EPiC Series in Computing
Volume 74, 2020, Pages 16–48
ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)

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

Matthias Althoff[1], Stanley Bak[2], Zongnan Bao[8], Marcelo Forets[3], Goran Frehse[4], Daniel Freire[3], Niklas Kochdumper[1], Yangge Li[8], Sayan Mitra[8], Rajarshi Ray[5], Christian Schilling[6], Stefan Schupp[7], and Mark Wetzlinger[1]

[1] Technical University of Munich, Department of Informatics, Munich, Germany
althoff@in.tum.de, niklas.kochdumper@tum.de, m.wetzlinger@tum.de
[2] Stony Brook University, Stony Brook, NY, United States
stanleybak@gmail.com
[3] Universidad de la República, Montevideo, Uruguay
mforets@gmail.com, dfreire@fisica.edu.uy
[4] ENSTA Paris, Palaiseau, France
goran.frehse@ensta-paris.fr
[5] Indian Association for the Cultivation of Science, Kolkata, India.
rajarshi.ray@iacs.res.in
[6] IST Austria, Klosterneuburg, Austria
christian.schilling@ist.ac.at
[7] RWTH Aachen University, Theory of hybrid systems, Aachen, Germany
stefan.schupp@cs.rwth-aachen.de
[8] University of Illinois at Urbana-Champaign, Champaign, IL, United States
mitras@illinois.edu,li213@illinois.edu,zb3@illinois.edu

Abstract

We present the results of the ARCH[1] 2020 friendly competition for formal verification of continuous and hybrid systems with linear continuous dynamics. In its fourth edition, eight tools have been applied to solve eight different benchmark problems in the category for linear continuous dynamics (in alphabetical order): CORA, C2E2, HyDRA, Hylaa, Hylaa-Continuous, JuliaReach, SpaceEx, and XSpeed. 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]Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH), cps-vo.org/group/ARCH

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.

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.

A novelty compared to last year is that all tools are run on a AWS g4dn.4xlarge machine with 16 Xeon vCPUs and 64 GB RAM. Although we use the same machine, one still has to factor in the efficiency of the programming language of the tools.

2 Participating Tools

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

CORA The tool *COntinuous Reachability Analyzer* (CORA) [2, 4, 5] 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. CORA is available at cora.in.tum.de.

C2E2 The tool Compare Execute Check Engine (C2E2) is a tool for verifying time-bounded invariant properties of hybrid automata models [24, 22, 25]. It supports models with non-linear dynamics, discrete transitions, and sets of initial states. Internally, C2E2 implements simulation-based verification algorithms. C2E2 uses an on-the-fly discrepancy computation algorithm to automatically generate neighborhoods that conservatively contain all the behaviors of neighboring trajectories.

The most recent version of C2E2 also implements a simulation-based verification algorithm from [23], which uses generalized star sets and exploits the superposition principle of linear systems to compute an over approximation of the sets of reachable states. C2E2 is available at `http://publish.illinois.edu/c2e2-tool/`.

HyDRA The Hybrid systems Dynamic Reachability Analysis (HyDRA) tool implements flow-pipe construction based reachability analysis for linear hybrid automata. The tool is built on top of HyPro [39] available at ths.rwth-aachen.de/research/projects/hypro/, a C++ library for reachability analysis. HyPro provides different implementations of set representations tailored for reachability analysis, such as boxes, convex polyhedra, support functions, or zonotopes, all sharing a common interface. This interface allows one to easily exchange the utilized set representation in HyDRA. We use this to extend state-of-the art reachability analysis by CEGAR-like parameter refinement loops, which (among other parameters) allow us to vary the used set representation. Furthermore, HyDRA incorporates the capability to explore different branches of the search tree in parallel. Being in an early stage of development, HyDRA already shows promising results on some benchmarks, although there is still room for improvements. An official first release is planned.

Hylaa The tool Hylaa [12, 13] computes the reachable set using discrete-time semantics. Hylaa is a Python-based tool, which can produce live plots during computation, as well as images and video files of the reachable set. Hylaa's website is `http://stanleybak.com/hylaa`.

The biggest differences with discrete-time semantics is that safety is not checked between time steps, and time-varying inputs are constant between time steps, rather than considering all possible input frequencies [14]. The benefit of this is that Hylaa's analysis is exact with respect to discrete-time semantics and it can always generate a counter-example when specifications are violated. Note that continuous-time and discrete-time semantics are incomparable, neither one is strictly contained in the other. However, when small time steps are used, the results are qualitatively similar, as can be seen from some of the reach set plots in this year's competition.

Hylaa-Continuous For the Heat3D benchmark we also included a measurement for the *continuous* branch in the Hylaa repository, which uses initial space / output space projections as well as Krylov-subspace methods to speed up verification for high-dimensional continuous systems [15].

