EPiC Series in Computing
Volume 80, 2021, Pages 32–54
8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)

# ARCH-COMP21 Category Report: Continuous and Hybrid Systems with Nonlinear Dynamics

Luca Geretti[1], Julien Alexandre dit Sandretto[2], Matthias Althoff[3], Luis Benet[4], Alexandre Chapoutot[2], Pieter Collins[5], Parasara Sridhar Duggirala[6], Marcelo Forets[7], Edward Kim[6], Uziel Linares[8], David P. Sanders[8,9], Christian Schilling[10,11], and Mark Wetzlinger[3]

[1] Department of Computer Science, University of Verona, Verona, Italy
luca.geretti@univr.it
[2] ENSTA Paris, Institut Polytechnique de Paris, Palaiseau, France
julien.alexandre-dit-sandretto@ensta-paris.fr,alexandre.chapoutot@ensta-paris.fr
[3] Technische Universität München, Munich, Germany
althoff@in.tum.de,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] Department of Computer Science, University of North Carolina at Chapel Hill, US
psd@cs.unc.edu,ehkim@cs.unc.edu
[7] Universidad de la República, Montevideo, Uruguay
mforets@gmail.com
[8] Dept. de Física, Facultad de Ciencias, Universidad Nacional Autónoma de México (UNAM), Mexico City, México
uzielnmtz@ciencias.unam.mx,dpsanders@ciencias.unam.mx
[9] Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA, United States
[10] University of Konstanz, Konstanz, Germany
[11] Aalborg University, Aalborg, Denmark
christianms@cs.aau.dk

**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 2021. This year, 5 tools Ariadne, CORA, DynIbex, JuliaReach and Kaa (in alphabetic order) participated. These tools are applied to solve reachability analysis problems on five 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 [16] containers at gitlab.com/goranf/ARCH-COMP.

In this report, we summarize the results of the fifth 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, 5 tools, namely Ariadne, CORA, DynIbex, JuliaReach and Kaa participated in solving problems defined on three continuous and two hybrid benchmarks. This year the Kaa tool joined the competition, while the Flow$^*$ and Isabelle/HOL tools were not able to participate. The continuous benchmarks are 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* [19, 15] is a library based on *Computable Analysis* [35] that uses a rigorous numerical approach to all its algebraic, geometric and logical operations. In particular, it performs numerical rounding control of all external and internal operations, in order to enforce conservative interpretation of input specification and guarantee formal correctness of the computed output. It focuses on nonlinear systems, both continuous and hybrid, supporting differential and algebraic relations. It is written in 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.** (Alexandre Chapoutot, Julien Alexandre dit Sandretto) A library merging interval constraint satisfaction problem algorithms and guaranteed numerical integration methods based on Runge-Kutta numerical schemes implemented with affine arithmetic. This library is able to solve ordinary differential equations [2] and algebraic differential equations of index 1 [3], combined with numerical constraints on state variables and reachable tubes. It produces **sound results** taking into account round-off errors in floating-point computations and truncation errors generated by numerical integration methods [29]. 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 [24] 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, Uziel Linares, David P. Sanders, Christian Schilling) JuliaReach [17] 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` implemented in TaylorModels.jl [14] integrates the TaylorSeries.jl [11, 12] package, TaylorIntegration.jl [30] and the IntervalArithmetic.jl [13] package for interval methods. The latter is a self-contained implementation of interval arithmetic in pure Julia, which is competitive, in terms of performance, with C++ interval packages. 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 our core set-computations library LazySets.jl [32].

**Kaa.** (Parasara Sridhar Duggirala, Edward Kim) Kaa [26] is a Python redesign of Sapo [21], a tool written to compute and plot the reachable sets of discrete polynomial non-linear dynamical systems using parallelotope bundles. Reachable set computation of nonlinear systems using template polyhedra and Bernstein polynomials was first proposed in [20]. The representation of parallelotope bundles for reachability was proposed in [22] and the effectiveness of using bundles for reachability was demonstrated in [21, 23]. We extend these techniques to effectively utilize dynamic templates strategies, i.e schemes created to automatically generate template directions and parallelotopes. We use two techniques to generate such template directions. The first involves computing *local linear approximations* of the dynamics, and the second involves performing *Principle Component Analysis (PCA)*. Both techniques are performed using sample trajectory data. The sample trajectories are found by calculating *support points* over the parallelotope bundle and propagating them to the next step using the provided dynamics. At each step, we add at least one parallelotope defined by either PCA or Linear Approximation template directions. Each parallelotope added to the bundle has an associated *lifespan*, indicating the number of time steps the parallelotope and its template directions exist in the bundle before being removed. We employ a Python wrapper over NASA's Kodiak[1] as the optimization library responsible for computing the upper and lower offsets for all utilized template directions at each step. The original Sapo program could only handle polynomial dynamics through Bernstein polynomials. However, Kodiak allows us to circumvent this restriction and extend our techniques to general non-linear dynamics. Sympy is used for symbolic manipulation and substitution while Numpy is used for general linear-algebraic computations. Detailed explanations of the underlying techniques used in Kaa can be found in a recent paper including the co-authors [25].

