EPiC Series in Computing
Volume 96, 2023, Pages 34–60
Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)

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

Matthias Althoff[1], Marcelo Forets[2], Yangge Li[4], Sayan Mitra[4], Christian Schilling[3], Mark Wetzlinger[1], and Daniel Zhuang[4]

[1] Technical University of Munich, Department of Computer Engineering, Munich, Germany
althoff@in.tum.de, m.wetzlinger@tum.de
[2] Universidad de la República, Montevideo, Uruguay
mforets@gmail.com
[3] Aalborg University, Aalborg, Denmark
christianms@cs.aau.dk
[4] University of Illinois Urbana-Champaign
li213@illinois.edu,mitras@illinois.edu,dzhuang6@illinois.edu

Abstract

We present the results of the ARCH[1] 2023 friendly competition for formal verification of continuous and hybrid systems with linear continuous dynamics. In its seventh edition, three tools participated to solve nine different benchmark problems in the category for linear continuous dynamics (in alphabetical order): CORA, JuliaReach, and Verse. This report is a snapshot of the current landscape of tools and the types of benchmarks they are particularly suited for. Due to the diversity of problems, we are not ranking tools, yet the presented results provide one of the most complete assessments of tools for the safety verification of continuous and hybrid systems with linear continuous dynamics up to this date.

1 Introduction

Disclaimer The presented report of the ARCH friendly competition for *continuous and hybrid systems with linear continuous dynamics* aims at providing a landscape of the current capabilities of verification tools. We would like to stress that each tool has unique strengths—not all of the specificities can be highlighted within a single report. To reach a consensus in what benchmarks are used, some compromises had to be made so that some tools may benefit more from the presented choice than others.

[1]Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH), cps-vo.org/group/ARCH

We consider the verification of hybrid systems (i.e., mixed discrete/continuous systems) with linear continuous dynamics

$$\dot{x}(t) = Ax(t) + Bu(t),$$

where $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$, $B \in \mathbb{R}^{n\times m}$, and $u \in \mathbb{R}^m$. For all results reported by each participant, we have run an independent repeatability evaluation. To establish further trustworthiness of the results, the code with which the results have been obtained is publicly available at gitlab.com/goranf/ARCH-COMP. The selection of the benchmarks has been conducted within the forum of the ARCH website (cps-vo.org/group/ARCH), which is visible for registered users and registration is open to anybody. All tools presented in this report use some form of reachability analysis. This, however, is not a constraint set by the organizers of the friendly competition. We hope to encourage further tool developers to showcase their results in future editions. All tools are run on the same machine.

2 Participating Tools

The tools participating in the category *Continuous and Hybrid Systems with Linear Continuous Dynamics* are subsequently introduced in alphabetical order.

CORA (Matthias Althoff, Mark Wetzlinger) The tool *COntinuous Reachability Analyzer* (CORA) [2, 5, 6, 4, 24] 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. CORA is available at cora.in.tum.de.

JuliaReach (Marcelo Forets, Christian Schilling) *JuliaReach* is a software suite for reachability computations of dynamical systems [16], available at `http://juliareach.com/`. It is written in Julia, a modern high-level language for scientific computing. The core library is ReachabilityAnalysis.jl and for the set computations we use our LazySets.jl library [22]. JuliaReach can analyze systems in either continuous-time or discrete-time semantics. For some of the models we use our custom parser for SpaceEx model files, and otherwise we create the models in Julia. In this competition, we use the following algorithms: `BFFPSV18` (based on the support function on low-dimensional subspaces [15]), `GLGM06` (based on zonotopes [26]), `ASB07` (based on zonotopes for parametric systems [10]), and `LGG09` (based on support functions [29]). These algorithms can be combined with different *approximation models* [23], such as forward and correction hull, adapted from [15] and [1], respectively. For hybrid systems with time-triggered transitions, we use the algorithm from [21].

Verse Verse. (Yangge Li, Sayan Mitra, Daniel Zhuang) Verse [30] 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 both linear and nonlinear 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 [20]. 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 Verification of Benchmarks

For the 2023 edition, we have decided to keep all benchmarks from our 2022 friendly competition [11].

Special Features We briefly list the special features of each benchmark:

- *Heat 3D* benchmark from [13]: This is a purely continuous benchmark resulting from a spatial discretization of a heat partial differential equation in three dimensions. The system can be scaled from a $5 \times 5 \times 5$ mesh (125 dimensions) to a $100 \times 100 \times 100$ mesh (one million dimensions), each variation being roughly an order of magnitude apart.

- *Clamped beam* benchmark from [31, Sec. 4.2]: This purely continuous benchmark models a spatially discretized beam clamped on one end and pulled on the other end yielding interesting oscillations. The system dimension ranges from 200 to 2000 depending on the number of nodes used for the discretization. A challenge of this benchmark is that it has very little damping.

- *Space station* benchmark from [35]: This is a purely continuous benchmark of medium size with 270 state variables and three inputs.

- *Spacecraft rendezvous* benchmark from [17]: This benchmark has hybrid dynamics and is a linearization of a benchmark in the other ARCH-COMP category *Continuous and Hybrid Systems with Nonlinear Dynamics*. Consequently, the reader can observe the difference in computation time and verification results between the linearized version and the original dynamics.

- *Powertrain* benchmark from [7, Sec. 6]: This is a hybrid system for which one can select the number of continuous state variables and the size of the initial set. Up to 51 continuous state variables are considered.

- *Platooning* benchmark from [14]: A rather small number of continuous state variables is considered, but one can arbitrarily switch between two discrete states: a normal operation mode and a communication-failure mode.

- *Gearbox* benchmark from [18]: This benchmark has the smallest number of continuous state variables, but the reachable set does not converge to a steady state and the reachable set for one point in time might intersect multiple guards at once.

- *Brake* benchmark from [34]: This hybrid benchmark has a time-triggered discrete transition that has to be taken 1,001 times.

Types of Inputs Generally, we distinguish between three types of inputs:

1. Fixed inputs, where $u(t)$ is precisely known. In some cases, $u(t) = \texttt{const}$ as in the gearbox benchmark.
2. Uncertain but constant inputs, where $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ is uncertain within a set $\mathcal{U}$, but each uncertain input is constant over time: $u(t) = \texttt{const}$.
3. Uncertain, time-varying inputs $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ where $u(t) \neq \texttt{const}$. Those systems do not converge to a steady state solution and consider uncertain inputs of all frequencies. For tools that cannot consider arbitrarily varying inputs, we have stated that changes in inputs are only considered at fixed points in time.

Different Paths to Success When tools use a fundamentally different way of solving a benchmark problem, we add further explanations.

Computation Time The computation times specified in this report include the computation time of the reachable set and the time needed for the verification of the specifications.

3.1 Heat3D

3.1.1 Model

Using a mesh, the Heat3D benchmark is a spatially-discretized partial differential equation (PDE) for heat transfer in three dimensions, resulting in ordinary differential equations (ODEs), where each variable represents a mesh point. Depending on the granularity of the discretization, one can adjust the number of variables. This system has no switching or inputs and serves to evaluate the scalability with respect to the number of system dimensions. It is an academic example, although modifications, such as external inputs or more complicated specifications, can be added in the future. This benchmark was used in [13] and is based on a 2D version originally described and evaluated in [28, 27].

All of the sides of the considered heated block are insulated, except the $x = 1$ edge, causing heat exchange with the ambient environment with a heat exchange constant of 0.5. A heated initial region is present in the region, where $x \in [0.0, 0.4]$, $y \in [0.0, 0.2]$, and $z \in [0.0, 0.1]$. The entire initial heated region is the same temperature, which is nondeterministic and chosen in the range 0.9 to 1.1, with the remaining material initially at temperature 0.0. The system dynamics is given by the heat equation PDE $u_t = \alpha^2(u_{xx} + u_{yy} + u_{zz})$, where $\alpha = 0.01$ is the diffusivity of the material.

A linear model of the system is obtained using the semi-finite difference method, discretizing the block with an $m \times m \times m$ grid. This results in an m^3-dimensional linear system describing the evolution of the temperature at each mesh point.

Due to the initially heated region, we expect the temperature at the center of the block to first increase, and then decrease due to the heat loss along the $x = 1$ edge. Further, the discretization error increases for smaller m motivating the higher-dimensional versions of the benchmark. We suggest a time bound of $T = 40$ and a step size of 0.02 (2000 steps).

