EPiC Series in Computing

Volume 96, 2023, Pages 61–88

Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)

ARCH-COMP23 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics

Luca Geretti[1], Julien Alexandre dit Sandretto[2], Matthias Althoff[3], Luis Benet[4], Pieter Collins[5], Marcelo Forets[6], Elena Ivanova[2], Yangge Li[9], Sayan Mitra[9], Stefan Mitsch[7], Christian Schilling[8], Mark Wetzlinger[3], and Daniel Zhuang[9]

[1] Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA
geretti@usc.edu
[2] ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France
julien.alexandre-dit-sandretto@ensta-paris.fr,elena.ivanova@ensta-paris.fr
[3] Technische Universität München, Munich, Germany
althoff@in.tum.de,m.wetzlinger@tum.de
[4] Instituto de Ciencias Físicas, Universidad Nacional Autónoma de México (UNAM), México
benet@icf.unam.mx
[5] Department of Data Science and Knowledge Engineering, Maastricht University, Maastricht, The Netherlands
pieter.collins@maastrichtuniversity.nl
[6] Universidad de la República, Montevideo, Uruguay
mforets@gmail.com
[7] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
smitsch@cs.cmu.edu
[8] Aalborg University, Aalborg, Denmark
christianms@cs.aau.dk
[9] University of Illinois Urbana-Champaign
li213@illinois.edu,mitras@illinois.edu,dzhuang6@illinois.edu

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 2023. This year, 6 tools participated: Ariadne, CORA, DynIbex, JuliaReach, KeYmaera X and Verse (in alphabetic order). 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 [17] containers at gitlab.com/goranf/ARCH-COMP.

In this report, we summarize the results of the seventh 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, JuliaReach, KeYmaera X and Verse participated in solving problems defined on three continuous and two hybrid benchmarks. The continuous benchmarks are the Traffic scenario, the Robertson chemical reaction system, the Coupled Van der Pol oscillator and the Laub-Loomis model of enzymatic activities. The hybrid benchmarks model a Lotka-Volterra predator-prey system with a Tangential Crossing, and a Space Rendezvous system.

The benchmarks were selected based on discussions between the tool authors, with a preference on keeping a significant set of the benchmarks from the previous year. It is apparent that they come from very different domains and aim at identifying issues specific to nonlinear dynamics, possibly with the addition of hybrid behavior.

2 Participating Tools

Ariadne. (Luca Geretti, Pieter Collins) *Ariadne* [22, 16] is a library based on *Computable Analysis* [44] 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, with a focus on compositionality [20]. It has been mainly applied to the verification of robotic tasks [28]. The library is written in modern C++ with an optional Python interface. The official site for Ariadne is `https://www.ariadne-cps.org`.

CORA. (Matthias Althoff, Mark Wetzlinger) 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. (Julien Alexandre dit Sandretto, Elena Ivanova) 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 [**?**]. 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 [23] which is available at `http://enstabretagnerobotics.github.io/VIBES/`. Computations are performed on a Lenovo laptop with i5 processor, and computation times gather all the process from compilation to figure producing.

JuliaReach. (Luis Benet, Marcelo Forets, Christian Schilling) JuliaReach [18] 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 the Taylor-model based solvers, (`TMJets20`, `TMJets21a`, and `TMJets21b`), which are implemented in TaylorModels.jl [15], integrates the packages TaylorSeries.jl [12, 13] and TaylorIntegration.jl [36], and the IntervalArithmetic.jl [14] package for interval methods. The algorithms applied in this report first 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 small deviations 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 the set library LazySets.jl [24].

KeYmaera X. (Stefan Mitsch) KeYmaera X [26] is a theorem prover for the hybrid systems logic differential dynamic logic (dL). It implements the uniform substitution calculus of dL [37]. A comparison of the internal reasoning principles in the KeYmaera family of provers with a discussion of their relative benefits and drawbacks is in [35], and model structuring and proof management on top of uniform substitution is discussed in [33]. KeYmaera X supports systems with nondeterministic discrete jumps, nonlinear differential equations, nondeterministic inputs, and allows defining functions implicitly through their characterizing differential equations [27]. It provides invariant construction and proving techniques for differential equations [41, 38], and stability verification techniques for switched systems [42]. Unlike numerical hybrid systems reachability analysis tools, KeYmaera X also supports unbounded initial sets and unbounded time analysis. Proofs in KeYmaera X can be conducted interactively [34], steered with tactics [25], or attempted fully automatic.

Verse. (Yangge Li, Sayan Mitra, Daniel Zhuang) Verse [32] is an open-source Python library for verifying multi-agent scenarios. It converts the scenario into a hybrid system and uses reachability analysis to verify the generated hybrid system. Verse can handle systems with black box dynamics, uncertainty in discrete transitions, and uncertainty in initial states. Discrete transitions in Verse are written using executable Python code, and the library parses the code to obtain guards and resets. Under the hood, Verse employs the simulation-driven reachability analysis algorithm from [**?**]. It uses a finite number of simulations of trajectories in a given mode, along with a discrepancy function that bounds the sensitivity of these trajectories, to over-approximate the reachable states. To compute the discrepancy function, Verse employs a probabilistic algorithm that learns the parameters of an exponential discrepancy function from simulation data by solving a linear program. Verse is publicly available on GitHub at https://github.com/AutoVerse-ai/Verse-library.

3 Benchmarks

For the 2023 edition of the competition we essentially kept the original benchmark suite. However, we modified the *van der Pol* system to make it more amenable to analysis with respect to last year.

3.1 Traffic scenario benchmark (TRAF22)

The avoidance of collisions in traffic scenarios is of utmost interest in the development of motion planners for autonomous driving. Recently [30], a workflow for the automated generation of verification tasks has been proposed based on an extraction of traffic scenario benchmarks from the CommonRoad framework [8].

3.1.1 Model

The nonlinear continuous-time dynamics are represented by a kinematic single-track model [30, Eq. (1)]:

$$\begin{cases} \dot{\delta} = u_1 + w_1 \\ \dot{\psi} = \dfrac{v}{l_{wb}} \tan \delta \\ \dot{v} = u_2 + w_2 \\ \dot{s}_x = v \cos \psi \\ \dot{s}_y = v \sin \psi, \end{cases}$$

where the state vector $x \in \mathbb{R}^5$ consists of the steering angle δ, the vehicle heading ψ, the vehicle velocity v, and the positions s_x, s_y of the vehicle along the x-axis and y-axis. The control inputs u_1, u_2 represent the steering angle and acceleration, respectively. Additionally, model uncertainties and disturbances affecting the vehicle are modeled by the disturbances w_1, w_2. In order to follow a reference trajectory $x_{ref} \in \mathbb{R}^5$, we apply a feedback controller of the form [30, Eq. (2)]

$$u_{fb}(\widehat{x}) = u_{ref} + K(\widehat{x} - x_{ref})$$

with the time-varying reference input $u_{ref} \in \mathbb{R}^2$, the time-varying feedback matrix $K \in \mathbb{R}^{2\times 5}$, and the measured state $\widehat{x} := x + z$ defined using the measurement error $z \in \mathbb{R}^5$. Thus, the ten-dimensional closed-loop system $f(x, u, w)$ is obtained by inserting the control law into the five-dimensional model:

$$\begin{cases} \dot{x} = f(x, u_{ref} + K(x + z - x_{ref}), w) \\ \dot{x}_{ref} = f(x_{ref}, u_{ref}, 0) \end{cases}$$

