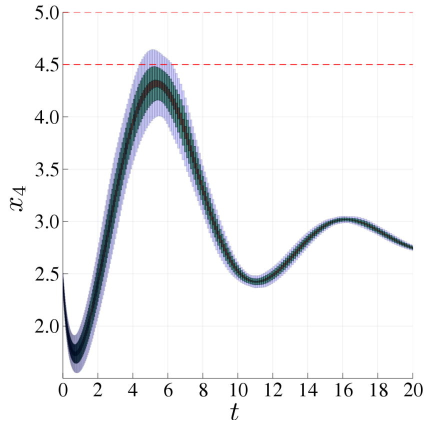
(b) CORA


(c) DynIbex


(d) Flow*



(e) Isabelle/HOL



(f) JuliaReach.

Figure 2: Reachable set overapproximations for the Laub-Loomis model (Overlayed plots for $W = 0.01$, $W = 0.05$, $W = 0.1$). $t \in [0, 20]$, $x_4 \in [1.5, 5]$.

Table 2: Results of the Laub-Loomis model. Details of the platforms are described in Section A.

	computation time in [s]			**platform**	
tool	$W = 0.01$	$W = 0.05$	$W = 0.1$	**language**	**machine**
Ariadne	7.7	1039	1317	C++	$\mathrm{M}_{\mathrm{Ariadne}}$
CORA	0.82	18	56	MATLAB	$\mathrm{M}_{\mathrm{CORA}}$
DynIbex	20	64	–	C++	$\mathrm{M}_{\mathrm{DynIbex}}$
Flow*	1.1	2.8	4.9	C++	$\mathrm{M}_{\mathrm{Flow*}}$
Isabelle/HOL	11	18	–	SML	$\mathrm{M}_{\mathrm{Isabelle}}$
JuliaReach	3.4	3.3	3.4	Julia	$\mathrm{M}_{\mathrm{JuliaReach}}$

	width of x_4 in final enclosure		
tool	$W = 0.01$	$W = 0.05$	$W = 0.1$
Ariadne	0.05	0.10	1.96
CORA	0.02	0.07	0.11
DynIbex	0.103	0.403	–
Flow*	0.0049	0.0253	0.0444
Isabelle/HOL	0.06	0.45	∞
JuliaReach	0.01	0.03	0.05

Setting for Ariadne. The maximum step size used is 0.2, the temporal order is 4 and a maximum spacial error enforced for each step equal to $2 \cdot 10^{-3}$. 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 0.03 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).

Setting 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 algorithm presented in [8] is executed. For the larger initial set $W = 0.1$, the more accurate but slower algorithm from [5] is used. 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 0.01 for $W = 0.1$. The maximum zonotope order for all initial sets is chosen as 50.

Setting 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^{-5} 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.

Setting for Flow*. For all of the 3 tests, Flow* uses a fixed stepsize of 0.05, an adaptive order from 3 to 8, and a cutoff threshold of 10^{-7}. The remainder estimation is 10^{-4} in all dimensions and we set the precision to 100 for floating-point numbers. For the initial set sizes $W = 0.01$ and $W = 0.05$, we set the remainder queue size to 100, while for $W = 0.1$, we raise the queue size to 200. The new version of Flow* shows a much better performance than the previous one which is presented last year, however the accuracy is nearly the same or even better.

Setting for Isabelle/HOL. Maximum Zonotope order is set to 60. Reachability analysis is carried out with an (absolute and relative) error tolerance of 2^{-12}. We do not perform subdivisions of the initial set. This results in failure to maintain precise enclosures for $W = 0.1$ and $t \geq 7$

Setting for JuliaReach. We used $n_Q = 1$, $n_T = 7$ and an absolute tolerance of 10^{-10} for the three cases.

3.3 Quadrotor Benchmark

3.3.1 Model

We study the dynamics of a quadrotor as derived in [9, 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 [9, 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

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

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

3.3.2 Specification

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 position of the quadrotor is uncertain in all directions within $[-0.4, 0.4]$ [m] and also the velocity is uncertain within $[-0.4, 0.4]$ [m/s] for all directions. All other values are initialized as 0.

3.3.3 Results

The results of the reachability computation for the quadrotor model are given in Figure 3 and Table 3. We give the tool settings below.