3.1.2 Specifications

The goal is to find the maximum temperature reached at the center of a $1 \times 1 \times 1$ block, where one edge of the block is initially heated. This can be converted to a safety verification problem by checking that $T_{\max}$ is reachable but $T_{\max} + \delta$ is not, for some small δ like 10^{-4}.

There are five suggested sizes, roughly each one an order of magnitude apart in terms of the number of dimensions. The higher-dimensional versions usually prevent explicitly representing the dynamics as a dense matrix in memory. Storing a million by million dense matrix requires a trillion numbers, which at 8 bytes per double-precision number would require eight terabytes of storage.

HEAT01 $5 \times 5 \times 5$ (125 dimensions). Note: the initial set is modified to be heated when $z \in [0.0, 0.2]$ (single mesh point), since that is the best we can do with this granularity. $T_{\max}$: 0.10369 at time 9.44.

HEAT02 $10 \times 10 \times 10$ (1000 dimensions). $T_{\max}$: 0.02966 at time 25.5.

HEAT03 $20 \times 20 \times 20$ (8000 dimensions). $T_{\max}$: 0.01716 at time 22.62.

HEAT04 $50 \times 50 \times 50$ (125,000 dimensions). $T_{\max}$: 0.01161 at time 18.88.

HEAT05 $100 \times 100 \times 100$ (1,000,000 dimensions). $T_{\max}$: 0.01005 at time 17.5.

3.1.3 Results

Plots for the $5 \times 5 \times 5$ case are shown in Figure 1. Results are shown in Table 1.

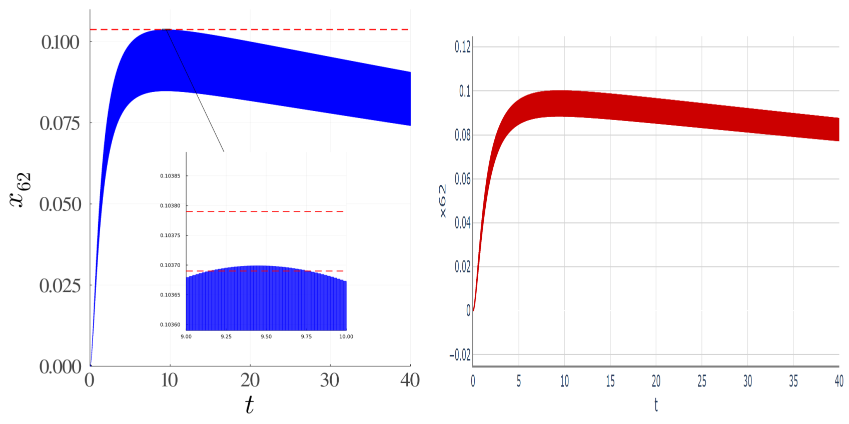
Note CORA CORA applies the fully-automated verification algorithm in [36] for the benchmark instances HEAT01 and HEAT02. For the higher-dimensional benchmark instance HEAT03 we compute the reachable set using the Krylov-subspace-based reachability algorithm in [3] using a time step size of 0.005.

Note JuliaReach We use the `LGG09` algorithm. For HEAT03 and HEAT04 we lazily compute the matrix exponential using the Lanczos algorithm [32] with Krylov subspace dimension of 94 and 211, respectively.

Note Verse A simulation time step of .02s was used with 35 simulated trajectories for HEAT01 and HEAT02. Verse is not optimized to handle large quantities of variables and is currently not able to solve HEAT03 - 05 in a reasonable amount of time or memory. We are currently exploring new algorithms to properly handle this type of benchmark.

Table 1: Computation Times for the Heat3D Benchmark in [s].

tool	**HEAT01**	**HEAT02**	**HEAT03**	**HEAT04**	**HEAT05**	**language**
CORA	1.04	2.98	419.8	–	–	MATLAB
JuliaReach	0.11	20.8	–	–	–	Julia
Verse	58.3	1024.9	–	–	–	Python
	discrete-time tools					
JuliaReach	0.03	1.15	155	5,518	–	Julia



(a) JuliaReach. (b) Verse.

Figure 1: Heat3D: Reachable sets for the temperature at the center of the block over time for benchmark version HEAT01.

3.2 Clamped Beam

3.2.1 Model

Similarly to the Heat3D benchmark, the Clamped Beam benchmark [31, Sec. 4.2] also results from the spatial discretization of a partial differential equation (PDE), where each variable represents a node along the beam. The number of states scales proportionally with the number of nodes used for the discretization and the system is influenced by a single external input.

A beam of length L is fixed at one end, while an external load $F(t)$ acts on the other end. Further model parameters are the cross-section area A as well as the Young modulus E and the density ρ of the material. The governing PDE models the displacement $u(x,t)$ as a function of the position x along the beam and the time t: $EAu_{xx} - \rho A u_{tt} = 0$. Here the indices indicate partial derivatives with respect to the indexed variables. To obtain a linear system, the beam is spatially discretized using N nodes from $x = 0$ to $x = L$ depending on the stiffness matrix $K \in \mathbb{R}^{N\times N}$ and the mass matrix $M \in \mathbb{R}^{N\times N}$. The original model is extended by introducing a damping matrix $D = aK + bM$, where $a = b = 10^{-6}$, to model a more realistic beam. Rewriting the equation into a first-order system of linear ODEs yields a system dimension of $2N$ describing the displacement and velocity of each node over time. The sparsity pattern reveals four blocks of size $N \times N$: The block (1,1) is all-zero, the block (1,2) is the identity matrix, and the blocks (2,1) and (2,2) are tridiagonal matrices.

The initial condition is chosen such that all nodes have displacement and velocity zero, i.e., $\forall x \in [0, L] : u(x, 0) = 0, u_t(x, 0) = 0$. The boundary condition keeps the displacement at the fixed end zero at all times so that $\forall t : u(0,t) = 0$. The load is modeled by $F(t) = 10000\, H(t)$ with $H(t)$ denoting the Heaviside function. Finally, the time horizon is set to $T = 0.01$. For discrete-time tools, a step size of $9.88 \cdot 10^{-7}$ is used.

3.2.2 Specifications

The maximum velocity reached at the position $x = 0.7L$ should be below 76.

CB01 $N = 100$ (200 dimensions).

CB02 $N = 500$ (1000 dimensions).

CB03 $N = 1000$ (2000 dimensions).

The load $F(t)$ is modeled in two different ways:

CBC (constant inputs) The inputs are uncertain only in their initial value and constant over time: $F(0) \in \mathcal{F}$, $\dot{F}(t) = 0$, with $\mathcal{F} = [9900, 10100]$.

CBF (time-varying inputs) The inputs can change arbitrarily over time: $\forall t : F(t) \in \mathcal{F}$, with $\mathcal{F} = [9900, 10100]$.

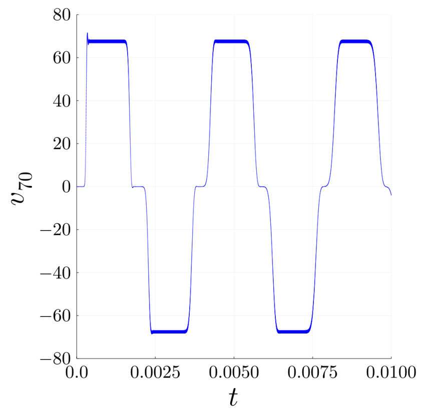
Note that the load $F(t)$ of the original model is different from the input $u(t)$ to the spatially discretized system $\dot{x}(t) = Ax(t) + u(t)$.

3.2.3 Results

Plots for the velocity of the node at $x = 0.7L$, corresponding to node 70 are shown in Figure 2 over the time interval $t \in [0, 0.01]$. The computation times are shown in Table 2.

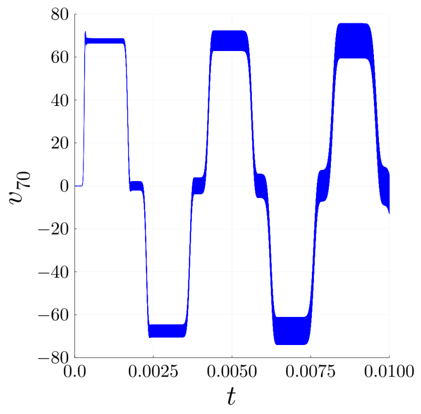
Note CORA For all instances, CORA uses the fully-automated verification algorithm from [36].