JuliaReach *JuliaReach* is a software suite for reachability computations of dynamical systems [17], available at `http://juliareach.com/`. It is written in Julia, a modern high-level language for scientific computing. For the set computations we use the independent *LazySets* library, which is also part of JuliaReach. JuliaReach can also analyze systems with discrete-time semantics as in Hylaa. For some of the models we use our custom *SX* parser for parsing SX (SpaceEx format) model files, and otherwise we create the models in Julia using our *MathematicalSystems* package. Compared to last year we have developed *ReachabilityAnalysis*, with many improved features such as higher performance, a simplified user interface, and implementations of various reachability algorithms from the literature which we call `BFFPSV18` (based on support functions on low-dimensional subspaces [18]), `GLGM06` (based on zonotopes [31]), `ASB07` (based on zonotopes for parametric systems [9]), and `LGG09` (based on support functions [35]). These algorithms can be combined with different *approximation models* such as forward and correction hull, adapted from [27] and [1], respectively.

SpaceEx *SpaceEx* is a tool for computing reachability of hybrid systems with complex, high-dimensional dynamics [27, 28, 26]. It can handle hybrid automata whose continuous and jump dynamics are piecewise affine with nondeterministic inputs. Its input language facilitates the construction of complex models from automata components that can be combined to networks and parameterized to construct new components. The analysis engine of SpaceEx combines explicit set representations (polyhedra), implicit set representations (support functions) and linear programming to achieve a maximum of scalability while maintaining high accuracy. It constructs an overapproximation of the reachable states in the form of template polyhedra. Template polyhedra are polyhedra whose faces are oriented according to a user-provided set of directions (template directions). A cover of the continuous trajectories is obtained by time-discretization with an adaptive time-step algorithm. The algorithm ensures that the approximation error in each template direction remains below a given value. SpaceEx is available at `http://spaceex.imag.fr`.

XSpeed The tool *XSpeed* implements algorithms for reachability analysis for continuous and hybrid systems with linear dynamics. The focus of the tool is to exploit the modern multicore architectures and enhance the performance of reachability analysis through parallel computations. XSpeed realizes two algorithms to enhance the performance of reachability analysis of purely continuous systems. The first is the parallel support function sampling algorithm and the second is the time-slicing algorithm [37, 38]. The performance of hybrid systems reachability analysis is enhanced using an adaptation of the G.J. Holzmann's parallel BFS algorithm in the SPIN model checker, called the AGJH algorithm [32]. In addition, a task parallel and an asynchronous variant of AGJH are also implemented in the tool. XSpeed is available at `http://xspeed.nitmeghalaya.in/`

3 Verification of Benchmarks

For the 2020 edition, we have decided to keep all benchmarks from our 2019 friendly competition [10] and have added two further benchmarks: a high-dimensional heat transfer problem and an electro-mechanical braking system subject to many discrete transitions.

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

- *Heat 3D benchmark* from [15]: 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.
- *Space station benchmark* from [41]: This is a purely continuous benchmark of medium size with 270 state variables and three inputs.

- *Spacecraft Rendezvous benchmark* from [19]: 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 [6, 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.
- *Building benchmark* from [41, No. 2]: A purely continuous linear system with a small number of continuous state variables; the benchmark does not only have safety properties, but also properties that should be violated to check whether the reachable sets contain certain states.
- *Platooning benchmark* from [16]: 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 [20]: 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 [**?**]: 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 [15] and is based on a 2D version originally described and evaluated in [34, 33].

All of the sides of the considered heated block are insulated, except the $x = 1$ edge, which allows for 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 rest of 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 reachability algorithm in [29] with a time step size of 0.02 for the benchmark instance HEAT01. For the higher-dimensional benchmark instances we compute the reachable set using the Krylov-subspace-based reachability algorithm in [3] using a time step size of 0.05 for HEAT02 and 0.005 for HEAT03.

Note C2E2 C2E2 uses the newly implemented algorithm with generalized star sets [23] to solve this problem. The step size is 0.02 for benchmark version HEAT01. C2E2 cannot solve benchmark version HEAT02-HEAT05 due to constraint memory.

Note HyDRA We use a time step size of 0.03 for the instance HEAT01 and a time step size of 0.014 for the instance HEAT02. Both models are verified using a recently added, more efficient implementation of support functions.

Note JuliaReach For HEAT03 and HEAT04 we use an implementation of `LGG09` that lazily computes the matrix exponential using the Lanczos algorithm [36], with chosen Krylov subspace dimension of 94 and 211 respectively.

Note SpaceEx SpaceEx computes the matrix exponential with a Padé approximation on dense matrices. It therefore does not scale well to more than ≈500 variables.

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

tool	**HEAT01**	**HEAT02**	**HEAT03**	**HEAT04**	**HEAT05**	**language**
CORA	2.19	10.1	290	–	–	MATLAB
C2E2	52.91	–	–	–	–	C++
HyDRA	0.55	175	–	–	–	C++
JuliaReach	0.064	3.2	–	–	–	Julia
SpaceEx	3.98	–	–	–	–	C++
XSpeed	2868.19	–	–	–	–	C++
	discrete-time tools					
Hylaa	25.75	533	–	–	–	Python
Hylaa-Continuous	0.67	0.82	1.48	8.98	39.09	Python
JuliaReach	0.028	1.14	146	4868	–	Julia

3.2 International Space Station Benchmark

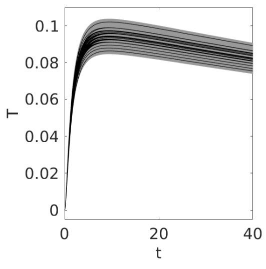
3.2.1 Model

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 [41]. In particular, the considered system is a structural model of component 1R (Russian service module), which has 270 state variables with three inputs.