## 3 Benchmarks

For the 2021 edition of the competition we introduced one new continuous benchmark: the *Robertson* system. This is a continuous system aimed at identifying the stability of integration schemes under a stiff behavior; it replaces the *Production-Destruction* system, which had the same objective. In addition, we modified the *Lotka-Volterra system with tangential crossing* and the *Space Rendezvous* system to make them slightly more difficult compared to last year. The *Laub-Loomis* system and the *Coupled Van der Pol* system were not modified, in order to identify any improvements to the tools from 2020.

### 3.1 Robertson chemical reaction benchmark (ROBE21)

#### 3.1.1 Model

As proposed by Robertson [31], this chemical reaction system models the kinetics of an auto-catalytic reaction.

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

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

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 $z(0) = 0$.

#### 3.1.2 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.

#### 3.1.3 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.

Table 1: 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 | 30 | 168 | 294 |
| CORA | 22 | 58 | 347 |
| DynIbex | 304 | 3362 | 5094 |
| JuliaReach | 63 | 1029 | 4392 |
| Kaa | – | – | – |

| | **number of steps** | | | | **width of $x + y + z$** | | |
|---|---|---|---|---|---|---|---|
| **tool** | (1) | (2) | (3) | **tool** | (1) | (2) | (3) |
| Ariadne | 10000 | 49851 | 123677 | Ariadne | 1.0e-5 | 4.0e-5 | 8.0e-6 |
| CORA | 3400 | 9500 | 48000 | CORA | 3.4e-4 | 1.3e-4 | 7.1e-6 |
| DynIbex | 8694 | 84460 | 123248 | DynIbex | 7.9e-4 | 1.4e-3 | 7.6e-4 |
| JuliaReach | 3494 | 30147 | 71367 | JuliaReach | 3.8e-5 | 8.1e-8 | 1.2e-9 |
| Kaa | – | – | – | Kaa | – | – | – |

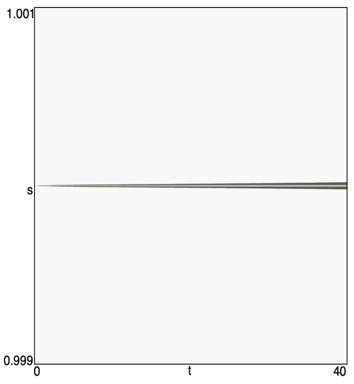
#### 3.1.4 Results

Except for Kaa, 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, to possibly increase further in the case of DynIbex. 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.

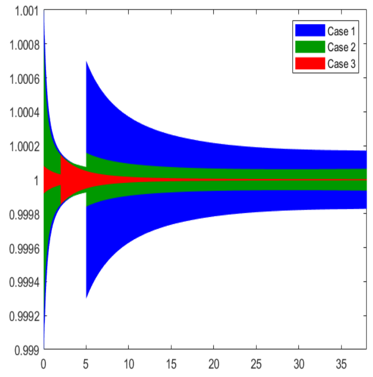
**Settings for Ariadne.** A GradedTaylorSeriesIntegrator is used, with a maximum error per integration step of $10^{-9}$. A maximum step size of 0.004 is imposed in all three setups, though the actual value dynamically identified along evolution for (2) and (3) is sensibly lower.

**Settings for CORA.** In all cases, we used the conservative linearization approach [9] with maximum zonotope orders of 30, 30, and 200, respectively. In order to accelerate the computation, we increased the time step size during the computation, at the loss of some amount of tightness of the reachable sets. The used time step sizes are (1) $t \in [0, 5] : 0.0025, t \in [5, 40] : 0.025$, (2) $t \in [0, 5] : 0.002, t \in [5, 40] : 0.005$, and (3) $t \in [0, 2] : 0.0002, t \in [2, 40] : 0.001$.