3.1.2 Analysis

The set for the measurement error $\mathcal{Z} \subset \mathbb{R}^5$, the input set $\mathcal{U} \subset \mathbb{R}^2$, and the set of disturbances $\mathcal{W} \subset \mathbb{R}^2$ are respectively bounded by

$$\mathcal{Z} = \begin{pmatrix} [-0.0004, 0.0004] \\ [-0.0004, 0.0004] \\ [-0.006, 0.006] \\ [-0.002, 0.002] \\ [-0.002, 0.002] \end{pmatrix} \qquad \mathcal{U} = \begin{pmatrix} [-0.7, 0.7] \\ [-11, 11] \end{pmatrix} \qquad \mathcal{W} = \begin{pmatrix} [-0.02, 0.02] \\ [-0.3, 0.3] \end{pmatrix}.$$

Table 1: Results of TRAF22 in terms of computation time and verification.

tool	**computation time in [s]**	**Verified?**
Ariadne	N/A	N/A
CORA	36	Yes
DynIbex	98[1]	No
JuliaReach	57	Yes
KeYmaera X	N/A	N/A
Verse	N/A	N/A

[1] Computed on local machine

The initial state is uncertain within the set $x_0 \oplus \mathcal{Z} \times x_0$. The inputs u_1, u_2 and the disturbances w_1, w_2 can change arbitrarily over time within their respective sets.

In this case, we analyze the scenario with the identifier *BEL_Putte-4_2_T-1*: The time horizon is determined by the length of the piecewise-constant control values, i.e., the reference trajectory x_{ref}, reference input u_{ref}, and feedback matrix K. All of these are provided by a .csv-file in a format as detailed in [30, Sec. 5].

The following two specifications have to be satisfied:

- **Input constraints**: The controller input $u_{fb} \in \mathbb{R}^2$ should be contained within the input set $\mathcal{U}$ at all times. The set of control inputs is computed according to [30, Eq. (5)].

- **Collision avoidance**: The car should not collide with static or dynamic obstacles as well as the road boundaries. Therefore, one requires to compute the car's occupancy set according to [30, Eq. (4)]. After rewriting the occupancy set as a .csv-file using the format in [30, Fig. 4], the collision check is performed fully automatically by calling a provided Python script as detailed in [30, Sec. 5].

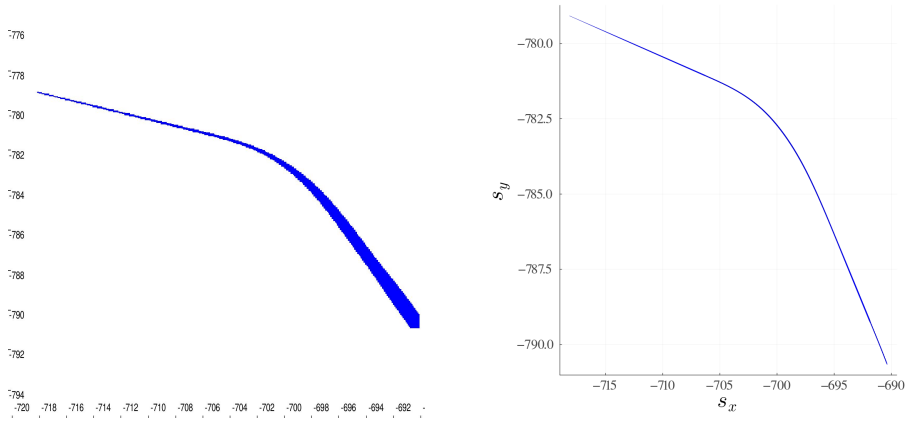
3.1.3 Evaluation

There are two metrics to evaluate the performance of each tool. First, we measure the computation time only comprising the time spent during the reachable set computation, exempt the time step in the pre- and post-processing steps. Second, we explicitly tabulate the results of the verification since a collision could occur at any time and therefore might not be captured in the figures below.

3.1.4 Results

The results from this benchmark are shown in Table 1. Since last year, DynIbex and JuliaReach that added their support; although, some of the tools still do not support the format required by the benchmark.

Settings for Ariadne. Ariadne is currently able to express disturbances within purely continuous dynamics, while the piecewise-constant input requires extension to the hybrid space. We plan on supporting hybrid systems for the next year.



(a) DynIbex (b) JuliaReach

Figure 1: Reachable set overapproximations for TRAF22.

Settings for CORA. We used the conservative linearization approach [10] with a time step size of $\Delta t = 0.005$, resulting in 20 steps per piecewise-constant input. Despite the relatively high system dimension, a zonotope order of 20 was sufficient for successful verification.

Settings for DynIbex. The Runge-Kutta method selected is Kutta at three order (called KUTTA3 in DynIbex). The absolute precision is 10^{-7}. The noise number for affine arithmetic is 200.

Settings for JuliaReach. We use $n_Q = 1$, $n_T = 5$ and an adaptive absolute tolerance 10^{-11}. JuliaReach does not support time-varying disturbances; the disturbances are instead modeled as uncertain but constant state variables $w(0) \in W, \dot{w} = 0$. The reported time consists of computing the reachable states and checking the input constraints.

Settings for KeYmaera X. KeYmaera X does not currently support reading streams of recorded control inputs and outputting occupancy sets as required in the collision avoidance simulation step of this benchmark. In future editions, we plan to formalize streams of control inputs fully symbolically to characterize the safety-relevant properties of such streams and conduct proofs for any control input stream satisfying these properties.

Settings for Verse. Verse currently cannot handle the inputs u_1, u_2 or the disturbances w_1, w_2 that can change arbitrarily overtime. We are planning to tackle this problem in future version of the tool.

3.2 Robertson chemical reaction benchmark (ROBE21)

3.2.1 Model

As proposed by Robertson [39], this chemical reaction system models the kinetics of an autocatalytic reaction.

$$\begin{cases} \dot{x} = -\alpha x + \beta yz \\ \dot{y} = \alpha x - \beta yz - \gamma y^2 \\ \dot{z} = \gamma y^2 \end{cases}$$

where x, y and z are the (positive) concentrations of the species, with the assumption that $x+y+z = 1$. Here α is a small constant, while β and γ take on large values. In this benchmark we fix $\alpha = 0.4$ and analyze the system under three different pairs of values for β and γ:

1. $\beta = 10^2$, $\gamma = 10^3$
2. $\beta = 10^3$, $\gamma = 10^5$
3. $\beta = 10^3$, $\gamma = 10^7$

The initial condition is always $x(0) = 1$, $y(0) = 0$ and $z(0) = 0$.