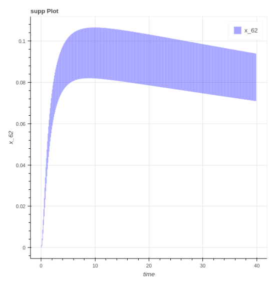
Initially, all 270 variables are in the range $[-0.0001, 0.0001]$, u_1 is in $[0, 0.1]$, u_2 is in $[0.8, 1]$, and u_3 is in $[0.9, 1]$. The time bound is 20. Discrete-time analysis for the space station benchmark should be done with a step size of 0.01. The A, B, and C matrices are available in MATLAB 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:

[2] slicot.org/objects/software/shared/bench-data/iss.zip

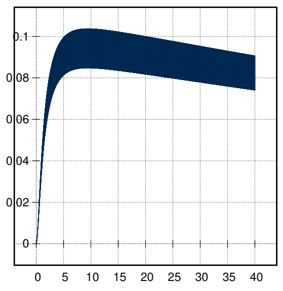
[3] cps-vo.org/node/34059



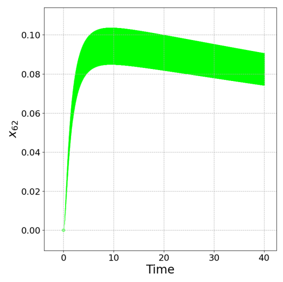
(a) CORA.



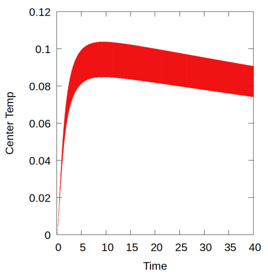
(b) C2E2.



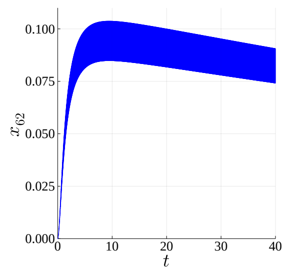
(c) HyDRA.



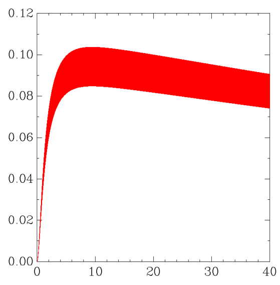
(d) Hylaa.



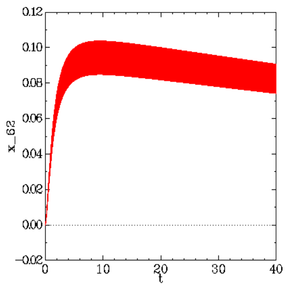
(e) Hylaa Continuous.



(f) JuliaReach.



(g) SpaceEx.



(h) XSpeed.

Figure 1: Heat3D: Reachable sets for the temperature at the center of the block over time for benchmark version HEAT01. Some tools additionally show possible trajectories.

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: $u(0) \in \mathcal{U}$, $\dot{u}(t) = 0$.

3.2.2 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.

3.2.3 Results

Results of the international space station benchmark for state y_3 over time are shown in Fig. 2 and Fig. 3. The computation times of various tools for the benchmark are listed in Tab. 2.

Note CORA CORA applies the block-decomposition algorithm [18] with step size 0.01 and zonotope order 30 for benchmark version ISSF01. For version ISSC01, a step size of 0.02 and a zonotope order of 10 is used.

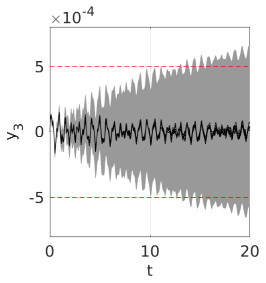
Note C2E2 C2E2 applies the algorithm with generalized star sets [23]. For benchmark version ISSC01, C2E2 uses step size 0.005. C2E2 is able to properly solve the ISS02 version of the benchmark. However, C2E2 is not able to solve the ISU02 version of the benchmark since C2E2 is computing an overapproximation of the reachable set. C2E2 is not able to solve the ISSF01 scenario since C2E2 cannot handle arbitrarily changing inputs.

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

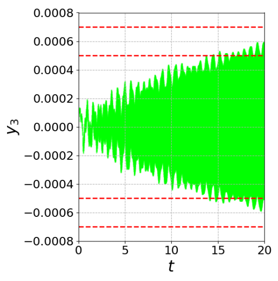
Note SpaceEx SpaceEx was run with the LGG algorithm. The sampling was chosen as 0.005 for ISSF01 and 0.05 for ISSC01. Only the two template directions $\pm y_3$ were used. Since y_3 is an algebraic variable that is a linear expression of the state variables, we replaced it in the forbidden states and the direction definition by the corresponding linear expression. To model the constant inputs in ISSC01, we introduced u_1, u_2, u_3 as state variables with $\dot{u}_1 = \dot{u}_2 = \dot{u}_3 = 0$. A custom algorithm for constant inputs could avoid such an artificial augmentation and significantly reduce the runtime for ISSC01. Note that SpaceEx treats the initial states as a general polyhedron, i.e., a linear program is solved at every time step. SpaceEx also computes the full matrix exponential, a 270×270 matrix, even though in the LGG algorithm it would suffice to compute the vector $e^{At}\ell$ for each template direction ℓ.