Note JuliaReach We use `LGG09` with step sizes 10^{-6} and 10^{-7} for CBC and CBF, respectively. For CBC we additionally use Krylov methods to efficiently compute the large matrix exponentials.



(a) JuliaReach.

Figure 2: Clamped Beam CBC01: Reachable sets for the velocity at node 70. Some tools additionally show possible trajectories.



(a) JuliaReach.

Figure 3: Clamped Beam CBF01: Reachable sets for the velocity at node 70. Some tools additionally show possible trajectories.

Table 2: Computation Times in [s] for the clamped beam benchmark.

tool	**CBC01**	**CBF01**	**CBC02**	**CBF02**	**CBC03**	**CBF03**	**language**
CORA	0.30	0.38	3.50	4.93	40.9	72.4	MATLAB
JuliaReach	7.11	9.61	6.08	47.9	17.2	–	Julia
	discrete-time tools						
JuliaReach	1.2	0.85	2.27	2.32	6.7	8.25	Julia

3.3 International Space Station Benchmark

3.3.1 Model

The International Space Station (ISS) is a continuous linear time-invariant system $\dot{x}(t) = Ax(t) + Bu(t)$ proposed as a benchmark in ARCH 2016 [35]. In particular, the considered system is a structural model of the component 1R (Russian service module), which has 270 state variables with three inputs.

Initially, all 270 variables are in the range $[-0.0001, 0.0001]$, u_1 is in $[0, 0.1]$, u_2 is in $[0.8, 1]$, and u_3 is in $[0.9, 1]$. The time bound is 20. Discrete-time analysis for the space station benchmark should be done with a step size of 0.01. The A, B, and C matrices are available in MATLAB format[2] (that can also be opened with Python using `scipy.io.loadmat`) and in SpaceEx format[3]. There are two versions of this benchmark:

ISSF01 The inputs can change arbitrarily over time: $\forall t : u(t) \in \mathcal{U}$.

ISSC01 (constant inputs) The inputs are uncertain only in their initial value and constant over time: $u(0) \in \mathcal{U}$, $\dot{u}(t) = 0$.

[2]slicot.org/objects/software/shared/bench-data/iss.zip
[3]cps-vo.org/node/34059

3.3.2 Specifications

The verification goal is to check the ranges reachable by the output y_3, which is a linear combination of the state variables ($y = Cx$, $C \in \mathbb{R}^{3\times 270}$). In addition to the safety specification, for each version there is an UNSAT instance that serves as a sanity check to ensure that the model and the tool work as intended. But there is a caveat: In principle, verifying an UNSAT instance only makes sense formally if a witness is provided (counter-example, under-approximation, etc.). Since most of the participating tools do not have this capability, we run the tools with the same accuracy settings on an SAT-UNSAT pair of instances. The SAT instance demonstrates that the over-approximation is not too coarse, and the UNSAT instance demonstrates that the over-approximation is indeed conservative, at least in the narrow sense of the specification.

ISS01 Bounded time, safe property: For all $t \in [0, 20]$, $y_3(t) \in [-0.0007, 0.0007]$. This property is used with the uncertain input case (ISSF01) and assumed to be satisfied.

ISS02 Bounded time, safe property: For all $t \in [0, 20]$, $y_3(t) \in [-0.0005, 0.0005]$. This property is used with the constant input case (ISSC01) and assumed to be satisfied.

ISU01 Bounded time, unsafe property: For all $t \in [0, 20]$, $y_3(t) \in [-0.0005, 0.0005]$. This property is used with the uncertain input case (ISSF01) and assumed to be unsatisfied.

ISU02 Bounded time, unsafe property: For all $t \in [0, 20]$, $y_3(t) \in [-0.00017, 0.00017]$. This property is used with the constant input case (ISSC01) and assumed to be unsatisfied.

3.3.3 Results

Results of the international space station benchmark for state y_3 over time are shown in Fig. 4 and Fig. 5. The computation times of various tools for the benchmark are listed in Tab. 3.

Note CORA CORA applies the fully-automated verification algorithm from [36].

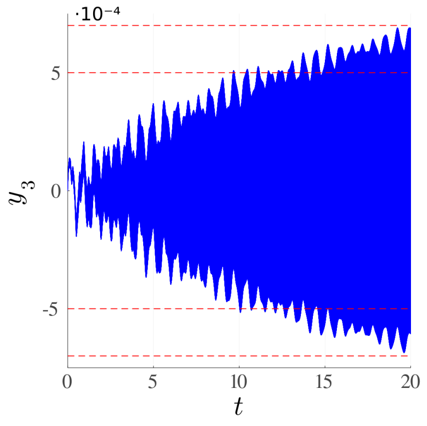
Note JuliaReach We use the `LGG09` implementation. The step sizes in dense time are 6×10^{-4} for ISSF01 and 1×10^{-2} for ISSC01.

Note Verse A simulation time step of .01s was used with 16 simulated trajectories for computing the discrepancy function. For ISSF01, Verse currently does not support inputs that change over time but we may implement this in a later version.

3.4 Spacecraft Rendezvous Benchmark

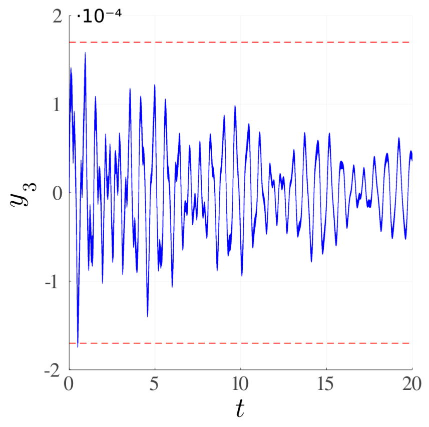
3.4.1 Model

Spacecraft rendezvous is a perfect use case for formal verification of hybrid systems since mission failure can cost lives and is extremely expensive. This benchmark is taken from [17]; its original continuous dynamics is nonlinear, and the original system is verified in the ARCH-COMP category *Continuous and Hybrid Systems with Nonlinear Dynamics*. When spacecraft are in close proximity (such as rendezvous operations), a common approximation to analyze the nonlinear dynamics is to use the linearized Clohessy-Wiltshire-Hill (CWH) equations [19]. This benchmark analyzes this linear hybrid model.

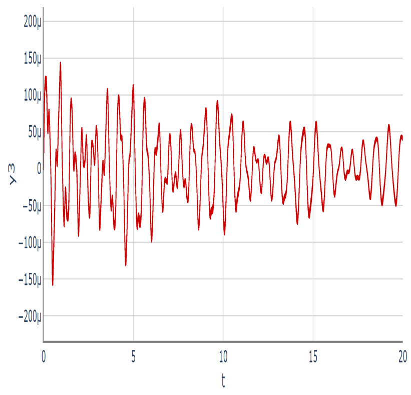


(a) JuliaReach.

Figure 4: ISS: Reachable sets of y_3 plotted over time for the uncertain input case.



(a) JuliaReach.



(b) Verse

Figure 5: ISS: Reachable sets of y_3 plotted over time for the constant input case.

The hybrid nature of this benchmark originates from a switched controller, while the dynamics of the spacecraft is purely continuous. In particular, the modes are *approaching* (100m-1000m), *rendezvous attempt* (less than 100m), and *aborting*. Discrete-time analysis for the rendezvous system should be done with a step size of 0.1. The model is available in C2E2, SDVTool, and SpaceEx format on the ARCH website[4]. The set of initial states is

$$\mathcal{X}_0 = \begin{bmatrix} -900 \\ -400 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} [-25, 25] \\ [-25, 25] \\ 0 \\ 0 \end{bmatrix}.$$

The following benchmark instances are considered:

[4]cps-vo.org/node/36349

Table 3: Computation Times for the International Space Station Benchmark in [s].

	ISSF01		**ISSC01**		
tool	**ISS01**	**ISU01**	**ISS02**	**ISU02**	**language**
CORA	1.12	0.27	0.20	0.34	MATLAB
JuliaReach	5.99	5.99	0.84	0.83	Julia
Verse	–	–	1152.9	1123	Python
	discrete-time tools				
JuliaReach	6.02	6.02	0.84	0.84	Julia

SRNA01 The spacecraft approaches the target as planned and there exists no transition into the *aborting* mode.