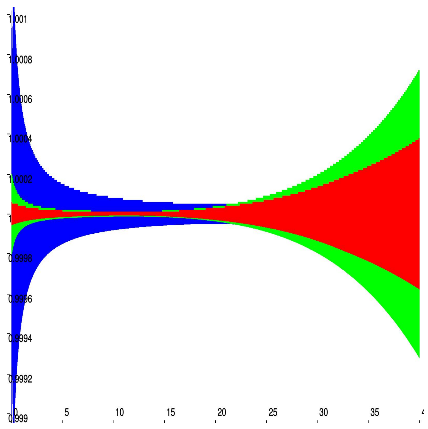
**Settings for DynIbex.** The Runge-Kutta method selected is implicit Lobatto at fourth order (called LC3 in DynIbex) for the three setups. The absolute precision is respectively $10^{-11}$, $10^{-12}$ and $10^{-12}$. The other parameters are set by default.



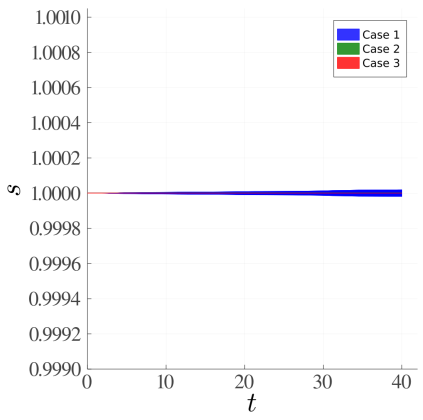
(a) Ariadne



(b) CORA



(c) DynIbex



(d) JuliaReach

Figure 1: Reachable set overapproximations of $s = x + y + z$ vs time for ROBE21 in the three setups.

**Settings for JuliaReach.** In all cases we used $n_Q = 1$, 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 was also adjusted, reflecting the results presented in Table 1. For the results displayed in Fig. 1, we used the much tighter approximation obtained by computing $s$ directly from the Taylor models produced by the integration, and then evaluating them in each time interval.

**Settings for Kaa.** For all cases, we've attempted to deploy both dynamic strategies involving PCA and local linear approximation templates and static strategies involving templates defined by random diagonal directions. However, in all cases, Kaa was unable to complete execution of the benchmark for the total time period $t \in [0, 40]$. From closer inspection, the dynamics causes explosive growth within the beginning of the computation; this explosion generally occurs between $t \in [0, 2]$. Our techniques, both static and dynamic, seem to be unable to control this dramatic behavior, which causes Kodiak to eventually crash.

## 3.2 Coupled van der Pol benchmark (CVDP20)

### 3.2.1 Model

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

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

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

### 3.2.2 Analysis

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

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

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

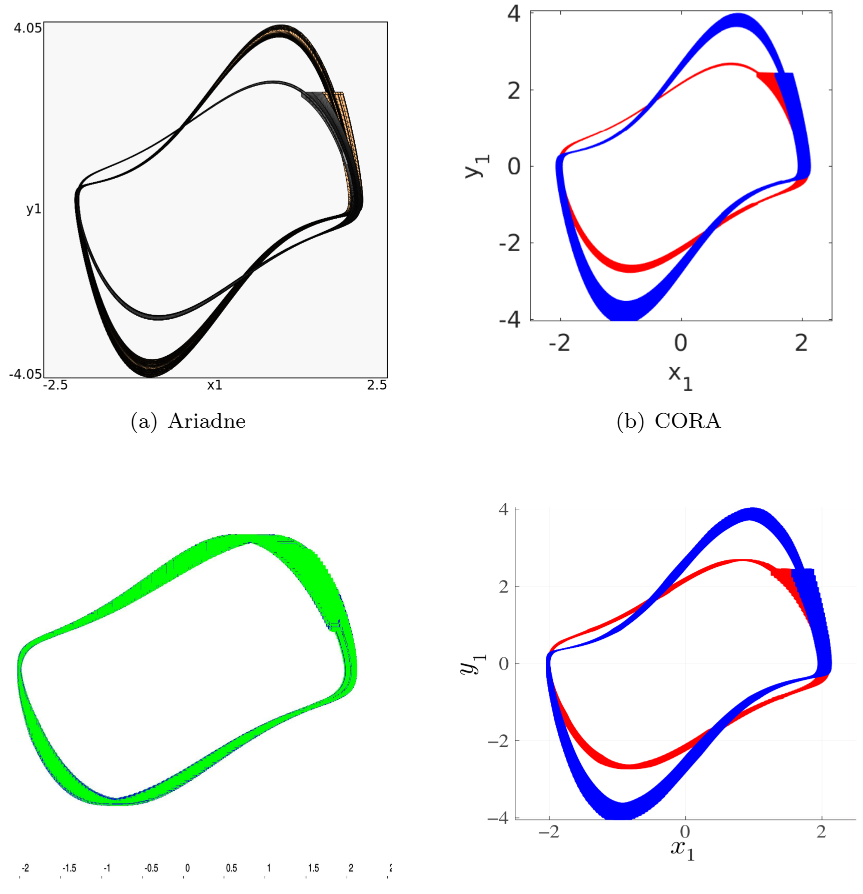
### 3.2.3 Evaluation

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

### 3.2.4 Results

The computation results of the tools are given in Table 2. Results are comparable to those from last year. The benchmark proved to be too hard for Kaa, however.