Since SpaceEx does not currently support the plotting of algebraic variables, we used the following trick to plot y_3 over time: we introduced a state variable z with dynamics $\dot{z} = -1000(z - y_3)$. Since the time constant for z is about two orders of magnitude below that of y_3, we expect the plots to be practically identical to a true plot of y_3.

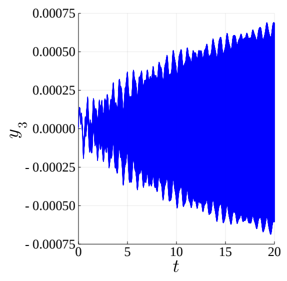
Note XSpeed XSpeed is run with BOX template directions and a sampling time of 0.01 and 0.0005 for ISSC01 and ISSF01. To model constant inputs in ISSC01, u_1, u_2, u_3 are treated as state variables with a constant dynamics of 0, similar to how SpaceEx deals with constant inputs. XSpeed computes the reachable set over the state-variables and applies linear transformation on it with the C matrix, in order to get the reachable set over the algebraic output variables, namely y_1, y_2, y_3. However, there are provisions in XSpeed to plot reachable set over both state as well as output variables. Application of linear transformation is straightforward due to the favorable properties of support functions.



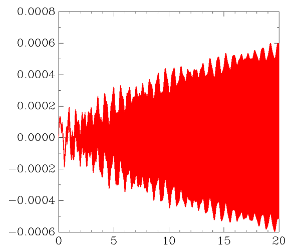
(a) CORA.

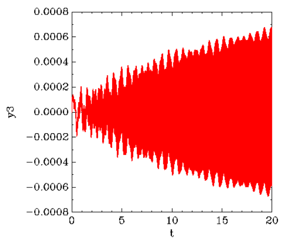


(b) Hylaa.



(c) JuliaReach.


(d) SpaceEx.

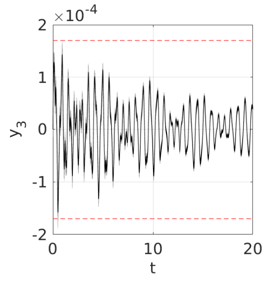


(e) XSpeed.

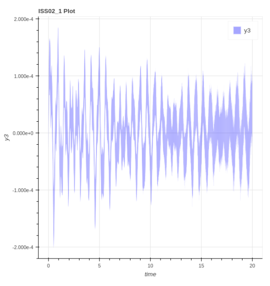
Figure 2: ISS: Reachable sets of y_3 plotted over time for the uncertain input case.

Table 2: Computation Times for the International Space Station Benchmark in [s].

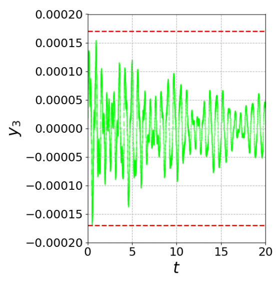
	ISSF01		ISSC01		
tool	**ISS01**	**ISU01**	**ISS02**	**ISU02**	**language**
CORA	65	39	1.56	0.13	MATLAB
C2E2	–	–	543.49	–	C++
JuliaReach	9.15	9.55	0.86	0.87	Julia
SpaceEx	47.22	47.33	28.41	28.46	C++
XSpeed	19368.2	19367.6	1009.64	1009.82	C++
	discrete-time tools				
Hylaa	401	109	60	0.89	Python
JuliaReach	9.67	8.71	0.83	0.92	Julia



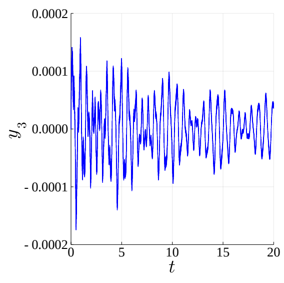
(a) CORA.



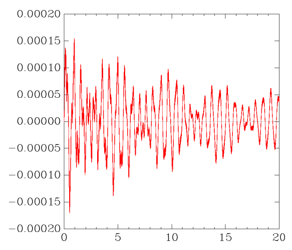
(b) C2E2.



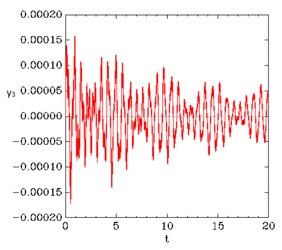
(c) Hylaa.



(d) JuliaReach.



(e) SpaceEx.



(f) XSpeed.

Figure 3: ISS: Reachable sets of y_3 plotted over time for the constant input case.