SRA01 A transition into *aborting* mode occurs at time $t = 120$ [min].

SRA02 A transition into *aborting* mode occurs nondeterministically, $t \in [120, 125]$ [min].

SRA03 A transition into *aborting* mode occurs nondeterministically, $t \in [120, 145]$ [min].

SRA04 A transition into *aborting* mode occurs at time $t = 240$ [min].

SRA05 A transition into *aborting* mode occurs nondeterministically, $t \in [235, 240]$ [min].

SRA06 A transition into *aborting* mode occurs nondeterministically, $t \in [230, 240]$ [min].

SRA07 A transition into *aborting* mode occurs nondeterministically, $t \in [50, 150]$ [min].

SRA08 A transition into *aborting* mode occurs nondeterministically, $t \in [0, 240]$ [min].

An initial, discrete-time analysis indicated it is safe to enter the *aborting* mode up to around time $t = 250$ [min]. We also added the following two instances, which are presumably unsafe. For timing, tools should use the same settings for these as for the safe cases.

SRU01 A transition into *aborting* mode occurs at time $t = 260$ [min].

SRU02 A transition into *aborting* mode occurs nondeterministically, $t \in [0, 260]$ [min].

3.4.2 Specifications

Given the thrust constraints of the specified model, in mode *rendezvous attempt*, the absolute velocity must stay below 0.055 m/s. In the *aborting* mode, the vehicle must avoid the target, which is modeled as a box $\mathcal{B}$ with 0.2 m edge length and the center placed as the origin. In the *rendezvous attempt* the spacecraft must remain within the line-of-sight cone $\mathcal{L} = \{[x, y]^T \mid (x \geq -100\,m) \wedge (y \geq x\ \tan(30°)) \wedge (-y \geq x \tan(30°))\}$. It is sufficient to check these parameters for a time horizon of 300 minutes.

Let us denote the discrete state by $z(t)$ and the continuous state vector by $x(t) = [s_x, s_y, v_x, v_y]^T$, where s_x and s_y are the positions in x- and y-direction, respectively, and v_x and v_y are the velocities in x- and y-direction, respectively. The mode *approaching* is denoted by z_1, the mode *rendezvous attempt* by z_2, and the mode *aborting* by z_3. We can formalize the specification as

SR02 $\forall t \in [0, 300\,min], \forall x(0) \in \mathcal{X}_0 : (z(t) = z_2) \implies \Big(\sqrt{v_x^2 + v_y^2} \leq 0.055\,m/s \wedge [s_x, s_y]^T \in \mathcal{L}\Big) \wedge (z(t) = z_3) \implies ([s_x, s_y]^T \notin \mathcal{B})$.

To solve the above specification, all tools under-approximate the nonlinear constraint $\sqrt{v_x^2 + v_y^2} \leq 0.055\,m/s$ by an octagon as shown in Fig. 6.

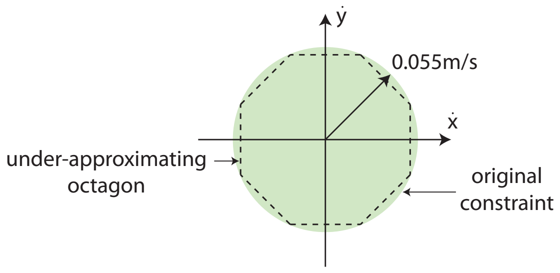


Figure 6: Under-approximation of the nonlinear velocity constraint by an octagon.

Remark on nonlinear constraint In the original benchmark, the constraint on the velocity was set to 0.05 m/s, but it can be shown that this constraint cannot be satisfied by a counterexample. For this reason, we have relaxed the constraint to 0.055 m/s.

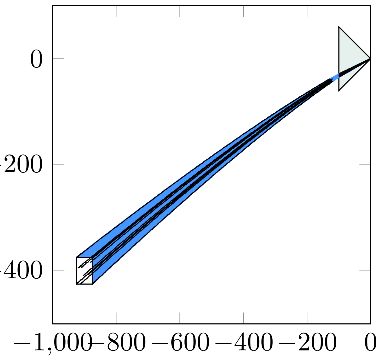
3.4.3 Results

Results of the spacecraft rendezvous benchmark for the s_x-s_y-plane are shown for the version SRNA01 in Fig. 7 and for the version SRA01 in Fig. 8. The computation times of various tools for the spacecraft rendezvous benchmark are listed in Tab. 4.

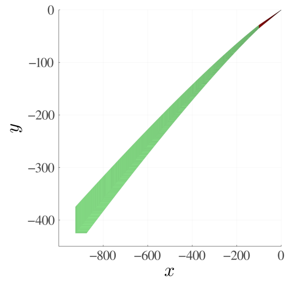
Note CORA For both benchmark versions, CORA was run with a zonotope order of 10 and with the following step sizes: 0.2 [min] for the mode *approaching*, 0.02 [min] for the mode *rendezvous attempt*, and 0.2 [min] for the mode *aborting* (does not exist for version SRNA01). Intersections with deterministic guards are calculated with the method of Girard and Le Guernic in [25]. In order to find suitable orthogonal directions for the method in [25], we perform the following procedure: first, we project the last zonotope not intersecting the guard set onto the guard set; second, we apply principal component analysis to the generators of the projected zonotope, providing us with the orthogonal directions. For non-deterministic guards we first unite all reachable sets intersecting the guard set and then compute the intersection using constrained zonotopes [33].

Note JuliaReach We use `BFFPSV18` with a one-block partition and hyperrectangular reach sets with step size 0.04 (in dense time) for instances SRA01-SRA03. We handle discrete transitions by computing the intersection with invariants and guards lazily before their overapproximation with a hyperrectangle. For the instance SRA04, we use a clustering strategy of order 40 and step size 0.01. For the unsat instances we use the step size 0.04.

Note Verse The run time is significantly longer for examples with uncertain switches due to the amount of extra computation. In addition, the initial segment length was reduced to 10 for cases SRA04 to SRU02 to reduce overapproximation inaccuracies from rectangle merging. A time step of .1s was used for all cases. We use 16 simulated trajectories for computing the discrepancy function.



(a) CORA.



(b) JuliaReach.

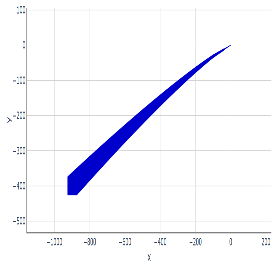
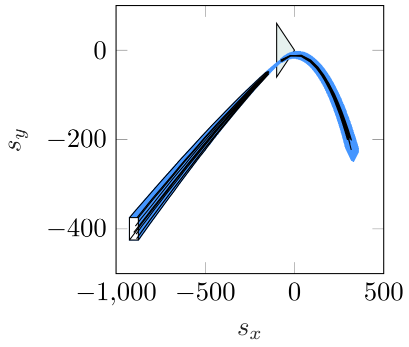
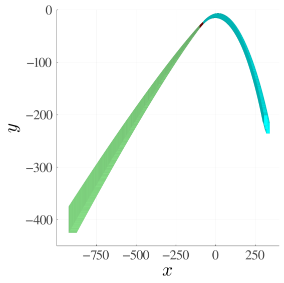

(c) Verse.

Figure 7: Reachable sets for the spacecraft rendezvous benchmark in the s_x-s_y-plane for the benchmark variant without maneuver abortion (SRNA01).



(a) CORA.



(b) JuliaReach.

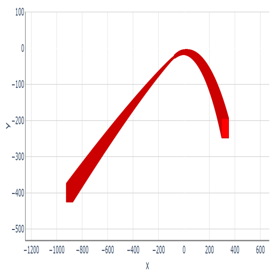

(c) Verse.