Table 3: Results of the quadrotor model. Details of the platforms are described in Section A.

tool	**computation time in [s]**	**language**	**machine**
Ariadne	7.0	C++	$\mathrm{M}_{\text{Ariadne}}$
CORA	4.3	MATLAB	M_{CORA}
DynIbex	108	C++	$\mathrm{M}_{\text{DynIbex}}$
Flow*	1.8	C++	$\mathrm{M}_{\text{Flow*}}$
Isabelle/HOL	27	SML	$\mathrm{M}_{\text{Isabelle}}$
JuliaReach	10.2	Julia	$\mathrm{M}_{\text{JuliaReach}}$

Setting for Ariadne. The maximum step size used is 0.01, with a maximum temporal order of 2. The maximum spacial error enforced for each step is 10^{-2}.

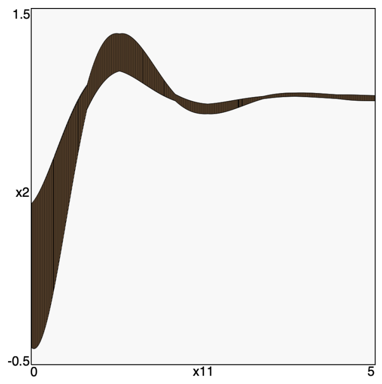
Setting for CORA. CORA uses the step size 0.1 and the zonotope order 50. The computation is carried out using the approach in [8], which conservatively linearizes the system dynamics for each consecutive time interval by adding the linearization error as an uncertain input. The linearization error is obtained using the Lagrange remainder, which are evaluated via interval arithmetic. This results in many function calls (especially for this example), whose overhead has been reduced since MATLAB R2015b. So the execution time for the quadrotor benchmark depends significantly on the MATLAB version (more than twice as fast since R2015b).

Setting for DynIbex. Maximum zonotope order is set to 25, reachability analysis is carried out with an (absolute and relative) error tolerance of 10^{-7} using an implicit midpoint Runge-Kutta method of order 2. No splitting of the initial state has been performed.

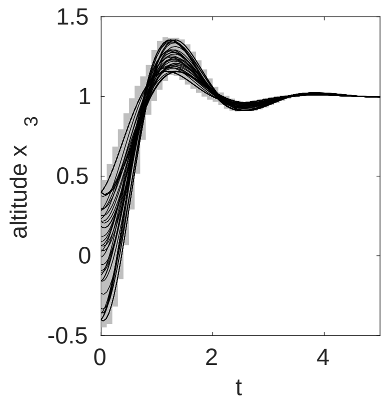
Setting for Flow*. Flow* uses the step size 0.025 and an adaptive TM order from 2 to 4. The cutoff threshold is 10^{-6} and the precision is set to be 53 for floating-point numbers. The TM flowpipe remainders are kept symbolically every 10 steps. All floating-point roundoff errors are included in the overapproximation. Figure 3(d) illustrates the interval overapproixmations for the flowpipes. To better evaluate the approximation error, we provide the maximum overapproximation error of the last flowpipe which is below 10^{-4}. Besides, the altitude at $t = 5$ is below 1 according to the computed flowpipes.

Setting 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}.

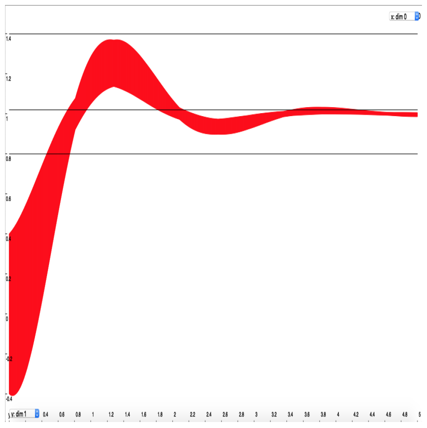
Setting for JuliaReach. We used $n_Q = 1$, $n_T = 5$ and an absolute tolerance of 10^{-7}.



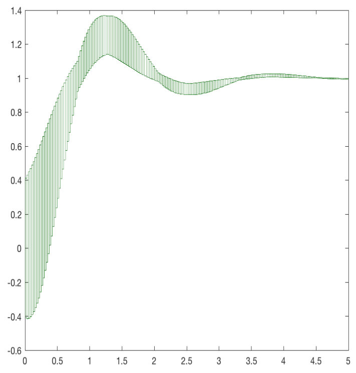
(a) Ariadne.