3.3 Spacecraft Rendezvous Benchmark

3.3.1 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 [19]; 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 [21]. This benchmark analyzes this linear hybrid model.

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

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].

3.3.2 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. 4.

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.

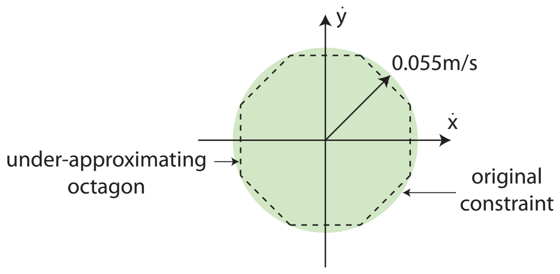


Figure 4: Under-approximation of the nonlinear velocity constraint by an octagon.

3.3.3 Results

Results of the spacecraft rendezvous benchmark for the s_x-s_y-plane are shown for the version SRNA01 in Fig. 5 and for the version SRA01 in Fig. 6. The computation times of various tools for the spacecraft rendezvous benchmark are listed in Tab. 3.

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 [30]. In order to find suitable orthogonal directions for the method in [30], 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 [40].

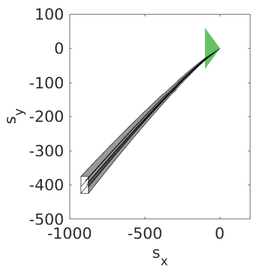
Note C2E2 C2E2 uses the newly implemented generalized-star-based algorithm from [23]. Since C2E2 cannot handle nondeterministic transitions, C2E2 can only handle benchmark instances SRNA01, SRA01, and SRA04. For benchmark SRNA01, C2E2 is using a time step of 0.1. For benchmark SRA01 and SRA04, C2E2 is is using a time step of 0.01.

Note Hylaa Hylaa was run with urgent (0-step) transitions disabled, since upon entering the approaching mode, the state may be on the boundary of the line-of-sight cone (strict inequalities are not allowed by linear programming tools and therefore not allowed in Hylaa either).

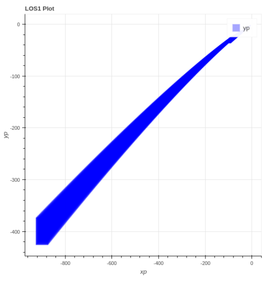
Note JuliaReach We used `BFFPSV18` and chose a one-block partition and hyperrectangular reach-sets with a step size in dense time of 0.04 for instances SRA01-SRA03. We handled discrete transitions by computing the intersection with invariants and guards lazily before their overapproximation with a hyperrectangle. For the instance SRA04, we use a clustering strategy of order 16 and step size of 0.01. Unsat instances are ran with step size 0.04.

Note SpaceEx SpaceEx was run with the LGG algorithm, box directions, and a flowpipe tolerance of 0.2.

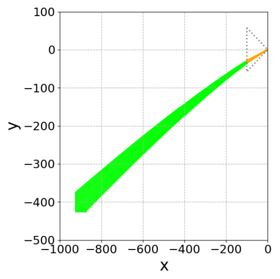
Note XSpeed XSpeed on SRNA was run with box directions, time step of 0.2 and set aggregation at discrete jumps turned off. Not aggregating sets slows down the performance, but allows to prove the LoS specification.



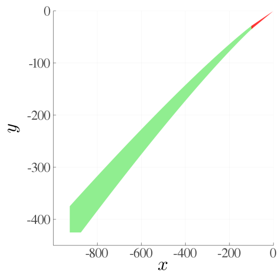
(a) CORA.



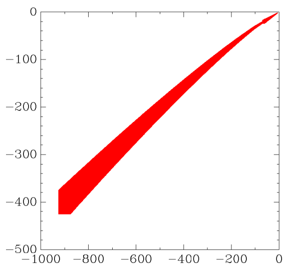
(b) C2E2.



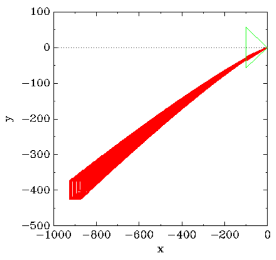
(c) Hylaa.



(d) JuliaReach.



(e) SpaceEx.

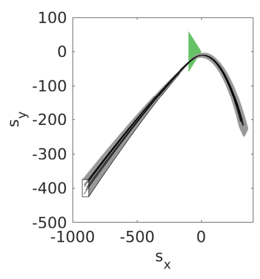


(f) XSpeed.

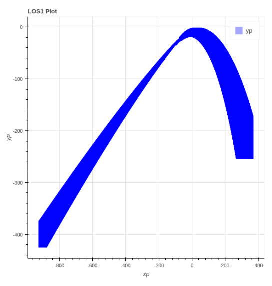
Figure 5: Reachable sets for the spacecraft rendezvous benchmark in the s_x-s_y-plane for the benchmark variant without maneuver abortion (SRNA01).

Table 3: Computation time [s] for the spacecraft rendezvous benchmarks (**SR***) for specification **SR02**.