3.2.2 Analysis

We are interested in computing the reachable tube until $t = 40$, to see how the integration scheme holds under the stiff behavior. No verification objective is enforced.

3.2.3 Evaluation

For each of the three setups, the following three measures are required:

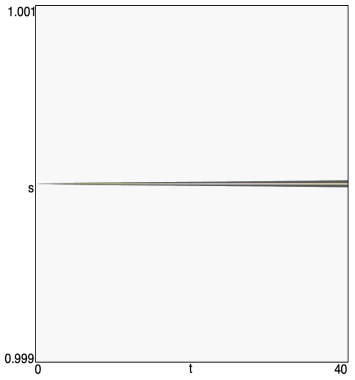
1. the execution time for evolution;
2. the number of integration steps taken;
3. the width of the sum of the concentrations $s = x + y + z$ at the final time.

Additionally, a figure with s (in the $[0.999, 1.001]$ range) w.r.t. time overlaid for the three setups should be shown.

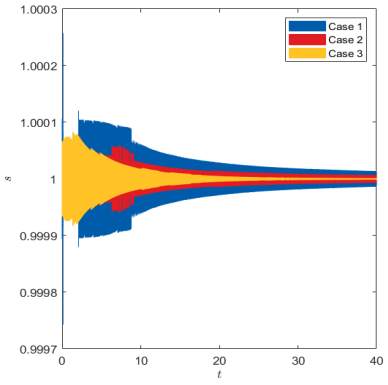
3.2.4 Results

All tools were able to get to completion. However, very different results were obtained. In the case of Ariadne and JuliaReach, the width started small and increased monotonically, while for DynIbex and CORA the width started decreasing from a given value. It is also interesting to analyze the number of integration steps taken, which turned out to be sensibly lower for JuliaReach and especially CORA. While JuliaReach obtained the best width for the stiffer cases, this came at the expense of a significantly higher computation time. Perhaps for the next year some verification constraints should be enforced, in order to provide a better baseline for comparison between the tools.

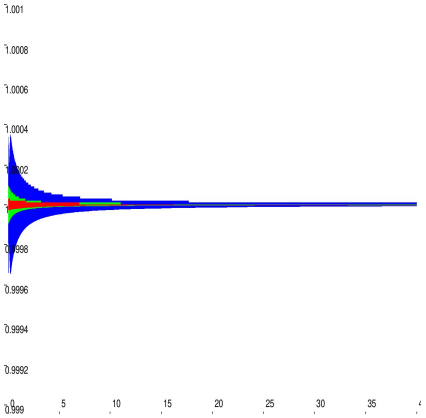
Settings for Ariadne. A GradedTaylorSeriesIntegrator is used, with a maximum error per integration step of 10^{-9}. A maximum step size of 0.004 is imposed in all three setups, though the actual value dynamically identified along evolution for (2) and (3) is sensibly lower.



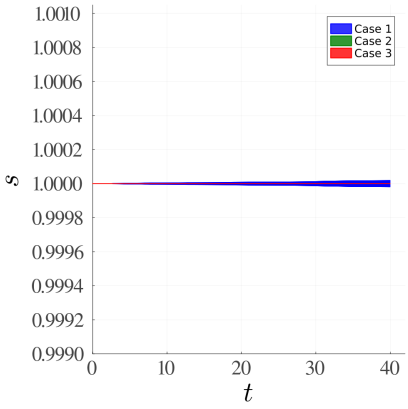
(a) Ariadne



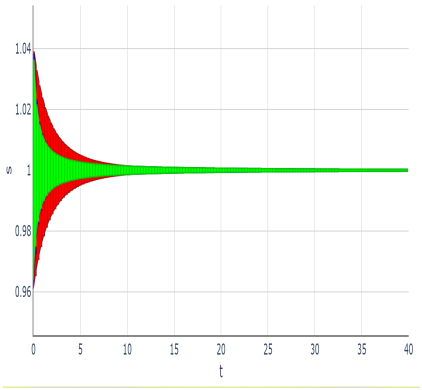
(b) CORA



(c) DynIbex



(d) JuliaReach



(e) Verse

Figure 2: Reachable set overapproximations of $s = x + y + z$ vs time for ROBE21 in the three setups.

Table 2: Results of ROBE21 in terms of computation time, number of steps and width of $s = x + y + z$.

	computation time in [s]		
tool	(1)	(2)	(3)
Ariadne	49	253	464
CORA	159	356	856
DynIbex	744[1]	4115[1]	6861[1]
JuliaReach	77	1110	4413
KeYmaera X[2]	0.3	0.3	0.3
Verse	7.8	8.6	8.3

[1] Computed on local machine

[2] Single symbolic proof solves all 3 examples

	number of steps		
tool	(1)	(2)	(3)
Ariadne	10000	49849	123675
CORA	17941	37437	74785
DynIbex	8694	84460	123248
JuliaReach	3494	30119	71367
KeYmaera X[1]	326	326	326
Verse	400	400	400

[1] Proof steps, symbolic proof solves all 3 examples

	width of $x + y + z$		
tool	(1)	(2)	(3)
Ariadne	1.1e-5	4.3e-5	1.1e-5
CORA	2.2e-5	9.8e-6	1.1e-6
DynIbex	4.2e-7	1.8e-5	1.3e-6
JuliaReach	3.8e-5	8.1e-8	1.2e-9
KeYmaera X[1]	0	0	0
Verse	1.3e-4	6.4e-4	6.4e-4

[1] Exact computation without overapproximation

Settings for CORA. In all cases, we used the approach from [45], which adaptively tunes all algorithm parameters during runtime.

Settings for DynIbex. The Runge-Kutta method selected is implicit Lobatto at fourth order (called LC3 in DynIbex) for the three setups. The absolute precision is respectively 10^{-14}, 10^{-14} and 10^{-14}. The other parameters are set by default.

Settings for JuliaReach. In all cases we use $n_Q = 1$, an initial adaptive absolute tolerance 10^{-10} and the `TMJets21a` algorithm, adapting only the n_T parameter as follows: (1) $n_T = 5$, (2) $n_T = 7$ and (3) $n_T = 10$. The maximum number of integration steps is also adjusted, reflecting the results presented in Table 2. For the results displayed in Fig. 2, we evaluate s directly on the Taylor models produced by the integration.

Settings for KeYmaera X. The KeYmaera X proof is fully parametric, without approximation, and shows stability of all possible population sums s for any (even negative) choice of a, b, and g, which includes the specific parametrizations (i) $b = 10^2, g = 10^3$, (ii) $b = 10^3, g = 10^5$, and (iii) $b = 10^3, g = 10^7$.

```
1 Problem
2   x+y+z=s
```

```
  ->
  [{ x' = -a*x + b*y*z,
     y' = a*x - b*y*z - g*y^2,
     z' = g*y^2
   }
  ](x+y+z=s)
End.

Tactic "Scripted proof" unfold; dIClose(1) End.
Tactic "Automated proof" autoClose End.
```

Settings for Verse Verse uses implicit Runge-Kutta method of the Radau IIA family of order 5 (Radau) to solve the problem. We are using the implementation of the algorithm from Python `scipy` package. The absolute tolerance used is 10^{-12}. The integration is evaluated every 0.1s.