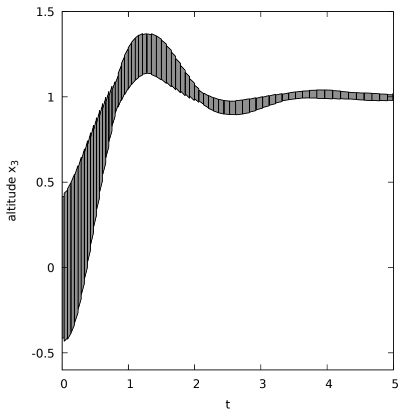
(b) CORA.



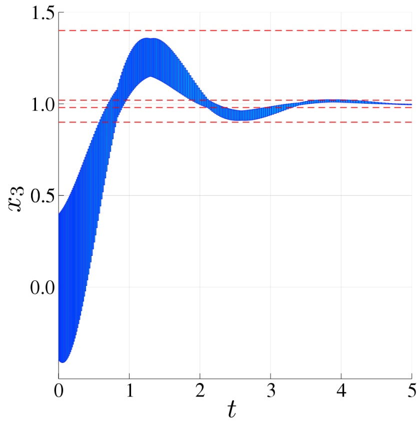
(c) DynIbex.



(d) Flow*.



(e) Isabelle/HOL.



(f) JuliaReach.

Figure 3: Reachable set overapproximations for the quadrotor model. CORA shows simulations in black.

3.4 Space Rendezvous Benchmark

3.4.1 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 [16]. 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:

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

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* (time $t \geq 120$ [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 $\binom{u_x}{u_y} = \binom{0}{0}$.

3.4.2 Specification

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].

Table 4: Results of the spacecraft rendezvous model. Details of the platforms are described in Section A.

tool	**computation time in [s]**	**language**	**machine**
Ariadne	172	C++	M_{Ariadne}
CORA	11.8	MATLAB	M_{CORA}
DynIbex	294	C++	M_{DynIbex}
Flow*	18.7	C++	$M_{\text{Flow*}}$
Isabelle/HOL	295	SML	M_{Isabelle}

Remark on velocity constraint In the original benchmark [16], 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].

3.4.3 Results

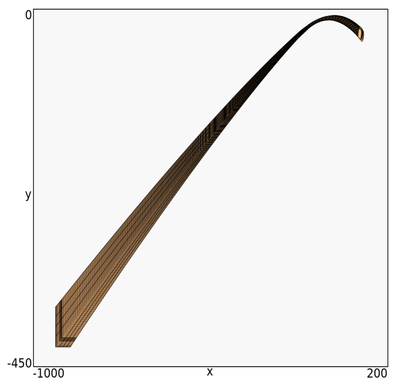
The results of the reachability computation for the spacecraft rendezvous model are given in Figure 4 and Table 4, with the tool settings below.

Setting for Ariadne. The maximum step size used is 1.0, essentially meaning that we allow the step size to vary widely along evolution: this choice turned out to be preferable in terms of execution time. The maximum temporal order is 4 and the maximum spacial error enforced for each step equal is 10^{-3}. A splitting strategy for the initial set is used; the strategy compares 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.

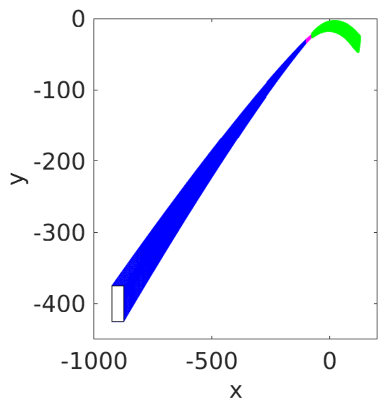
Setting for CORA. CORA was run with a time step size of 0.2 [min] for the modes *approaching* and *aborting*, and with a time step size of 0.05 [min] for mode *rendezvous attempt*. The intersections with the guard sets were calculated with the method of Girard, which was introduced in [23]. In order to find suitable orthogonal directions, the last zonotope that did not intersect the guard set is projected onto the hyperplane that represents the guard set. Then, Principal Component Analysis is applied to the generators of the projected zonotope.

Setting for DynIbex. The library does not support hybrid systems automatically but event detection can computed from the reachable tube computed in each mode. 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 2 (Heun's method). No splitting of the initial state has been performed. The computation of the state-space at a given event time interval $[t_{\min}, t_{\max}]$ is performed by the union of all the boxes of the reachable tube occurring at $[t_{\min}, t_{\max}]$.