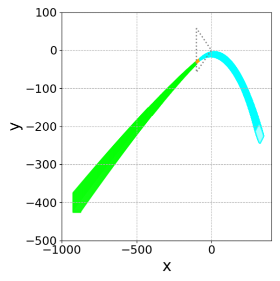
tool	**NA01**	**A01**	**A02**	**A03**	**A04**	**A05**	**A06**	**A07**	**A08**	**U01**	**U02**
CORA	9.9	2.6	2.9	4.3	7.7	11.1	11.2	61.7	165	8.7	158
C2E2	65.05	31.40	–	–	37.65	–	–	–	–	–	–
JuliaReach	0.44	0.45	0.53	0.55	34.4	–	–	–	–	4.11	10.4
SpaceEx	0.21	0.22	–	–	–	–	–	–	–	0.30	2.19
XSpeed	32.17	317.31	–	–	–	–	–	–	–	–	–
	discrete-time tools										
Hylaa	9.0	2.3	2.8	15	30	48	53	15	180	26	489
JuliaReach	0.13	0.13	0.15	0.23	0.98	–	–	–	–	2.11	3.78



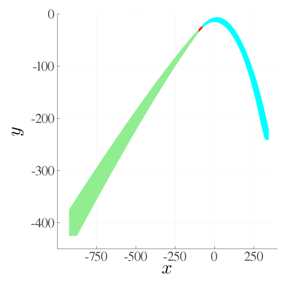
(a) CORA.



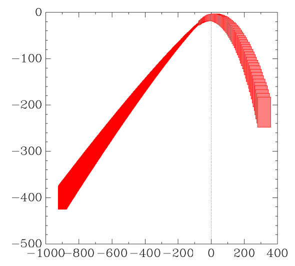
(b) C2E2.

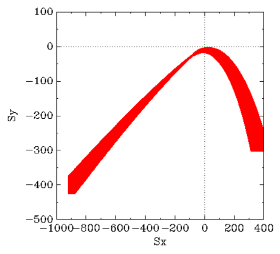


(c) Hylaa.



(d) JuliaReach.


(e) SpaceEx.



(f) XSpeed.

Figure 6: 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])

3.4 Powertrain with Backlash

3.4.1 Model

The powertrain benchmark is an extensible benchmark for hybrid systems with linear continuous dynamics taken from [6, Sec. 6] and [11, 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 different components of the powertrain, e.g., gears, differential, and clutch, as illustrated in Fig. 7. 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]. Discrete-time analysis for the powertrain system should be done with a step size of 0.0005 (5.0E-4). The set of initial states is

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

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

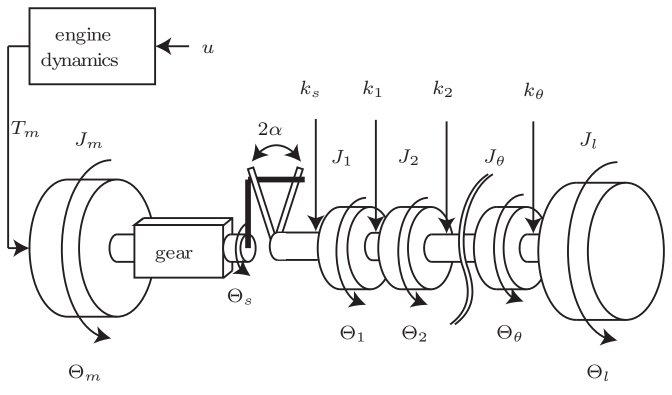


Figure 7: Powertrain model.

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

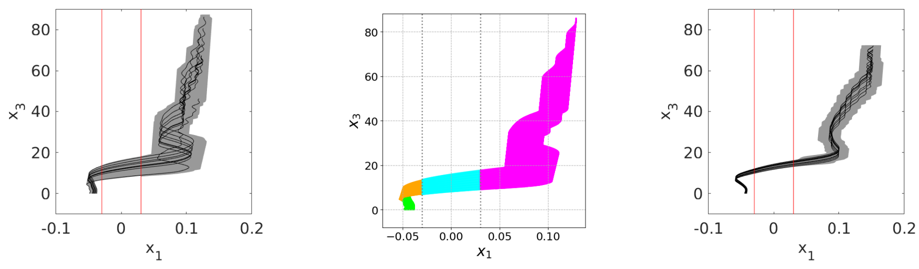
DTN06 $\theta = 22$, no change of $\mathcal{X}_0$.

3.4.3 Results

Results of the powertrain benchmark in the x_1-x_3-plane are shown in Fig. 8. The computation times of various tools for the powertrain benchmark are listed in Tab. 4.

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 [30], and principal component analysis is used to find suitable directions for the enclosure of the guard intersections.

Note Hylaa Hylaa was run with aggregation turned off. Convex hull aggregation worked, but was slower. The current deaggregation approach in Hylaa did not work well, as the overapproximation error seemed to allow states to remain in the dead zone indefinitely, so deaggregation was never triggered.



(a) CORA (DTN03). (b) Hylaa (DTN03). (c) CORA (DTN05).

Figure 8: Reachable sets in the x_1-x_3-plane.