**Settings for Ariadne.** For $\mu = 1$, we use a TaylorPicardIntegrator with a maximum step size of 0.005. The maximum spacial error enforced for each step is $10^{-5}$. The initial set is split once on $x_1$ and $x_2$, yielding 4 initial subsets. For $\mu = 2$ instead, the initial set is split into 256 subsets, using a maximum spacial error of $5 \times 10^{-6}$ and a maximum step size of 0.02. It must be noted that the memory consumption for $\mu = 2$ is significant, in some cases making the operating system kill the process on a 16 GB machine during the final plotting. To avoid that, we simply did not store intermediate evolve sets, which are normally returned along with the final sets and the reached sets.



(a) Ariadne (b) CORA (c) DynIbex (d) JuliaReach

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

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

| | **computation time in [s]** | |
|---|---|---|
| **tool** | $\mu = 1$ | $\mu = 2$ |
| Ariadne | 11 | 1413 |
| CORA | 7.7 | 109 |
| DynIbex | 510 | – |
| JuliaReach | 1.5 | 104 |
| Kaa | – | – |

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

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

**Settings for JuliaReach.** For both settings we used $n_Q = 1$ and $n_T = 8$, and an adaptive absolute tolerance. For $\mu = 2$ we split the set of initial states along the $x_1$ and $x_2$ directions into 49 boxes in total.

**Settings for Kaa.** For both cases, we computed the reachable set using a dynamic strategy using PCA templates whose lifespans are set to 10 time steps. We have attempted to employ other strategies without much success. Templates generated through local linear approximations yielded numerical instabilities arising from computing the inverse of matrices with incredibly high condition numbers. We have also tried static parallelotopes defined by random diagonal directions. Here, static parallelotopes are defined at the beginning and its directions do not change during the course of the computation. However, the quality of the reachable set varies wildly depending on the diagonal directions sampled, and the outputted over-approximation seems to be extremely conservative. Even with a large amount of random static parallelotopes ($> 30$), we could not produce well-formed reachable sets. Thus, we believe that our current method of choosing effective templates is ill-suited for producing meaningful over-approximations for this benchmark at this time. The step sizes for both cases were set to $\Delta = 0.1$.

## 3.3 Laub-Loomis benchmark (LALO20)

### 3.3.1 Model

The Laub-Loomis model is presented in [28] for studying a class of enzymatic activities. The dynamics can be defined by the following ODE with 7 variables.

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

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

### 3.3.2 Analysis

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

The initial sets are defined according to the ones used in [34]. 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 $i$th dimension is defined by the interval $[x_i(0) - W, x_i(0) + W]$. The width $W$ of the initial set is vital to the difficulty of the reachability analysis job. The larger the initial set the harder the reachability analysis.

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

### 3.3.3 Evaluation

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

### 3.3.4 Results

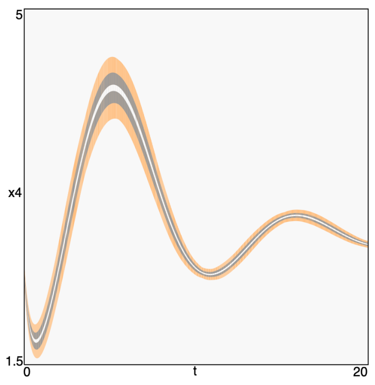
The computation results of the tools are given in Table 3. Results are comparable to those from the previous year, except for the newcomer Kaa for which convergence proved to be an issue. The tool settings are given as below.

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

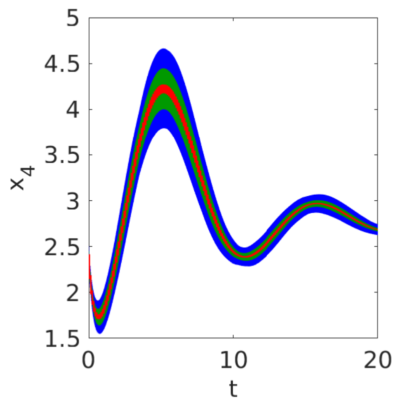
| | **computation time in [s]** | | |
|---|---|---|---|
| **tool** | $W = 0.01$ | $W = 0.05$ | $W = 0.1$ |
| Ariadne | 5.7 | 11 | 31 |
| CORA | 1.9 | 8.4 | 38 |
| DynIbex | 10 | 27 | 1851 |
| JuliaReach | 4.4 | 6.4 | 6.4 |
| Kaa | 238 | 253 | 257 |

| | **width of $x_4$ in final enclosure** | | |
|---|---|---|---|
| **tool** | $W = 0.01$ | $W = 0.05$ | $W = 0.1$ |
| Ariadne | 0.01 | 0.031 | 0.071 |
| CORA | 0.005 | 0.035 | 0.116 |
| DynIbex | 0.01 | 0.40 | 2.07 |
| JuliaReach | 0.0042 | 0.017 | 0.033 |
| Kaa | 22 | 23 | 49 |