3.3 Coupled van der Pol benchmark (CVDP23)

3.3.1 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 [11]. The system can be defined by the following ODE with 5 variables:

$$\begin{cases} \dot{x}_1 &= y_1 \\ \dot{y}_1 &= \mu(1-x_1^2)y_1 + b(x_2 - x_1) - x_1 \\ \dot{x}_2 &= y_2 \\ \dot{y}_2 &= \mu(1-x_2^2)y_2 - b(x_2 - x_1) - x_2 \\ \dot{b} &= 0 \end{cases} \tag{1}$$

with $\mu = 1$. The system has a stable limit cycle that becomes increasingly sharper for higher values of μ.

3.3.2 Analysis

We set the initial condition $x_{1,2}(0) \in [1.25, 1.55]$, $y_{1,2}(0) \in [2.35, 2.45]$ and $b \in [1\,, 3]$. The unsafe set is given by $y_{1,2} \geq 2.75$ in a time horizon of $[0, 7]$.

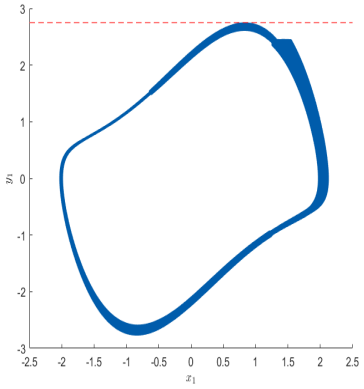
3.3.3 Evaluation

The computation time required to evolve the system and verify safety is provided. If the system can not be verified successfully, no value is given.

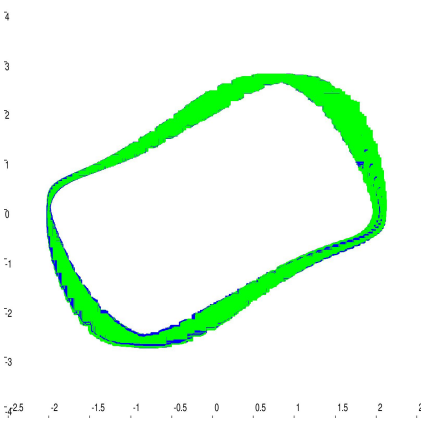
3.3.4 Results

The computation results of the tools are given in Table 3. While KeYmaera X was not able to participate in this specific benchmark, Ariadne and Cora encountered numerical problems that prevented completion in a reasonable time. Only JuliaReach was able to address the benchmark properly. DynIbex used a partial worst case analysis to obtain a result in a reasonable time.

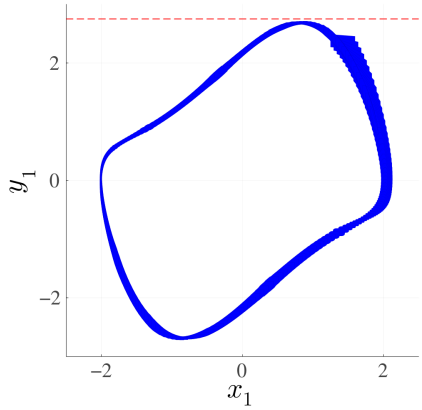
Settings for Ariadne. It was not possible to achieve completion is a reasonable time, due to the very high number of splittings theoretically required to guarantee numerical convergence.



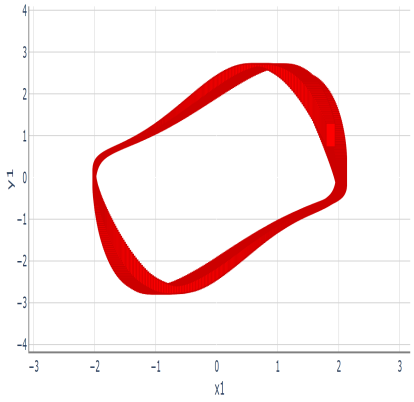
(a) CORA



(b) DynIbex



(c) JuliaReach



(d) Verse

Figure 3: Reachable set overapproximations for CVDP23.

Table 3: Results of CVDP23 in terms of computation time.

tool	**computation time in [s]**
Ariadne	N/A
CORA	744
DynIbex	1073[1]
JuliaReach	1.6
KeYmaera X	N/A
Verse	1.2

[1] Computed on local machine

Settings for CORA. Due to the strong nonlinearity induced by the parameter b, it was necessary to split the initial set into 13 smaller subsets. For each run, we used the polynomialization algorithm in [5] with a time step size of 0.005 and a zonotope order of 100. Additionally, we manually introduced two artificial guard sets orthogonal to the flow in order to shrink the reachable set. Otherwise, the abstraction error and thus the computed reachable set would explode in size.

Settings for DynIbex. Maximum zonotope order is set to 80, reachability analysis is carried out with an (absolute and relative) error tolerance of 10^{-6} using an explicit RK4 method of order 4. A formal B-series, based on recent developments [1], is computed with the help of Bseries Julia package. A partial worst case analysis is performed by considering initial value set at extremal value for one dimension w.r.t. others given in intervals. It leads to 5 initial conditions that must be verified. Moreover, a bissection is performed when safety cannot be verified (363 simulations are needed).

Settings for JuliaReach. We use $n_Q = 1$, $n_T = 8$, and an adaptive absolute tolerance 10^{-10}.

Settings for Verse We use 16 simulated trajectories to compute the discrepancy function. The simulation time step for this example is 0.01s.

Settings for KeYmaera X. The Coupled van der Pol benchmark was formalized for KeYmaera X but not yet proved. Below, we give the formal specification in KeYmaera X format:

```
Definitions Real m, b; End.
ProgramVariables Real x1, x2, y1, y2; End.
Problem
    1 <= b & b <= 3                                                    /* b in [1,3] */
  & m = 1
  & 1.25 <= x1 & x1 <= 1.55 & 1.25 <= x2 & x2 <= 1.55 /* x_{1,2}(0) in [1.25,1.55] */
  & 2.35 <= y1 & y1 <= 2.45 & 2.35 <= y2 & y2 <= 2.45 /* y_{1,2}(0) in [2.35,2.45] */
  & t = 0
  ->
  [{ x1' = y1,
     y1' = m*(1-x1^2)*y1 + b*(x2-x1) - x1,
     x2' = y2,
     y2' = m*(1-x2^2)*y2 - b*(x2-x1) - x2,
     t' = 1 & t <= 7                                          /* time horizon [0,7] */
   }
  ]!( y1>=2.75 & y2>=2.75)                                    /* not in unsafe set */
End.
```

In future editions, we plan to search and prove correct symbolic invariant conditions of the dynamics.