Table 4: Computation Times for the Powertrain Benchmark in [s].

tool	**DTN01**	**DTN02**	**DTN03**	**DTN04**	**DTN05**	**DTN06**	**language**
CORA	4.03	4.14	4.17	39.1	74.5	205	MATLAB
	discrete-time tools						
Hylaa	6.45	15.2	49.6	12.6	43.3	154	Python

3.5 Building Benchmark

3.5.1 Model

This benchmark is quite straightforward: The system is described by $\dot{x}(t) = Ax(t) + Bu(t)$, $u(t) \in \mathcal{U}$, $y(t) = Cx(t)$, where A, B, C are provided on the ARCH website[6]. The initial set and the uncertain input $\mathcal{U}$ are provided in [41, Tab. 2.2]. Discrete-time analysis for the building system should use a step size of 0.01. There are two versions of this benchmark:

[6]cps-vo.org/node/34059

BLDF01 The inputs can change arbitrarily over time: $\forall t : u(t) \in \mathcal{U}$.

BLDC01 (constant inputs) The inputs are uncertain only in their initial value, and constant over time: $u(0) \in \mathcal{U}$, $\dot{u}(t) = 0$. The purpose of this model instance is to accommodate tools that cannot handle time-varying inputs.

3.5.2 Specifications

The verification goal is to check whether the displacement y_1 of the top floor of the building remains below a given bound. In addition to the safety specification from the original benchmark, there are two UNSAT instances that serve as sanity checks 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 indicates that the over-approximation is indeed conservative.

BDS01 Bounded time, safe property: For all $t \in [0, 20]$, $y_1(t) \leq 5.1 \cdot 10^{-3}$. This property is assumed to be satisfied.

BDU01 Bounded time, unsafe property: For all $t \in [0, 20]$, $y_1(t) \leq 4 \cdot 10^{-3}$. This property is assumed to be violated. Property BDU01 serves as a sanity check. A tool should be run with the same accuracy settings on BLDF01-BDS01 and BLDF01-BDU01, returning UNSAT on the former and SAT on the latter.

BDU02 Bounded time, unsafe property: The forbidden states are $\{y_1(t) \leq -0.78 \cdot 10^{-3} \wedge t = 20\}$. This property is assumed to be violated for BLDF01 and satisfied for BLDC01. Property BDU02 serves as a sanity check to confirm that time-varying inputs are taken into account. A tool should be run with the same accuracy settings on BLDF01-BDU02 and BLDC01-BDU02, returning UNSAT on the former and SAT on the latter.

3.5.3 Results

Results of the building benchmark for state x_{25} over time are shown in Fig. 9-12. The computation times of various tools for the building benchmark are listed in Tab. 5.

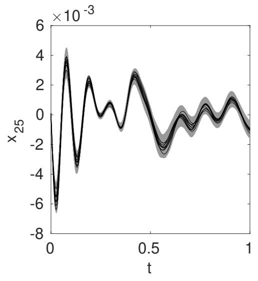
Note CORA Since the dynamics of this example is dominated by the input after one second, we use the step size 0.002 for $t \in [0, 1]$ and the step size 0.01 for $t \in [1, 20]$. The zonotope order is chosen as 100.

Note C2E2 For the BLDC version of the benchmark, C2E2 uses the newly implemented algorithm in [23] using generalized star sets. The time step size used is 0.005. C2E2 is not able to handle BLDF01 because arbitrarily changing bounded inputs are not yet supported.

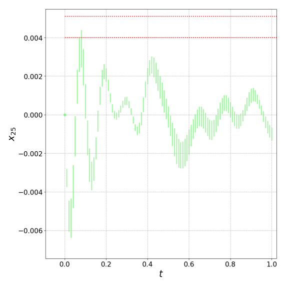
Note HyDRA We use a step size of 0.004 and support functions with an octagonal template as a state set representation. As HyDRA cannot handle uncertain inputs, we have added another variable to the model accounting for the uncertain input.

Note JuliaReach Since the safety property only involves one state variable, we use the LGG09 algorithm. We use the following step sizes (in dense time): 0.004 for BLDF01 and 0.006 for BLDC01.

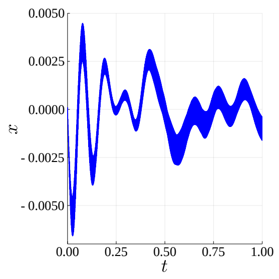
Note SpaceEx The accuracy of SpaceEx was set to the largest value possible that satisfies the specification, here $\varepsilon = 0.01$. This means the tool can exploit any margin to reduce the number of computations and/or the number of convex sets in the reach set. The resulting, intentional lack of accuracy shows in the plot.



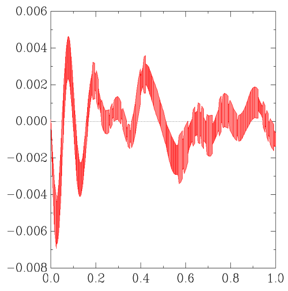
(a) CORA.