Figure 8: Reachable sets for the spacecraft rendezvous benchmark in the s_x-s_y-plane for the benchmark variant with maneuver abortion at $t = 120$ [min] (SRA01, over analysis time horizon of 300 [min])

3.5 Powertrain with Backlash

3.5.1 Model

The powertrain benchmark is an extensible benchmark for hybrid systems with linear continuous dynamics taken from [7, Sec. 6] and [12, Sec. 4]. The essence of this benchmark is recalled here, and the reader is referred to the above-cited papers for more details. The benchmark considers the powertrain of a vehicle consisting of its motor and several rotating masses representing

Table 4: Computation time [s] for the spacecraft rendezvous benchmarks (**SR***) for specification **SR02**.

tool	**NA01**	**A01**	**A02**	**A03**	**A04**	**A05**	**A06**	**A07**	**A08**	**U01**	**U02**
CORA	16.2	3.82	4.96	5.80	14.2	14.1	30.9	74.2	16.2	13.3	74.7
JuliaReach	2.6	2.98	2.67	3.8	236	–	–	–	–	26.5	53.4
Verse	12.8	25.7	28.3	98.4	75.4	27.8	37.3	135.8	415.4	17.5	401.9
	discrete-time tools										
JuliaReach	1.01	0.95	1.02	1.47	–	–	–	–	–	–	32.1

different components of the powertrain, e.g., gears, differential, and clutch, as illustrated in Fig. 9. The benchmark is extensible in the sense that the number of continuous states can be easily extended to $n = 7+2\theta$, where θ is the number of additional rotating masses. The number of discrete modes, however, is fixed and originates from backlash, which is caused by a physical gap between two components that are normally touching, such as gears. When the rotating components switch direction, for a short time they temporarily disconnect, and the system is said to be in the *dead zone*. The model is available in SpaceEx format on the ARCH website[5]. The set of initial states is

$$\begin{aligned}\mathcal{X}_0 &= \{c + \alpha g \mid \alpha \in [-1, 1]\},\\ c &= [-0.0432, -11, 0, 30, 0, 30, 360, -0.0013, 30, \ldots, -0.0013, 30]^T,\\ g &= [0.0056, 4.67, 0, 10, 0, 10, 120, 0.0006, 10, \ldots, 0.0006, 10]^T.\end{aligned}$$

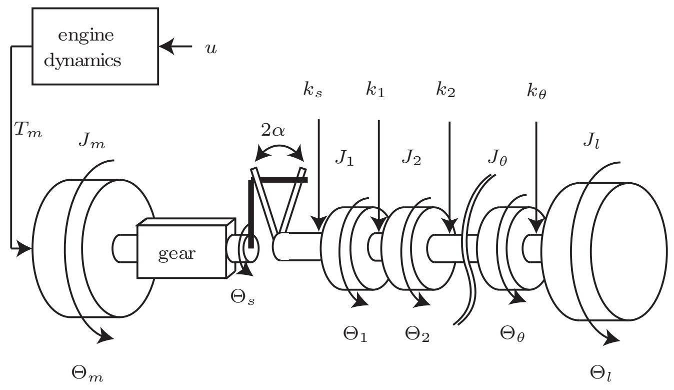


Figure 9: Powertrain model.

[5] cps-vo.org/node/49115

3.5.2 Specifications

We analyze an extreme maneuver from a maximum negative acceleration that lasts for 0.2 [s], followed by a maximum positive acceleration that lasts for 1.8 [s]. The initial states of the model are on a line segment in the n-dimensional space. We create different difficulty levels of the reachability problem by scaling down the initial states by some percentage. The model has the following non-formal specification: after the change of direction of acceleration, the powertrain completely passes the dead zone before being able to transmit torque again. Due to oscillations in the torque transmission, the powertrain should not re-enter the dead zone of the backlash.

To formalize the specification using linear time logic (LTL), let us introduce the following discrete states:

- z_1 : left contact zone
- z_2 : dead zone
- z_3 : right contact zone

For all instances, the common specification is: For all $t \in [0, 2]$, $x(0) \in \mathcal{X}_0$, $(z_2 U z_3) \implies G(z_3)$. The instances only differ in the size of the system and the initial set, where $\texttt{center}(\cdot)$ returns the volumetric center of a set.

DTN01 $\theta = 2$, $\mathcal{X}_0 := 0.05(\mathcal{X}_0 - \texttt{center}(\mathcal{X}_0)) + \texttt{center}(\mathcal{X}_0)$.

DTN02 $\theta = 2$, $\mathcal{X}_0 := 0.3(\mathcal{X}_0 - \texttt{center}(\mathcal{X}_0)) + \texttt{center}(\mathcal{X}_0)$.

DTN03 $\theta = 2$, no change of $\mathcal{X}_0$.

DTN04 $\theta = 22$, $\mathcal{X}_0 := 0.05(\mathcal{X}_0 - \texttt{center}(\mathcal{X}_0)) + \texttt{center}(\mathcal{X}_0)$.

DTN05 $\theta = 22$, $\mathcal{X}_0 := 0.3(\mathcal{X}_0 - \texttt{center}(\mathcal{X}_0)) + \texttt{center}(\mathcal{X}_0)$.

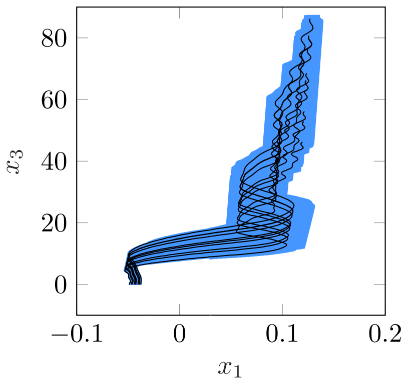
DTN06 $\theta = 22$, no change of $\mathcal{X}_0$.

3.5.3 Results

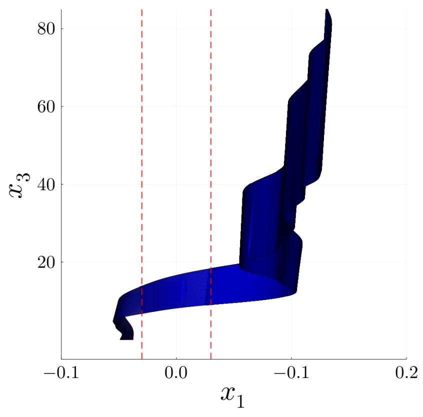
Results of the powertrain benchmark in the x_1-x_3-plane are shown in Fig. 10. The computation times of various tools for the powertrain benchmark are listed in Tab. 5.

Note CORA CORA uses the following time step sizes: $0.005s$ for DTN01, DTN02, and DTN03; $0.002s$ for DTN04; and $0.001s$ for DTN05 and DTN06. For all benchmark versions, CORA was run with a zonotope order of 20. The intersections with the guard sets are calculated with the approach from [25], and principal component analysis is used to find suitable directions for the enclosure of the guard intersections.

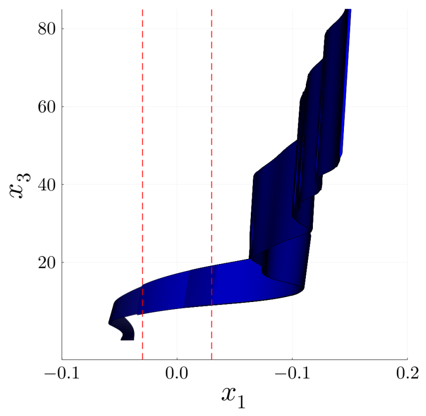
Note JuliaReach We use the `GLGM06` algorithm with a step size 0.001. In addition we use slightly different analysis parameters for different modes.



(a) CORA (DTN03).



(b) JuliaReach (DTN03).



(c) JuliaReach (DTN05).

Figure 10: Reachable sets in the x_1-x_3-plane.

Table 5: Computation Times for the Powertrain Benchmark in [s].

tool	**DTN01**	**DTN02**	**DTN03**	**DTN04**	**DTN05**	**DTN06**	**language**
CORA	8.0	6.4	6.1	43.7	77.4	204	MATLAB
JuliaReach	0.18	0.15	0.11	0.27	0.75	0.89	Julia

3.6 Platooning Benchmark

3.6.1 Model

The platooning benchmark considers a platoon of three vehicles following each other. This benchmark considers loss of communication between vehicles. The initial discrete state is q_c. Three scenarios are considered for the loss of communication:

PLAA01 (arbitrary loss) The loss of communication can occur at any time, see Fig. 11(a). This includes the possibility of no communication at all.

PLADxy (loss at deterministic times) The loss of communication occurs at fixed points in time, which are determined by clock constraints c_1 and c_2 in Fig. 11(b). The clock t is reset when communication is lost and when it is re-established. Note that the transitions have must-semantics, i.e., they take place as soon as possible.