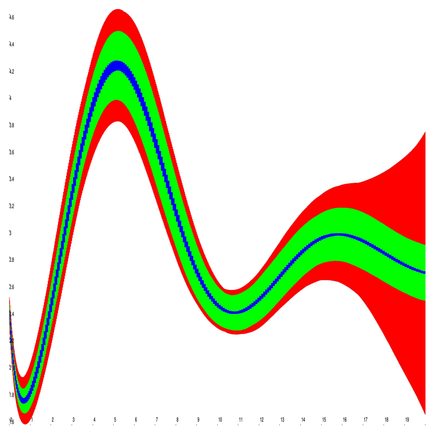
**Settings for Ariadne.** The maximum step size used is 0.2, with a TaylorPicardIntegrator with a maximum spacial error of $10^{-6}$ enforced for each step. Compared with last year, no splitting strategy for the initial set is necessary due to improvements in the integrator.



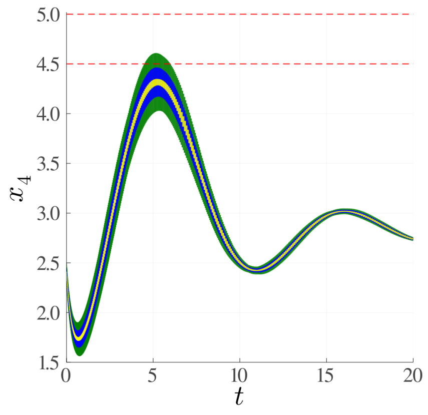
(a) Ariadne



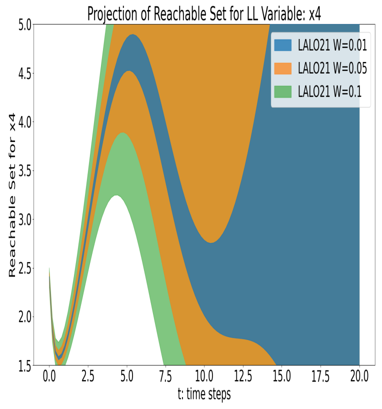
(b) CORA



(c) DynIbex



(d) JuliaReach



(e) Kaa

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

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

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

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

**Settings for Kaa.** For each case, we employed a dynamic strategy of PCA templates with their lifespan set to 10 steps. Once again, we faced difficulties with using local linear approximation. The approximate linear transformation calculated by our strategies give highly singular matrices, which in turn cause numerical instabilities. For this benchmark, it seems PCA templates are ineffective in controlling the error as time progresses. Due to this wrapping error, we capped the upper and lower offsets for each template direction to lie in between the intervals $[-10, 10]$ in order to ensure completion of the computation. We used a step size of $\Delta = 0.2$ for all cases.

## 3.4 Lotka–Volterra with tangential crossings benchmark (LOVO21)

### 3.4.1 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.

### 3.4.2 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.

### 3.4.3 Evaluation

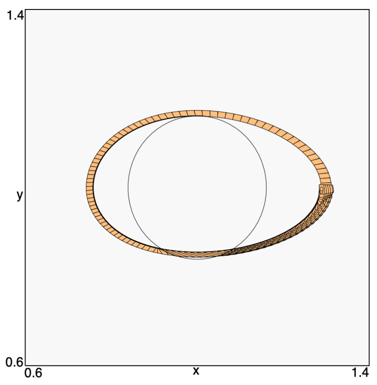
In terms of metrics, it is required to supply the following:

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.

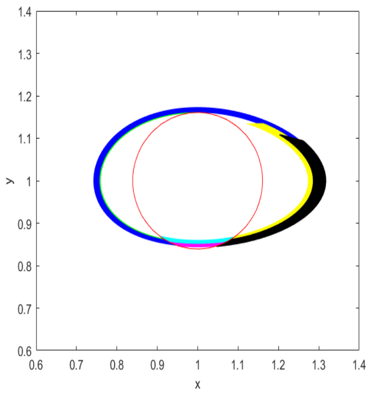
In addition, a figure showing the reachable set along with the circular guard shall be provided. The axes are $[0.6, 1.4] \times [0.6, 1.4]$.