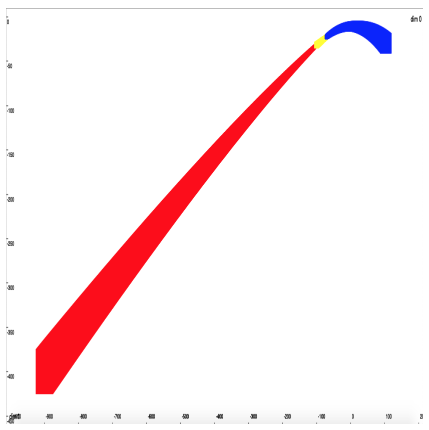
Setting for Flow*. We use the old version of Flow*. The model can be directly verified by the tool with the following setting for parameters: the TM order is fixed by 5, the stepsize is adaptively selected in the range from 0.001 to 0.5, the remainder estimation is the interval $[-10^{-3}, 10^{-3}]$ in all dimensions, and the cutoff threshold is 10^{-6}. Besides, we use the precision 100. We simply aggregate the intersections after each jump by a box instead of a parallelotope which is more time-costly to compute but more accurate, since it is already sufficient to prove the property.



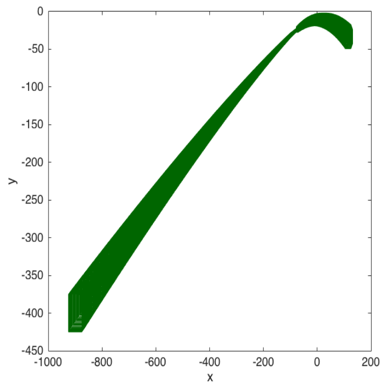
(a) Ariadne.



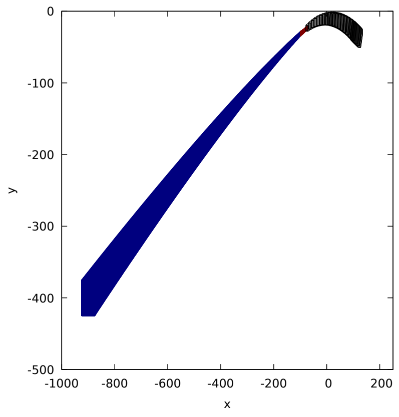
(b) CORA.



(c) DynIbex.



(d) Flow*.



(e) Isabelle/HOL.

Figure 4: Reachable set of the spacecraft position in the x-y-plane. CORA, Isabelle/HOL, and DynIbex use different colors to encode different modes of the hybrid system.

Setting for Isabelle/HOL. Isabelle/HOL does not support hybrid systems automatically. One can, however, compute the reachable sets for each mode seperately. The intersection with the guard set is computed with the method of Girard [23], simply using axis-aligned orthogonal directions, which results in box enclosures. We verify that the transition to mode *rendezvous attempt* occurs at $t \in [108.66, 111.71]$, $x = -100$, $y \in [-35.04, -28.43]$, $v_x \in [1.99365, 2.00644]$, $v_y \in [0.6489, 0.8130]$. Starting from this box, the transition to mode *aborting* occurs at $t = 120$, $x \in [-78.02, 71, 20]$, $y \in [-27.34, -20.23]$, $v_x \in [2.135, 2.341]$, $v_y \in [0.603, 0.825]$. From there we compute the reachable set until time $t = 200$, which satisfies $y < -1$ [m]. In those computations, maximum zonotope order is set to 50 and we use absolute and relative error tolerance of 2^{-10}.

4 Conclusion and Outlook

Three tools participated for the first time in the Nonlinear Dynamics category of ARCH-COMP: Ariadne, DynIbex, and JuliaReach. Unfortunately, three tools from last year did not continue to participate: CORA/SX, C2E2, and SymReach.

We always cared about repeatability of the reported results. This year, we enforced a uniform format for repeatability packages: All of the tools and benchmarks are made available as Docker [14] containers (on gitlab.com/goranf/ARCH-COMP). This improves and simplifies reproducibility of our results.

Compared to last year's competition [26], we made some careful adjustments to the existing benchmark problems to provide a more fine-grained overview of the capabilities of the different tools: We included a parameter for the stiffness of the van-der-Pol system, added another initial set to the Laub-Loomis problem, and report on the width of final enclosures.

We evaluate this inclusion of additional parameters as a success. One can see that μ has a significant influence on the difficulty of the van-der-Pol system for the participating tools, and we might expand more on this influence in future competitions. The additional initial set in the Laub-Loomis problem delineates limitations of the participating tools more precisely.