3.4 Laub-Loomis benchmark (LALO20)

3.4.1 Model

The Laub-Loomis model is presented in [31] 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, with the equilibrium point approximately [-0.87, 0.37, -0.56, -2.75, 0.22, -0.08, -0.27].

3.4.2 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 [43]. 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 $[0, 20]$.

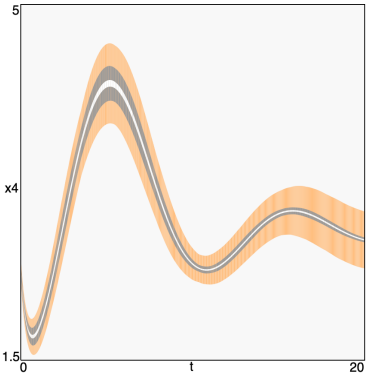
3.4.3 Evaluation

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.

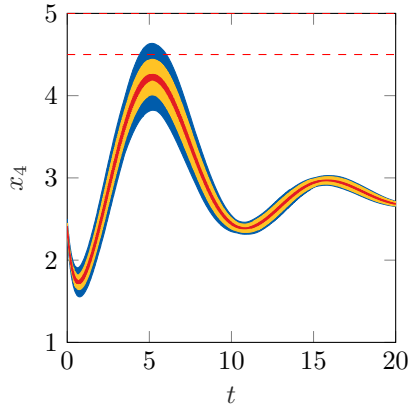
3.4.4 Results

The computation results of the tools are given in Table 4. Results are essentially the same as last year's.

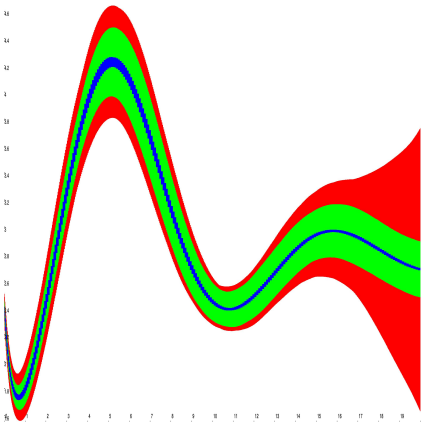
Settings for Ariadne. The maximum step size used is 0.2, with a TaylorPicardIntegrator with a maximum spacial error of 10^{-6} enforced for each step. Compared with last year, the same settings were used but a regression on quality was experienced.



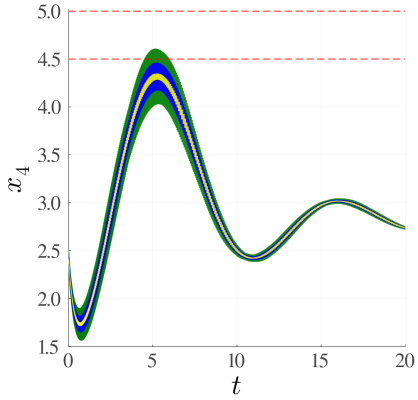
(a) Ariadne



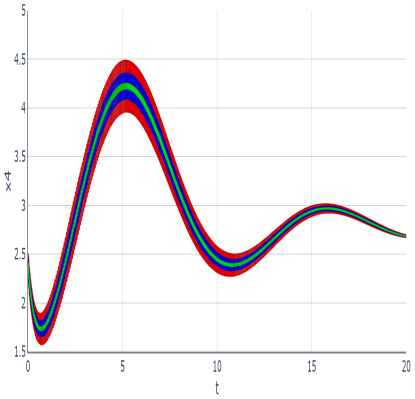
(b) CORA



(c) DynIbex



(d) JuliaReach



(e) Verse

Figure 4: 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 4: 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	9.1	39	113
CORA	9.7	14	284
DynIbex	10[1]	27[1]	1909[1]
JuliaReach	3.8	5.7	5.7
KeYmaera X	N/A	N/A	N/A
Verse	1.6	1.5	1.5

[1] Computed on local machine

	width of x_4 in final enclosure		
tool	$W = 0.01$	$W = 0.05$	$W = 0.1$
Ariadne	0.011	0.045	0.55
CORA	0.004	0.047	0.060
DynIbex	0.01	0.40	2.07
JuliaReach	0.0042	0.017	0.033
KeYmaera X	N/A	N/A	N/A
Verse	0.0024	0.0086	0.016

Settings for CORA. For the smaller initial sets $W = 0.01$ and $W = 0.05$, we applied an adaptively-tuned linearization algorithm [45], whereas the larger initial set $W = 0.1$ required a polynomialization algorithm, where we again used the adaptively-tuned version from [45].

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

Settings for KeYmaera X. The Laub-Loomis benchmark was formalized for KeYmaera X but not yet proved. Below, we give the formal specification in KeYmaera X format:

```
Definitions
  Real W = 0.1;
  Bool box(Real x, Real c, Real w) <-> c-w <= x & x <= c+w;
End.

ProgramVariables
  Real x1, x2, x3, x4, x5, x6, x7;  /* state space */
```

```
  Real t;                          /* time */
End.

Problem
    box(x1, 1.2, W)                /* initial sets */
  & box(x2, 1.05, W)
  & box(x3, 1.5, W)
  & box(x4, 2.4, W)
  & box(x5, 1, W)
  & box(x6, 0.1, W)
  & box(x7, 0.45, W)
  & t=0
  ->
  [{ x1' = 1.4*x3 - 0.9*x1,
     x2' = 2.5*x5 - 1.5*x2,
     x3' = 0.6*x7 - 0.8*x2*x3,
     x4' = 2 - 1.3*x3*x4,
     x5' = 0.7*x1 - x4*x5,
     x6' = 0.3*x1 - 3.1*x6,
     x7' = 1.8*x6 - 1.5*x2*x7,
     t'  = 1 & t<=20             /* time horizon [0,20] */
   }
  ]!( x4>=5)                       /* not in unsafe set */
End.
```

In future editions, we plan to search and prove correct symbolic invariant conditions of the dynamics.

Settings for Verse A simulation timestep of 0.02s was used for each case. We use 16 simulated trajectories for computing the discrepancy function.

3.5 Lotka–Volterra with tangential crossings benchmark (LOVO21)

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:

$$\sqrt{(x-1)^2 + (y-1)^2} = 0.161 \tag{3}$$

which is a circle of radius 0.161 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 close to tangent to the guard circle in the top half, while it crosses the circle on the bottom half. Hence, enlarging the width of the initial set would put the trajectory partially within the guard in the top half.