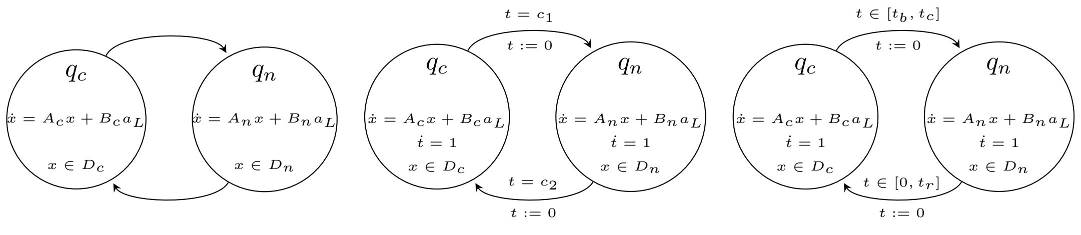
PLAD01: $c_1 = c_2 = 5$.

PLANxy (loss at nondeterministic times) The loss of communication occurs at any time $t \in [t_b, t_c]$ in Fig. 11(c). The clock t is reset when communication is lost and when it is re-established. Communication is reestablished at any time $t \in [0, t_r]$. This scenario covers loss of communication after an arbitrarily long time $t \geq t_c$ by reestablishing communication in zero time.

PLAN01: $t_b = 10$, $t_c = 20$, $t_r = 20$.

The models are available in SpaceEx, KeYmaera, and MATLAB/Simulink format on the ARCH website[6]. Discrete-time analysis for the platoon system should use a step size of 0.1.

Discussion The arbitrary-loss scenario (PLAA) subsumes the other two instances (PLAD, PLAN).



(a) Arbitrary switching. (b) Deterministic switching. (c) Nondeterministic switching.

Figure 11: Three options adapted from the original benchmark proposal [14]. On the left, the system can switch arbitrarily between the modes. In the middle, mode switches are only possible at given points in time. On the right, mode switches are only possible during given time intervals.

3.6.2 Specifications

The verification goal is to check whether the minimum distance between vehicles is preserved. The choice of the coordinate system is such that the minimum distance is a negative value.

BNDxy Bounded time (no explicit bound on the number of transitions): For all $t \in [0, 20]$ [s], $x_1(t) \geq -d_{min}$ [m], $x_4(t) \geq -d_{min}$ [m], and $x_7(t) \geq -d_{min}$ [m].

BND50: $d_{min} = 50$.

BND42: $d_{min} = 42$.

[6] cps-vo.org/node/15096

BND30: $d_{min} = 30$.

UNBxy Unbounded time and unbounded switching: For all $t \geq 0$ [s], $x_1(t) \geq -d_{min}$ [m], $x_4(t) \geq -d_{min}$ [m], and $x_7(t) \geq -d_{min}$ [m].

UNB50: $d_{min} = 50$.

UNB42: $d_{min} = 42$.

UNB30: $d_{min} = 30$.

3.6.3 Results

Results of the platoon benchmark for state x_1 over time are shown in Fig. 12-14. The computation times of various tools for the platoon benchmark are listed in Tab. 6.

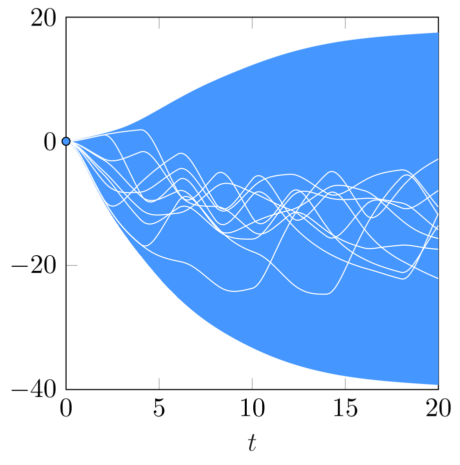
Note CORA CORA was run with the following settings:

- PLAA01-BND50: zonotope order 400 and time step size 0.02.
- PLAA01-BND42: zonotope order 800 and time step size 0.009.
- PLAD01-BND42: zonotope order 20 and time step size $0.02s$.
- PLAD01-BND30: zonotope order 200 and time step size 0.02.
- PLAN01-UNB50: zonotope order 400 and time step size 0.01. To verify the specification for all times, the reachable set was increased by 1% at $t = 50$ and it was checked whether this set is re-entered.
- PLAA01: we used *continuization* [8, 9] to rewrite the hybrid automaton as a purely continuous system with uncertain parameters.

Note JuliaReach For PLAD01-BND42, we use the `BFFPSV18` algorithm with a one-block partition, hyperrectangular reach-sets, and a step size (in dense time) 0.01. For PLAD01-BND30, we use the `LGG09` algorithm with a step size (in dense time) 0.03, and intersections with the guard are taken lazily with an octagonal template.

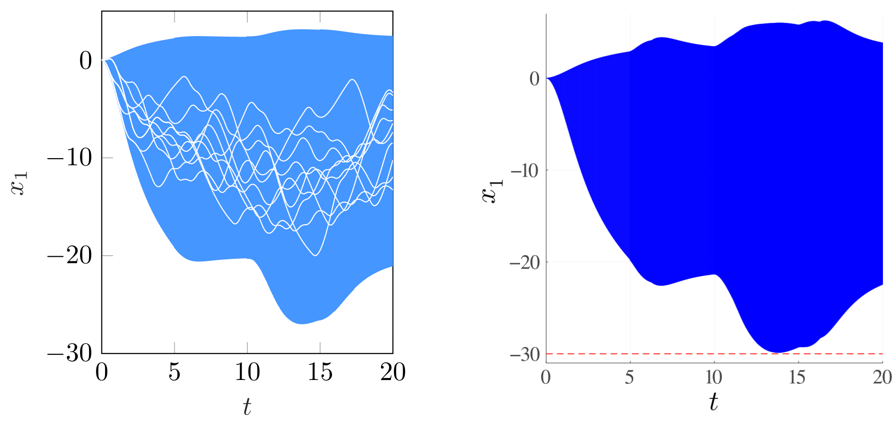
Table 6: Computation Times for the Platoon Benchmark in [s].

tool	PLAA01 BND50	PLAA01 BND42	PLAD01 BND42	PLAD01 BND30	PLAN01 UNB50	language
CORA	12.2	42.3	2.4	3.0	96	MATLAB
JuliaReach	–	–	0.17	97.5	–	Julia
	discrete-time tools					
JuliaReach	–	–	3.31	3.71	–	Julia



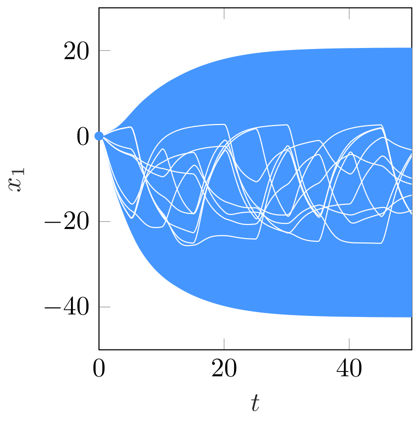
(a) CORA.

Figure 12: PLAA01: Reachable sets of x_1 plotted over time. CORA additionally shows possible trajectories.



(a) CORA (BND30). (b) JuliaReach (BND30).

Figure 13: PLAD01: Reachable sets of x_1 plotted over time. Some tools additionally show possible trajectories.



(a) CORA (UNB50).

Figure 14: PLAN01: Reachable sets of x_1 plotted over time.

3.7 Gearbox Benchmark

3.7.1 Model

The gearbox benchmark models the motion of two meshing gears. When the gears collide, an elastic impact takes place. As soon as the gears are close enough, the gear is considered *meshed*. The model includes a monitor state that checks whether the gears are meshed or free and is available in SpaceEx format[7] and as a Simulink model[8]. Once the monitor reaches the state *meshed*, it stays there indefinitely.

With four continuous state variables, the gearbox benchmark has a relatively low number of continuous state variables. The challenging aspect of this benchmark is that the solution heavily depends on the initial state as already pointed out in [18]. For some initial continuous states, the target region is reached without any discrete transition, while for other initial states, several discrete transitions are required.