### 3.4.4 Results

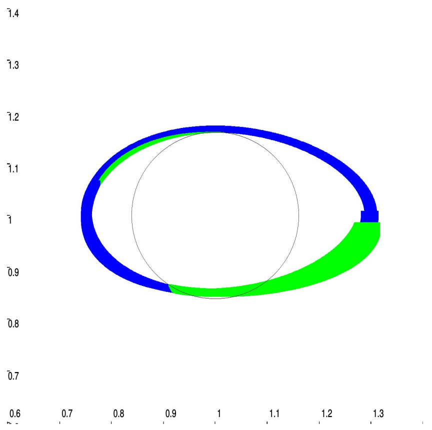
All tools were able to handle the benchmark, although with sensibly different quality of the final set. Table 4 gives the timing/quality results, while Fig. 4 shows the graphical output. A future improvement could be to introduce some restrictions on the final set in order to identify a baseline for comparison.



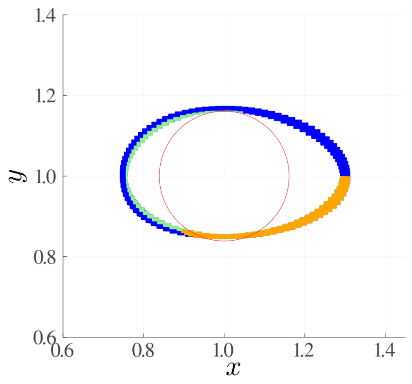
(a) Ariadne



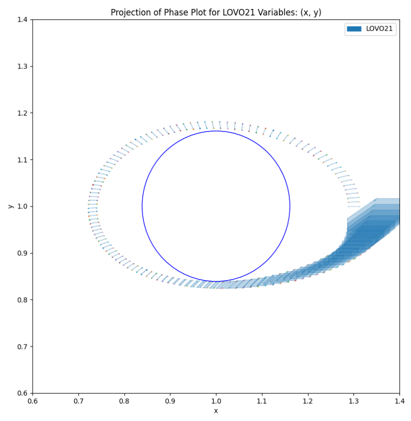
(b) CORA



(c) DynIbex



(d) JuliaReach



(e) Kaa

Figure 4: Reachable set overapproximation for LOVO21, with $x, y \in [0.6, 1.4]$, where the circular guard is shown.

Table 4: Results of LOVO21 in terms of computation time and area.

| **tool** | **computation time in [s]** | **area** |
|---|---|---|
| Ariadne | 8.0 | 1.2e-4 |
| CORA | 23 | 5.9e-3 |
| DynIbex | 75 | 8.8e-2 |
| JuliaReach | 3.4 | 1.4e-2 |
| Kaa | 180 | 4.7e-1 |

**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.

**Settings for CORA.** We use the approach in [27] to calculate the intersections with the non-linear guard set. For continuous reachability we apply the conservative linearization approach [9] with time step size of 0.005 and a zonotope order of 20 for all modes.

**Settings for DynIbex.** The library DynIbex does not support hybrid systems natively. However, based on constraint programming, event detection can be implemented and hybrid systems can be simulated. Reachability analysis is carried out with an error tolerance of $10^{-14}$ 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 used $n_T = 7$ and $n_Q = 1$ and split the initial set into 32 boxes. The crossings to the non-linear guard were handled by checking the flowpipes that do not lie strictly outside the circle.

**Settings for Kaa.** Although Kaa does not support hybrid dynamics, we can plot the reachable set utilizing a time step of $\Delta = 0.03$ for the dynamics given above. We employ a dynamic template strategy using only local linear approximation templates. Each added template has a lifespan of 20 steps.

## 3.5 Space rendezvous benchmark (SPRE21)

### 3.5.1 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 [18]. 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:

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

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 $\left(\begin{smallmatrix} u_x \\ u_y \end{smallmatrix}\right) = K_1\underline{x}$ for mode *approaching*, and $\left(\begin{smallmatrix} u_x \\ u_y \end{smallmatrix}\right) = K_2\underline{x}$ 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 $\left(\begin{smallmatrix} u_x \\ u_y \end{smallmatrix}\right) = \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$.

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

**Remark on velocity constraint** In the original benchmark [18], 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].