The corresponding hybrid automaton is used to model the system:

- Continuous variables: x, y;
- Locations: *outside* and *inside*;

- Dynamics: those from Eq. 2 for x, y in both locations;
- Guards:
$$\begin{cases} (x - Q_x)^2 + (y - Q_y)^2 \leq R^2 \text{ from } \textit{outside} \text{ to } \textit{inside} \\ (x - Q_x)^2 + (y - Q_y)^2 \geq R^2 \text{ from } \textit{inside} \text{ to } \textit{outside} \end{cases} \tag{4}$$
- 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 $\epsilon = 0.012$, 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. We must remark that, for reachability analysis purposes, it is important to carry the trace of discrete events along with the current evolution time.

The following three properties must be verified:

- At least one final set must have crossed two guards by entering and exiting the reference circle once;
- At least one final set must have crossed four guards by entering and exiting the reference circle twice;
- While a larger *even* number of crossings is allowed due to Zeno behavior during tangent crossing, no odd numbers are possible.

3.5.3 Evaluation

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

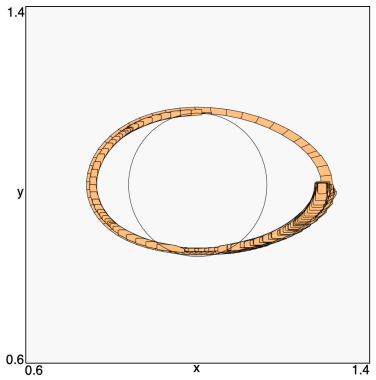
1. The execution time for computing the reachable set and checking the properties;
2. The area $x \times y$ of the box hull enclosing all the final sets.

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

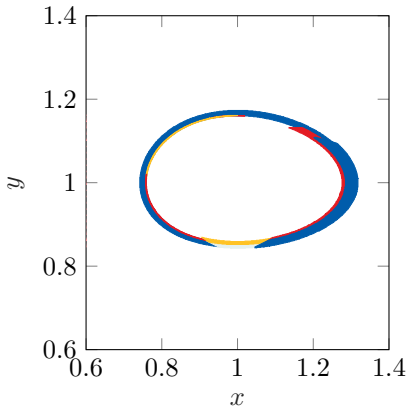
3.5.4 Results

All tools were able to handle the benchmark with results equivalent to last year. Table 5 gives the timing/quality results, while Fig. 5 shows the graphical output.

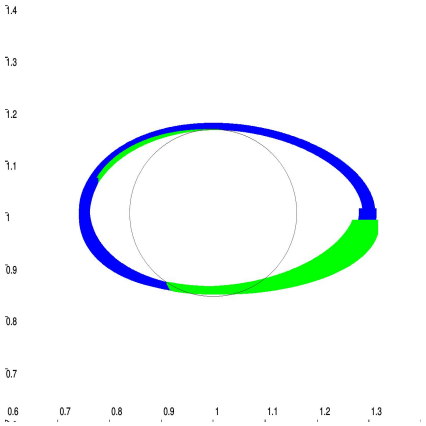
Settings for Ariadne. A GradedTaylorSeriesIntegrator is used with a maximum spacial error of $1e-7$. The maximum step size is 0.07. The maximum number of parameters for a set is 5 times the number of variables, instead of the default of 3 times. Please note that the settings are the same as last year, though apparently there is a regression both in quality (4 times) and especially in computational time (8 times).



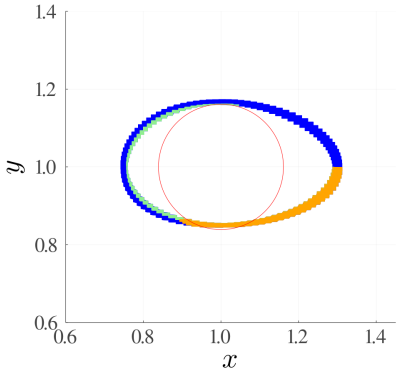
(a) Ariadne



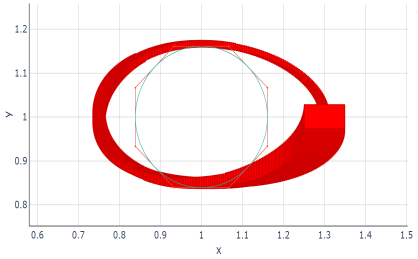
(b) CORA



(c) DynIbex



(d) JuliaReach



(e) Verse

Figure 5: Reachable set overapproximation for LOVO21, with $x, y \in [0.6, 1.4]$, where the circular guard is shown.

Table 5: Results of LOVO21 in terms of computation time and area.

tool	**computation time in [s]**	**area**
Ariadne	93	4.6e-4
CORA	26	4.8e-3
DynIbex	75[1]	5.9e-2
JuliaReach	3.6	1.4e-2
KeYmaera X	(1.5)[1,2]	0
Verse	12	5.3e-3

[1] Computed on local machine

[2] Duration of proving invariance (not checking crossing)

Settings for CORA. We use the approach in [29] to calculate the intersections with the non-linear guard set. For continuous reachability we apply the conservative linearization approach [10] 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^{-14} 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.

Settings for JuliaReach. We use $n_T = 7$, $n_Q = 1$, an adaptive absolute tolerance 10^{-10}, and split the initial set into 32 boxes. The crossings to the non-linear guard are handled by checking the reach sets that do not lie strictly outside the circle.

Settings for KeYmaera X. The KeYmaera X proof focuses on infinite-horizon population stability for any positive starting choice of populations $x > 0$ and $y > 0$, which includes the specific starting populations $x = 1.3 \pm \epsilon$ and $y = 1$. The population orbit is stable around $(\frac{\alpha}{\beta}, \frac{\gamma}{\delta})$ at population $e^{-\delta x - \beta y} x^\gamma y^\alpha$ for $\alpha = \beta = 3$ and $\delta = \gamma = 1$.

```
1  Definitions Real K(Real x, Real y) = exp(−d∗x−b∗y) ∗ x^g ∗ y^a; End.
2  Problem
3    a=3 & b=3 & d=1 & g=1 & x>0 & y>0 & K_0 = K(x,y)
4    −>
5    [{ x' = a∗x − b∗x∗y,
6       y' = d∗x∗y − g∗y
7     }
8    ] K(x,y) = K_0
9  End.
10
11 Tactic "Scripted proof"
12 useSolver("Mathematica");
13 unfold;
14 dIRule(1); <(
```

```
15    "dI Init": equalCommute(1); id,
16    "dI Step":
17      chaseAt(1);
18      QE using "(exp(−1∗x−3∗y)∗(−1∗(3∗x−3∗x∗y)−3∗(1∗x∗y−1∗y))∗x^1+exp(−1∗x−3∗y)∗(1∗x^(1−1)∗(3∗x
            ↪ −3∗x∗y)))∗y^3+exp(−1∗x−3∗y)∗x^1∗(3∗y^(3−1)∗(1∗x∗y−1∗y))=0"
19  )
20  End.
21  Tactic "Automated proof" autoClose End.
```

The formalization in the repeatability package also includes a symbolic characterization of the existence of crossing in and out of the nonlinear guard: this purely real arithmetic proof obligation is not yet tractable by the arithmetic backend verification procedures used in KeYmaera X. In future editions, we plan to additionally characterize the number of transitions symbolically.