In the original benchmark, the position uncertainty in the direction of the velocity vector of the gear teeth (x-direction) is across the full width of the gear spline. Uncertainties of the position and velocity in y-direction, which is perpendicular to the x-direction, are considered to be smaller. Due to the sensitivity with respect to the initial set, we consider smaller initial sets. The full uncertainty in x-direction could be considered by splitting the uncertainty in x-direction and aggregating the individual results. For discrete-time analysis of the gearbox system, a step size of 0.0001 (1.0E-4) should be used.

GRBX01: The initial set is $\mathcal{X}_0 = 0 \times 0 \times [-0.0168, -0.0166] \times [0.0029, 0.0031] \times 0$.

GRBX02: The initial set is $\mathcal{X}_0 = 0 \times 0 \times [-0.01675, -0.01665] \times [0.00285, 0.00315] \times 0$.

[7]cps-vo.org/node/34375
[8]cps-vo.org/node/34374

3.7.2 Specification

The goal is to show that the gears are *meshed* within a time frame of 0.2 [s] and that the bound $x_5 \leq 20$ [Nm] of the cumulated impulse is met. Using the monitor states *free* and *meshed*, and a global clock t, this can be expressed as a safety property as follows: For all $t \geq 0.2$, the monitor should be in *meshed*. Under nonblocking assumptions, this means that $t < 0.2$ whenever the monitor is not in *meshed*, i.e., when it is in *free*.

MES01: forbidden states: $(\mathit{free} \wedge t \geq 0.2) \vee (x_5 \geq 20)$

3.7.3 Results

Results of the benchmark for state x_3 and x_4 are shown in Fig. 15. The computation times of various tools for the benchmark are listed in Tab. 7.

Note CORA CORA was run with a time step size of 0.0011 and a zonotope order of 20. The intersections with the guard sets were calculated with the method of Girard and Le Guernic [25]. In order to find suitable orthogonal directions for the method in [25], we perform the following procedure: first, we project the last zonotope not intersecting the guard set onto the guard set; second, we apply principal component analysis to the generators of the projected zonotope, providing us with the orthogonal directions.

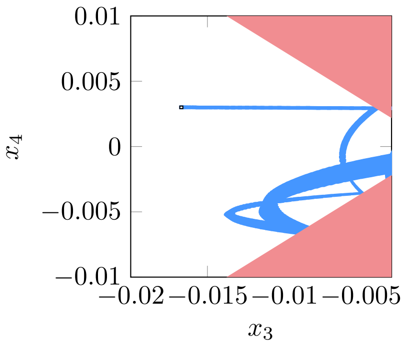
Note JuliaReach We use the `LGG09` algorithm with step sizes (in dense time) 0.0005 (GRBX01) resp. 0.0008 (GRBX02).

Note Verse A simulation timestep of 1e-4s was used. Due to an over-approximation error, Verse is only able to run this benchmark with a time horizon of up to .11s. We are improving the tool to reduce overapproximation to solve this benchmark.

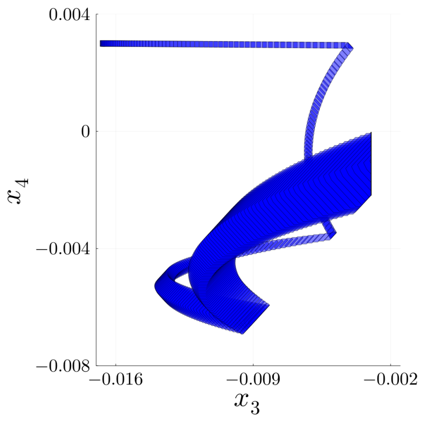
Table 7: Computation Times of the Gearbox Benchmark in [s].

tool	**GRBX01-MES01**	**GRBX02-MES01**	**language**
CORA	1.5	0.77	MATLAB
JuliaReach	16.9	11	Julia
Verse*	26.5	27.2	Python
	discrete-time tools		
JuliaReach	59	61.1	Julia

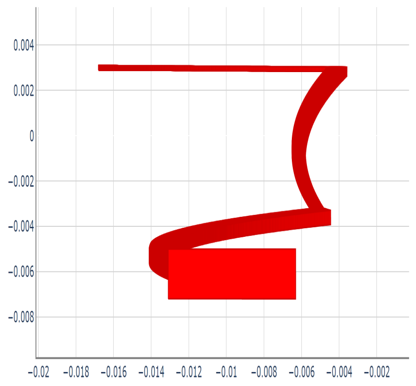
*Verse is only able to run this benchmark up to a time horizon of 1.1s



(a) CORA.



(b) JuliaReach.



(c) Verse.

Figure 15: Gearbox (GRBX01): Reachable sets of x_3 and x_4.

3.8 Brake Benchmark

3.8.1 Model

The brake benchmark models an electro-mechanical braking system, where a motor pushes a brake caliper against a brake disk that is connected to a (car) wheel [34]. The model describes a closed-loop system comprising a plant model as well as a controller and is representative for challenges in automotive systems. The original Simulink model has been simplified for usage in various analysis tools[9]. Here, we consider a linearized version with parameters.

The model is a hybrid automaton (see Fig. 16) with four state variables (the motor current I, the brake position x, and two auxiliary linearization variables) and a clock variable T. The automaton consists of a single mode and a self-loop transition. The transition is time-triggered, i.e., it only depends on the value of the clock variable.

[9]cps-vo.org/node/20289

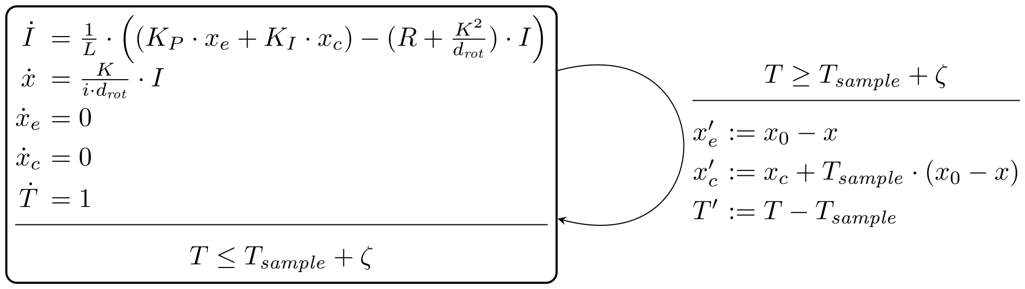

$\dot{I} = \frac{1}{L} \cdot \Big((K_P \cdot x_e + K_I \cdot x_c) - (R + \frac{K^2}{d_{rot}}) \cdot I \Big)$
$\dot{x} = \frac{K}{i \cdot d_{rot}} \cdot I$
$\dot{x}_e = 0$
$\dot{x}_c = 0$
$\dot{T} = 1$
$T \leq T_{sample} + \zeta$
$T \geq T_{sample} + \zeta$
$x_e' := x_0 - x$
$x_c' := x_c + T_{sample} \cdot (x_0 - x)$
$T' := T - T_{sample}$

Figure 16: Hybrid automaton of the electro-mechanical brake with periodic discrete-time PI controller and sampling jitter.

We consider two types of uncertainties in the model. The first uncertainty is a variation in the model parameters. We use the settings from [34] for the nonparametric and parametric scenarios. The second uncertainty is sampling jitter (i.e., nondeterministic switching). Unlike the linear model in [34], we consider jitter with a periodic clock (instead of a drifting clock).

3.8.2 Specification

While structurally simple, the benchmark is challenging due to the large number of 1,001 discrete jumps within the time horizon 0.1. The initial state is the origin, we use the parameters $x_0 = 0.05$ and $T_{sample} = 10^{-4}$, and in the case of nondeterministic switching the transitions are taken at multiples of T_{sample} with a nondeterministic jitter from the interval $\zeta = [-10^{-8}, 10^{-7}]$. We study the property $x < x_0$ in both scenarios without and with parameter ranges:

BRKDC01: Verify that $x < x_0$ holds for the whole time horizon 0.1 (non-parametric scenario with deterministic switching).

BRKNC01: Same as BRKDC01, but with non-deterministic switching.

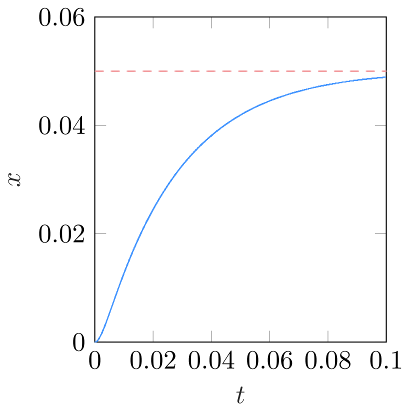
BRKNP01: Report the largest time horizon for which $x < x_0$ holds (parametric scenario with non-deterministic switching).