### 3.5.3 Evaluation

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

Table 5: Results of SPRE21 in terms of computation time.

| **tool** | **computation time in** [s] |
|---|---|
| Ariadne | – |
| CORA | 26 |
| DynIbex | 144 |
| JuliaReach | 24 |
| Kaa | – |

### 3.5.4 Results

The results of the reachability computation for the spacecraft rendezvous model are given in Figure 5 and Table 5, with the tool settings below. The introduction of a permissive guard prevented completion for Ariadne: too many trajectories were generated and the absence of a recombination strategy proved an issue. Therefore this benchmark requires proper support of crossings in the presence of large sets, even if the crossing region is very simple from a geometrical viewpoint. The hybrid nature of the problem was an obstacle for Kaa.

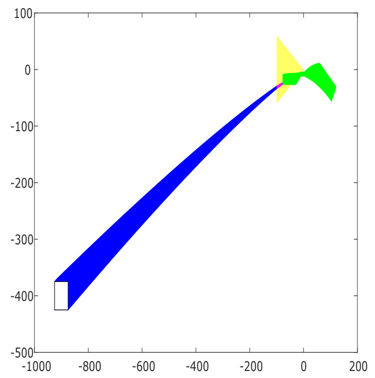
**Settings for Ariadne.** Ariadne was not able to complete evolution, due to the extremely large number of trajectories produced from the nondeterministic guard: this is caused by the lack of a recombination strategy. The maximum step size used was 1.0, essentially meaning that we allowed the step size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

**Settings for CORA.** CORA was run with a time step size of 0.2 [min] for the modes *approaching* and *aborting*, and with a time step size of 0.05 [min] for mode *rendezvous attempt*. The intersections with the guard sets are calculated with constrained zonotopes [33], and the intersection is then enclosed with a zonotope bundle [8]. In order to find suitable orthogonal directions for the enclosure *principal component analysis* is applied.

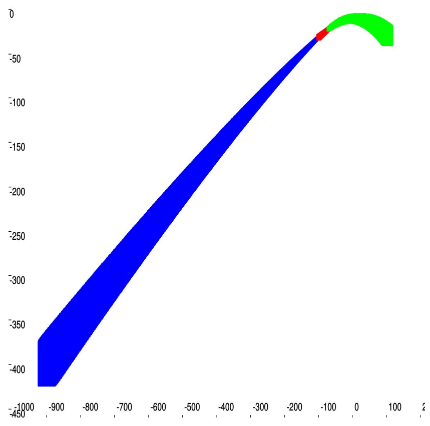
**Settings for DynIbex.** The library DynIbex does not support hybrid systems natively. However, based on constraint programming, event detection can be implemented and hybrid systems can be simulated. Maximum zonotope order is set to 10, reachability analysis is carried out with an error tolerance of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

**Settings for JuliaReach.** The transition to the aborting mode is handled by clustering and Cartesian decomposition 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 were `TMJets20` (first two modes), `TMJets21b` (third mode) with $n_T = 5, 4, 7$, $n_Q = 1, 1, 1$ and absolute tolerance $10^{-5}, 10^{-7}, 10^{-10}$, respectively.

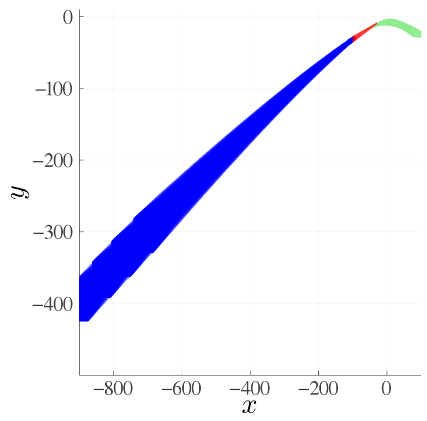
**Settings for Kaa.** Kaa does not support hybrid dynamics in its current state.



(a) CORA



(b) DynIbex



(c) JuliaReach

Figure 5: Reachable set of the spacecraft position in the $x$-$y$-plane for SPRE21.

## 4 Conclusion and Outlook

This year, the competition confirmed four participants from 2020, with two leaving and a new one joining. As a consequence, we did not introduce too many changes in order to accommodate the newcomer: we added only one new benchmark and modified two, along with the removal of one benchmark that was not deemed challenging anymore.

The new benchmark was the *Roberston* system (ROBE21), replacing the Production-Destruction (PRDE20) system, with a focus on stiffness. This was an interesting variation which returned sensibly different results. Given the different approaches, we might introduce some constraints in the evolution for the next year, in order to better compare the performances.

The Lotka-Volterra with Tangent Crossing (LOVO21) was modified in order to focus on crossings, practically increasing the number of trajectories involved. Results from the participants are quite different. Hence, for the next year, the addition of verification constraints might be introduced to simplify the comparison between tools.