Settings for Verse A simulation timestep of 0.01s was used. Since Verse cannot currently handle a non-linear guard, the circle was approximated using a octagon in which the guard circle is inscribed in the octagon. In addition, to avoid zeno behavior, we force the system to stay in each mode for 0.5s before a transition can happen.

3.6 Space rendezvous benchmark (SPRE22)

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 [21]. 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], $v_x \in [0, 5]$ [m/min] and $v_y \in [0, 5]$ [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 $\mathcal{L} = \{\binom{x}{y} \mid (x \geq -100) \wedge (y \geq x \tan(20°)) \wedge (-y \geq x \tan(20°))\}$.
- **Collision avoidance:** In mode *aborting*, the spacecraft has to avoid a collision with the target, which is modeled as a box $\mathcal{B}$ with 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 [21], 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 $y \in [-450, 0]$.

3.6.4 Results

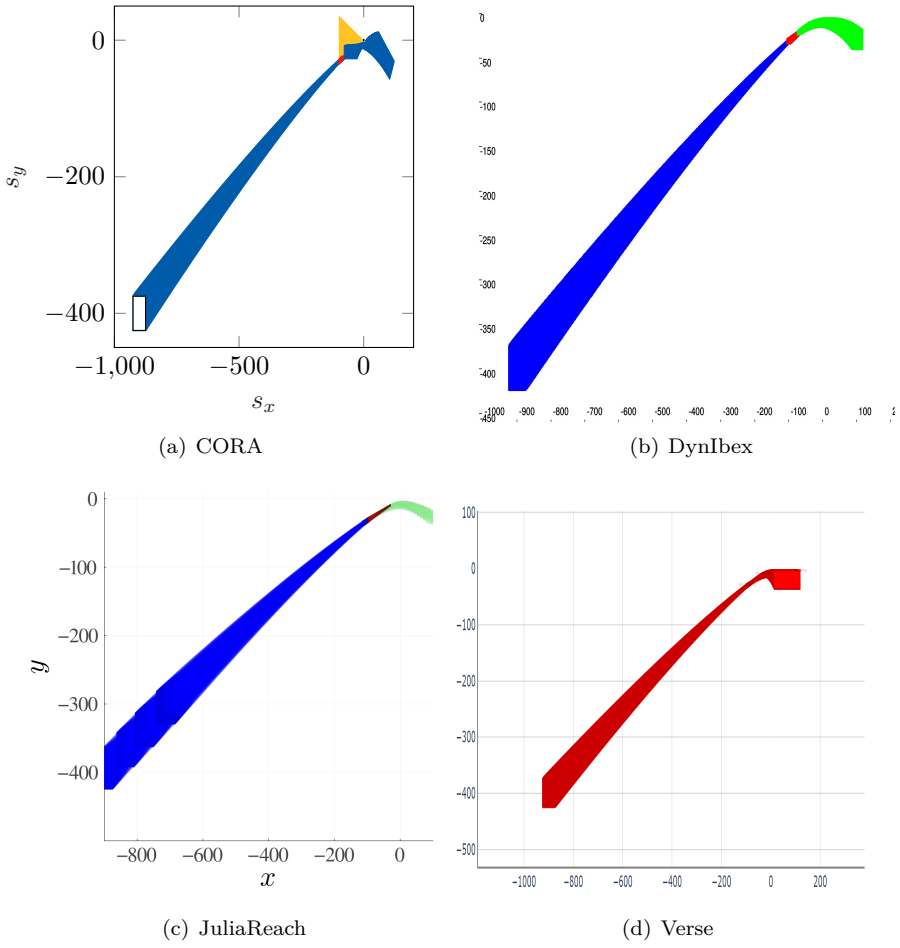
The results of the reachability computation for the spacecraft rendezvous model are given in Figure 6 and Table 6, with the tool settings below. The introduction of a permissive guard prevented completion for Ariadne: too many trajectories were generated and the absence of a recombination strategy proved 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. KeYmaera X formalized but not proved the problem yet.

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

Table 6: Results of SPRE21 in terms of computation time.

tool	**computation time in [s]**
Ariadne	–
CORA	27
DynIbex	61[1]
JuliaReach	21
KeYmaera X	N/A
Verse	277

[1] Computed on local machine



(a) CORA (b) DynIbex

(c) JuliaReach (d) Verse

Figure 6: Reachable set of the spacecraft position in the x-y-plane for SPRE21.

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 [40], and the intersection is then enclosed with a zonotope bundle [9]. 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 JuliaReach. The transition to the aborting mode is handled by clustering and Cartesian decomposition [19] with zonotope enclosures in low dimensions, (x, y) and (v_x, v_y). The continuous-time algorithms used in the modes approaching, rendez-vous attempt, and aborting are `TMJets20` (first two modes) and `TMJets21b` (third mode) with $n_T = 5, 4, 7$, $n_Q = 1, 1, 1$ and adaptive absolute tolerance $10^{-5}, 10^{-7}, 10^{-10}$, respectively.

Settings for Verse A timestep of .05s was used. We add a reset while switching to mode Rendezvous to reduce overapproximation caused by mode transition. Verse currently assume all transitions are urgent. Therefore, for this example we add a special variable `total_time` to perform the non-deterministic transition to aborting mode. It takes significant longer time to verify this benchmark due to large amount of computation introduced by this switch. We are planning to improve the algorithm to have better support for this type of transition in future version of Verse.

Settings for KeYmaera X. The example was formalized for KeYmaera X but not yet proved. The full model is included in the repeatability evaluation package.

4 Conclusion and Outlook

This year, the competition confirmed five out of six participants from 2021 and saw Verse as a new entry.

Speaking about benchmark evaluation, this year we chose to be particularly conservative, in order to allow all tools to tackle the existing suite. As such, only a modification to the van der Pol system was introduced, purely in order to make it more amenable to numerical analysis.

The CVDP23 benchmark still proved a bit too difficult for Ariadne and could not be addressed by KeYmaera X.

The TRAF22 benchmark was addressed by two more participants, which allowed to make some comparisons. We still expect every tool to support it for future years.

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

- The algorithms [10] and [5] for nonlinear continuous-time systems used in CORA have recently been extended by adaptive parameter tuning [45]. This year, the novel fully-automated algorithm was used for the LALO20 and ROBE21 benchmarks. While the computation time was longer than in previous editions (since algorithm parameters have to be tuned on-the-fly), there is no notable loss in accuracy. We hope to further develop the adaptive algorithm using the challenging benchmarks from this competition.

- JuliaReach was able to analyze the TRAF22 benchmark this year, which mainly required a parser for the input format.

Summarizing, 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. At the same time, we care about allowing all tools to solve all benchmarks and we will try to modify the most critical ones in order to achieve that. Consequently, for the next year we aim at refining the existing suite to advance in these directions, also possibly increasing the number of benchmarks.