3.8.3 Results

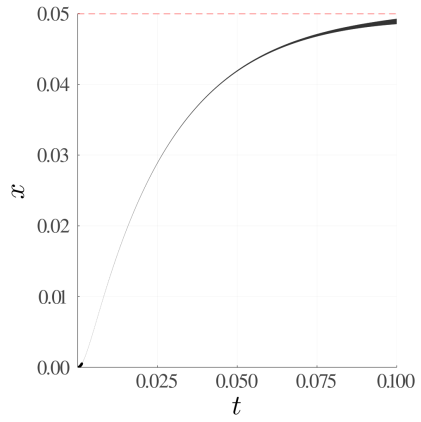
Results of the benchmark are shown in Fig. 17. The computation times of various tools for the benchmark are listed in Tab. 8.

Note CORA CORA was run with a time step size of 2^{-6}, a zonotope order of 20, and the intersections with the non-deterministic guard sets were calculated with constrained zonotopes [33].

Note JuliaReach For the BRKDC01 and BRKNC01 scenario, we use the `GLGM06` algorithm with fixed step sizes 10^{-7} resp. 10^{-8}. For the BRKNP01 scenario, we use the `ASB07` algorithm with step size 10^{-8}. In all scenarios we use the maximum zonotope order 1 (i.e., boxes). The discrete-time instances use the same step sizes as the dense-time instances. We use a custom analysis for dealing with the time-triggered transition efficiently, considering intersections with the guard separately from the flowpipe computation, as described in [21]. The largest time horizon for which $x < x_0$ can be proven in the BRKNP01 scenario is 0.0823s and 0.0824s in dense and discrete time, respectively.



(a) CORA.



(b) JuliaReach.

Figure 17: Brake: Reachable sets for x over time. Some tools additionally show possible trajectories.

Table 8: Computation Times of the Brake Benchmark in [s].

tool	BRKDC01	BRKNC01	BRKP01	language
CORA	23.7	–	550	MATLAB
JuliaReach	0.59	1.3	8.98	Julia
	discrete-time tools			
JuliaReach	0.05	0.24	8.85	Julia

4 Conclusion and Outlook

This report presents the results of the seventh friendly competition for the formal verification of continuous and hybrid systems with linear continuous dynamics as part of the ARCH'23 workshop. The reports of other categories can be found in the proceedings and on the ARCH website: cps-vo.org/group/ARCH.

A major observation of the results is that participating tools have significantly reduced computation times compared to the previous year. One can execute the dockerfiles of all tools on gitlab.com/goranf/ARCH-COMP. Information about the competition in 2024 will be announced on the ARCH website.

5 Acknowledgments

Matthias Althoff gratefully acknowledge financial support by the project justITSELF funded by the European Commission under grant number 817629 and by the project TRAITS funded by the German Federal Ministry of Education and Research under grant number 01IS21087.

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

References

[1] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars.* Dissertation, Technische Universität München, 2010. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20100715-963752-1-4.

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

[3] M. Althoff. Reachability analysis of large linear systems with uncertain inputs in the Krylov subspace. *IEEE Transactions on Automatic Control*, 65(2):477–492, 2020.

[4] M. Althoff. Guaranteed state estimation in CORA 2021. In G. Frehse and M. Althoff, editors, *Proc. of the 8th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 80 of *EPiC Series in Computing*, page 161–175. EasyChair, 2021.

[5] 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*, page 91–105, 2016.

[6] M. Althoff, D. Grebenyuk, and N. Kochdumper. Implementation of Taylor models in CORA 2018. In *Proc. of the 5th International Workshop on Applied Verification for Continuous and Hybrid Systems*, page 145–173, 2018.

[7] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Hybrid Systems: Computation and Control*, page 45–54, 2012.

[8] M. Althoff, C. Le Guernic, and B. H. Krogh. Reachable set computation for uncertain time-varying linear systems. In *Hybrid Systems: Computation and Control*, page 93–102, 2011.

[9] M. Althoff, A. Rajhans, B. H. Krogh, S. Yaldiz, X. Li, and L. Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. *Communications of the ACM*, 56(10):97–104, 2013.

[10] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of linear systems with uncertain parameters and inputs. In *Proc. of the 46th IEEE Conference on Decision and Control*, page 726–732, 2007.

[11] Matthias Althoff, Marcelo Forets, Christian Schilling, and Mark Wetzlinger. ARCH-COMP22 category report: Continuous and hybrid systems with linear continuous dynamics. In *Proc. of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 90 of *EPiC Series in Computing*, page 58–85. EasyChair, 2022.

[12] S. Bak, S. Bogomolov, and M. Althoff. Time-triggered conversion of guards for reachability analysis of hybrid automata. In *Proc. of the 15th International Conference on Formal Modelling and Analysis of Timed Systems*, page 133–150, 2017.

[13] Stanley Bak, Hoang-Dung Tran, and Taylor T Johnson. Numerical verification of affine systems with up to a billion dimensions. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 23–32, 2019.

[14] I. Ben Makhlouf and S. Kowalewski. Networked cooperative platoon of vehicles for testing methods and verification tools. In *Proc. of ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, page 37–42, 2015.

[15] 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.

[16] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. JuliaReach: a toolbox for set-based reachability. In *HSCC*, pages 39–44. ACM, 2019.

[17] N. Chan and S. Mitra. Verifying safety of an autonomous spacecraft rendezvous mission. In *Proc. of the 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, page 20–32, 2017.

[18] H. Chen, S. Mitra, and G. Tian. Motor-transmission drive system: a benchmark example for safety verification. In *Proc. of ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, page 9–18, 2015.

[19] WH Clohessy. Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, 1960.

[20] Chuchu Fan, Bolun Qi, Sayan Mitra, and Mahesh Viswanathan. Dryvr: Data-driven verification and compositional reasoning for automotive systems. In Rupak Majumdar and Viktor Kunčak, editors, *Computer Aided Verification*, pages 441–461, Cham, 2017. Springer International Publishing.

[21] Marcelo Forets, Daniel Freire, and Christian Schilling. Efficient reachability analysis of parametric linear hybrid systems with time-triggered transitions. In *MEMOCODE*, pages 1–6. IEEE, 2020.

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

[23] Marcelo Forets and Christian Schilling. Conservative time discretization: A comparative study. In *iFM*, volume 13274 of *LNCS*, pages 149–167. Springer, 2022.

[24] Victor Gaßmann and Matthias Althoff. Implementation of ellipsoidal operations in CORA 2022. In *Proc. of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems*, page 1–17, 2022.

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

[26] Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *HSCC*, volume 3927 of *LNCS*, pages 257–271. Springer, 2006.

[27] Zhi Han. *Formal verification of hybrid systems using model order reduction and decomposition*. PhD thesis, PhD thesis, Dept. of ECE, Carnegie Mellon University, 2005.

[28] Zhi Han and Bruce H Krogh. Reachability analysis of large-scale affine systems using low-dimensional polytopes. In *International Workshop on Hybrid Systems: Computation and Control*, pages 287–301. Springer, 2006.

[29] Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

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

[31] Mohammad Mahdi Malakiyeh, Saeed Shojaee, and Klaus-Jürgen Bathe. The bathe time integration method revisited for prescribing desired numerical dissipation. *Computers & Structures*, 212:289–298, 2019.

[32] Christopher Rackauckas and Qing Nie. DifferentialEquations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia. *The Journal of Open Research Software*, 5(1), 2017.

[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] Thomas Strathmann and Jens Oehlerking. Verifying properties of an electro-mechanical braking system. In *ARCH@CPSWeek*, volume 34 of *EPiC Series in Computing*, pages 49–56. EasyChair, 2015.

[35] H.-D. Tran, L. V. Nguyen, and T. T. Johnson. Large-scale linear systems from order-reduction. In *Proc. of ARCH16. 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, page 60–67, 2017.

[36] Mark Wetzlinger, Niklas Kochdumper, Stanley Bak, and Matthias Althoff. Fully-automated verification of linear systems using reachability analysis with support functions. In *Proc. of the 26th ACM International Conference on Hybrid Systems: Computation and Control*, 2023.