The extension of the Space Rendezvous benchmark to a larger initial set introduced an increase in the computation time, but did not cause many issues to those tools that could handle it. Still, Ariadne and Kaa have their own peculiar limitations for which this benchmark should still be proposed next year. Also, it is the only benchmark that is natively hybrid (whereas the Lotka-Volterra benchmark is enriched with transitions for the pure purpose of evaluating tangent crossings).

Finally, the Laub-Loomis (LALO20) and Coupled Van der Pol (CVDP20) benchmarks were kept the same as the previous three years. Except for Kaa, tools are consistently capable of returning results for them with a satisfactory quality. We expect to keep these benchmarks to accommodate any future improvements in Kaa, while trying to squeeze anything out of the remaining tools.

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

- Ariadne introduced parallelization of its continuous and hybrid evolution, which significantly helped in those cases where the initial set required splitting, or where transitions in hybrid evolution generated multiple trajectories. Miscellaneous minor improvements to its integrators allowed to avoid splitting strategy in some cases and improving convergence for stiff systems. Compared to last year, different integrators have been used across the benchmarks to exploit their specific advantages.

- CORA enabled the handling of tangential crossings without infinite switching between locations following the LOVO20 benchmark. Furthermore, an adaptive parameter tuning approach for nonlinear systems was designed as a consequence of the severe changes in the degree of nonlinearity seen in last year's PRDE20 benchmark, which is replicated this year by ROBE21.

- DynIbex is used in the same release than last year. However, compilation with high level of optimization (O3) is now supported and allows slightly better time execution.

- JuliaReach introduced several advances in the core packages `ReachabilityAnalysis.jl`, `LazySets.jl` and `TaylorModels.jl`. Notable developments were a thorough revision of the Taylor-model based algorithms, a generalization of the initial set representation for those methods, and improvements in clustering and overapproximation of Taylor models with convex sets such as zonotopes.

- As Kaa's first year of participating in the competition, the benchmarks elucidated many drawbacks of parallelotope-based methods. The salient challenge seems to lie with the calculation of template directions based on local linear approximation. Calculating the best-fit linear transformation with Kaa's methods results in a highly singular matrix for most benchmarks, leading to numerical instability. Furthermore, the addition of PCA templates seems to have imparted a weaker positive effect than we had originally hoped. We plan to use these benchmarks as a set of models to search for more sophisticated template-generating techniques. In particular, developing methods of decreasing the condition number of the approximate linear transformations through testing with the benchmarks will be the next natural step in improving Kaa.

Summarizing, the new benchmark and their variations were interesting by way of continuing to explore critical aspects of continuous/hybrid evolution, with the objective of pushing all tools forward. We believe that a benchmark suite with representative problems is of the utmost importance, in order to stimulate meaningful progress of all the participating tools. Consequently, for the next year we aim at refining the existing suite to advance in this direction, also possibly increasing the number of benchmarks.

## 5 Acknowledgments

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

Luis Benet acknowledges support from PAPIIT grant IG-100819.

David Sanders acknowledges support from PAPIIT grant IN-117117.

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.

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.

## References

[1] Kodiak, a C++ library for rigorous branch and bound computation. https://github.com/nasa/Kodiak, Accessed: July 2020.

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

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

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

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

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

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

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

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

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

[11] 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.

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

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

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] Thao Dang and David Salinas. Image computation for polynomial dynamical systems using the bernstein expansion. In *International Conference on Computer Aided Verification*, pages 219–232. Springer, 2009.

[21] Tommaso Dreossi. Sapo: Reachability computation and parameter synthesis of polynomial dynamical systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 29–34, 2017.

[22] Tommaso Dreossi, Thao Dang, and Carla Piazza. Parallelotope bundles for polynomial reachability. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 297–306, 2016.

[23] Tommaso Dreossi, Thao Dang, and Carla Piazza. Reachability computation for polynomial dynamical systems. *Formal Methods in System Design*, 50(1):1–38, 2017.

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

[25] Edward Kim, Stanley Bak, and Parasara Sridhar Duggirala. Automatic dynamic parallelotope bundles for reachability analysis of nonlinear systems, 2021. arXiv:2105.11796.

[26] Edward Kim and Parasara Sridhar Duggirala. Kaa: A python implementation of reachable set computation using bernstein polynomials. *EPiC Series in Computing*, 74:184–196, 2020.

[27] 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.

[28] 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.

[29] 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.

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

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

[32] Christian Schilling and Marcelo Forets. JuliaReach/LazySets.jl: v1.45.1. `https://github.com/JuliaReach/LazySets.jl`, June 2021. Accessed: 2021-05-31. `doi:10.5281/zenodo.4896008`.

[33] 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.

[34] 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.

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