5 Acknowledgments

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

Mark Wetzlinger acknowledges support from the research training group CONVEY funded by the German Research Foundation under grant GRK 2428.

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.

Luis Benet acknowledges support from PAPIIT-UNAM project IG-101122.

Parasara Sridhar Duggirala and Edward Kim acknowledge the support of the Air Force Office of Scientific Research under award number FA9550-19-1-0288 and FA9550-21-1-0121 and National Science Foundation (NSF) under grant numbers CNS 1935724 and CNS 2038960. Any opinions, findings, and conclusions 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 or National Science Foundation.

Stefan Mitsch was sponsored by the AFOSR under grant number FA9550-16-1-0288. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Christian Schilling acknowledges support from DIREC - Digital Research Centre Denmark and the Villum Investigator Grant S4OS.

References

[1] Julien Alexandre dit Sandretto. Set-based b-series. *Mathematics*, 10(17), 2022. URL: https://www.mdpi.com/2227-7390/10/17/3165, doi:10.3390/math10173165.

[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, M. Koschi, and S. Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017.

[9] 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.

[10] 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.

[11] 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.

[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. doi:10.21105/joss.01043.

[13] Luis Benet and David P. Sanders. JuliaDiff/TaylorSeries.jl. https://github.com/JuliaDiff/TaylorSeries.jl, April 2021. doi:10.5281/zenodo.2601941.

[14] Luis Benet and David P. Sanders. JuliaIntervals/IntervalArithmetic.jl. https://github.com/JuliaIntervals/IntervalArithmetic.jl, May 2021. doi:10.5281/zenodo.3336308.

[15] Luis Benet and David P. Sanders. JuliaIntervals/TaylorModels.jl. https://github.com/JuliaIntervals/TaylorModels.jl, June 2021. doi:10.5281/zenodo.2613102.

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

[18] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling. JuliaReach: a toolbox for set-based reachability. In *HSCC*, 2019. doi:10.1145/3302504.3311804.

[19] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Andreas Podelski, and Christian Schilling. Decomposing reach set computations with low-dimensional sets and high-dimensional matrices. *Inf. Comput.*, 2022. doi:10.1016/j.ic.2022.104937.

[20] Davide Bresolin, Pieter Collins, Luca Geretti, Roberto Segala, Tiziano Villa, and Sanja Živanović Gonzalez. A Computable and Compositional Semantics for Hybrid Automata. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, HSCC '20, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3365365.3382202.

[21] 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. URL: http://www.easychair.org/publications/paper/342723.

[22] 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.

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

[24] Marcelo Forets and Christian Schilling. LazySets.jl: Scalable symbolic-numeric set computations. *Proceedings of the JuliaCon Conferences*, 1(1):11, 2021. doi:10.21105/jcon.00097.

[25] Nathan Fulton, Stefan Mitsch, Rose Bohrer, and André Platzer. Bellerophon: Tactical theorem proving for hybrid systems. In *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, pages 207–224, 2017. doi:10.1007/978-3-319-66107-0_14.

[26] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. Keymaera X: an axiomatic tactical theorem prover for hybrid systems. In *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, pages 527–538, 2015. doi:10.1007/978-3-319-21401-6_36.

[27] James Gallicchio, Yong Kiam Tan, Stefan Mitsch, and André Platzer. Implicit definitions with differential equations for keymaera x (system description). In Jasmin Blanchette, Laura Kovacs, and Dirk Pattinson, editors, *IJCAR*, volume 13385 of *LNCS*. Springer, 2022. doi:10.1007/978-3-031-10769-6_42.

[28] L. Geretti, R. Muradore, D. Bresolin, P. Fiorini, and T. Villa. Parametric formal verification: the robotic paint spraying case study. In *Proceedings of the 20th IFAC World Congress*, pages 9248–9253, July 2017.

[29] 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.

[30] N. Kochdumper, P. Gassert, and M. Althoff. Verification of collision avoidance for CommonRoad traffic scenarios. In *Proc. of the 8th International Workshop on Applied Verification of Continuous and Hybrid Systems*, pages 184–194, 2021. doi:10.29007/1973.

[31] 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.

[32] Yangge Li, Haoqing Zhu, Katherine Braught, Keyi Shen, and Sayan Mitra. Verse: A python library for reasoning about multi-agent hybrid system scenarios, 2023. arXiv:2301.08714.

[33] Stefan Mitsch. Implicit and explicit proof management in keymaera X. In *Proceedings of the 6th Workshop on Formal Integrated Development Environment, F-IDE@NFM 2021, Held online, 24-25th May 2021*, pages 53–67, 2021. doi:10.4204/EPTCS.338.8.

[34] Stefan Mitsch and André Platzer. The keymaera X proof IDE - concepts on usability in hybrid systems theorem proving. In *Proceedings of the Third Workshop on Formal Integrated Development Environment, F-IDE@FM 2016, Limassol, Cyprus, November 8, 2016*, pages 67–81, 2016. doi:10.4204/EPTCS.240.5.

[35] Stefan Mitsch and André Platzer. A retrospective on developing hybrid system provers in the keymaera family - A tale of three provers. In *Deductive Software Verification: Future Perspectives - Reflections on the Occasion of 20 Years of KeY*, pages 21–64. 2020. doi:10.1007/978-3-030-64354-6_2.

[36] Jorge A. Pérez-Hernández and Luis Benet. PerezHz/TaylorIntegration.jl. https://github.com/PerezHz/TaylorIntegration.jl, May 2021. doi:10.5281/zenodo.2562352.

[37] André Platzer. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reason.*, 59(2):219–265, 2017. doi:10.1007/s10817-016-9385-1.

[38] André Platzer and Yong Kiam Tan. Differential equation invariance axiomatization. *J. ACM*, 67(1):6:1–6:66, 2020. doi:10.1145/3380825.

[39] H. H. Robertson. The solution of a set of reaction rate equations. In *"Numerical analysis: an introduction"*, page 178–182. Academic Press, 1966.

[40] 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.

[41] Andrew Sogokon, Stefan Mitsch, Yong Kiam Tan, Katherine Cordwell, and André Platzer. Pegasus: sound continuous invariant generation. *Formal Methods Syst. Des.*, 58(1-2):5–41, 2021. doi:10.1007/s10703-020-00355-z.

[42] Yong Kiam Tan, Stefan Mitsch, and André Platzer. Verifying switched system stability with logic. In *HSCC '22: 25th ACM International Conference on Hybrid Systems: Computation and Control, Milan, Italy, May 4 - 6, 2022*, pages 2:1–2:11, 2022. doi:10.1145/3501710.3519541.

[43] 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.

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

[45] M. Wetzlinger, A. Kulmburg, A. Le Penven, and M. Althoff. Adaptive reachability algorithms for nonlinear systems using abstraction error analysis. *Nonlinear Analysis: Hybrid Systems*, 46:101252, 2022. URL: https://www.sciencedirect.com/science/article/pii/S1751570X22000607, doi:10.1016/j.nahs.2022.101252.