Triggered by the participation in this competition, individual tools made progress:

- Flow* entirely updated its module for continuous dynamics, which shows a much better performance while maintaining the same (or even better) precision.

- Isabelle/HOL still cannot automatically deal with hybrid systems, but its internals have been refactored significantly. This will simplify automatic reachability analysis for hybrid systems. This refactoring will also help to implement a stand-alone application that parses a standard input file format in the future.

- Last year, JuliaReach participated only in the AFF category. This competition fostered the *first* collaboration between the two halves of the JuliaReach team. The JuliaReach team evaluates this collaboration as very fruitful and is expecting to participate in next year's edition.

5 Acknowledgments

The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921. Luis Benet acknowledges the support from PAPIIT grant IG-100819. Alexandre Chapoutot benefited from the support of the "Chair Complex Systems Engineering – Ecole Polytechnique, THALES, DGA, FX, Dassault Aviation, DCNS Research, ENSTA ParisTech, Télécom ParisTech, and Fondation ParisTech" and he is grateful to Julien Alexandre dit Sandretto for his precious help in the implementation of some examples in DynIbex.

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.

A Specification of Used Machines

A.1 $\mathbf{M_{Ariadne}}$

Virtual machine on VirtualBox 6.0 on macOS 10.14.3 with a single core CPU and 8.0 GB of reserved memory. The operating system of the VM is Ubuntu 18.04.2 LTS. The physical CPU is given as below:

- Processor: Intel Core i7-6920HQ CPU @ 2.90GHz x 4
- Average CPU Mark on www.cpubenchmark.net: 9599 (full), 2019 (single thread)

Ariadne currently does not exploit multi-threading.

A.2 $\mathbf{M_{CORA}}$

- Processor: Intel Core i7-7820HQ CPU @ 2.90GHz x 4
- Memory: 32 GB
- Average CPU Mark on www.cpubenchmark.net: 9409 (full), 2070 (single thread)

A.3 $\mathbf{M_{Flow*}}$

Virtual machine on VMware Workstation 11 with a single core CPU and 4.0 GB memory. The operating systems is Ubuntu 16.04 LTS. The physical CPU is given as below.

- Processor: Intel Xeon E3-1245 V3 @ 3.4GHz x 4
- Average CPU Mark on www.cpubenchmark.net: 9545 (full), 2155 (single thread)

A.4 $\mathbf{M_{Isabelle}}$

- Processor: Intel Core i7-8750H CPU @ 2.20GHz x 6
- Memory: 16 GB 2666 MHz DDR4
- Average CPU Mark on www.cpubenchmark.net: 12,516 (full), 2368 (single thread)

A.5 $\mathbf{M_{JuliaReach}}$

- Processor: Intel Core i7-4770HQ CPU @ 2.20GHz x 4
- Memory: 16 GB
- Average CPU Mark on `www.cpubenchmark.net`: 8948 (full), 1893 (single thread)

A.6 $\mathbf{M_{DynIbex}}$

- Processor: Intel(R) Core(TM) i5-7Y54 CPU @ 1.20GHz x 2
- Memory: 8 GB
- Average CPU Mark on `www.cpubenchmark.net`: 3603 (full), 1379 (single thread)

References

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

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

[3] 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.

[4] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. PhD thesis, Technischen Universität München, 2010.

[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, 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.

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

[10] Luis Benet and David P. Sanders. JuliaDiff/TaylorSeries.jl, March 2019.

[11] Luis Benet and David P. Sanders. JuliaIntervals/TaylorModels.jl, March 2019.

[12] 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.

[13] 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.

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

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

[16] 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.

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

[18] 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.

[19] 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.

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

[21] 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.

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

[23] A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proc. of Hybrid Systems: Computation and Control*, LNCS 4981, pages 215–228. Springer, 2008.

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

[25] 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.

[26] Fabian Immler, Matthias Althoff, Xin Chen, Chuchu Fan, Goran Frehse, Niklas Kochdumper, Yangge Li, Sayan Mitra, Mahendra Singh Tomar, and Majid Zamani. Arch-comp18 category report: Continuous and hybrid systems with nonlinear dynamics. In Goran Frehse, editor, *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 54 of *EPiC Series in Computing*, pages 53–70. EasyChair, 2018.

[27] 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.

[28] 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.

[29] Jorge A. Pérez-Hernández and Luis Benet. PerezHz/TaylorIntegration.jl, February 2019.

[30] 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.

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