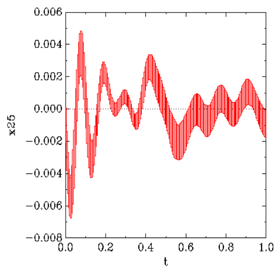
(b) Hylaa.



(c) JuliaReach.



(d) SpaceEx.



(e) XSpeed.

Figure 9: Building (BLDF01): Reachable sets of x_{25} plotted over time up to time 1. Some tools additionally show possible trajectories.

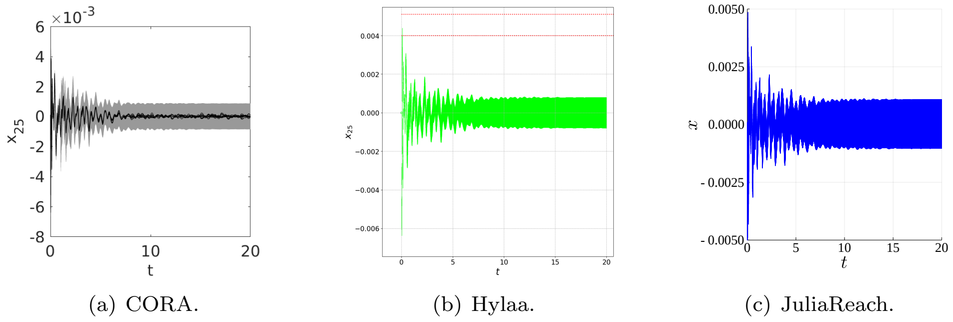
3.6 Platooning Benchmark

3.6.1 Model

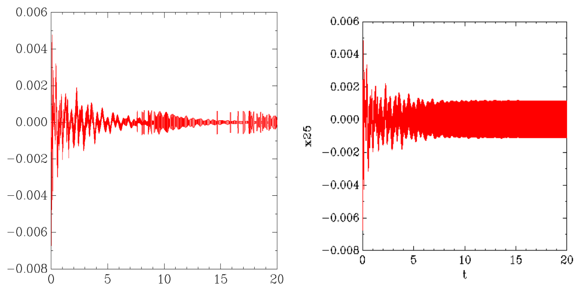
The platooning benchmark considers a platoon of three vehicles following each other. This benchmark considers loss of communication between vehicles. The initial discrete state is q_c. Three scenarios are considered for the loss of communication:

PLAA01 (arbitrary loss) The loss of communication can occur at any time, see Fig. 13(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. 13(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.



(a) CORA. (b) Hylaa. (c) JuliaReach.



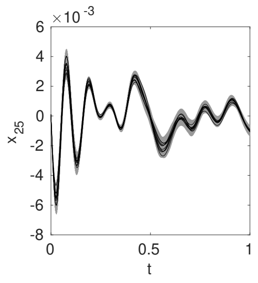
(d) SpaceEx. (e) XSpeed.

Figure 10: Building (BLDF01): Reachable sets of x_{25} plotted over time up to time 20. Some tools additionally show possible trajectories.

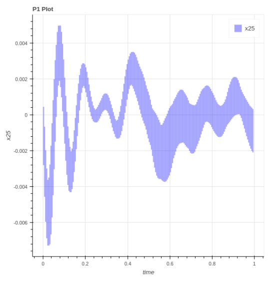
Table 5: Computation Times for the Building Benchmark in [s].

tool	**BLDC01 BDS01**	**BLDF01 BDS01**	**language**
CORA	2.65	3.1	MATLAB
C2E2	21.62	–	C++
HyDRA	0.57	–	C++
JuliaReach	0.020	0.013	Julia
SpaceEx	1.55	1.84	C++
XSpeed	403.07	386.1	C++
discrete-time tools			
Hylaa	1.4	11.7	Python
JuliaReach	0.0059	0.0035	Julia

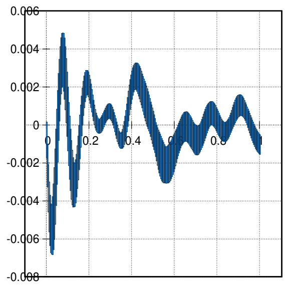
PLAD01: $c_1 = c_2 = 5$.



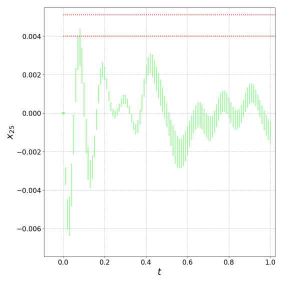
(a) CORA.



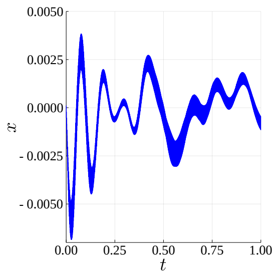
(b) C2E2.



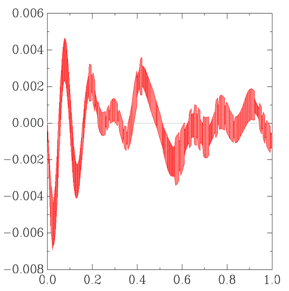
(c) HyDRA.



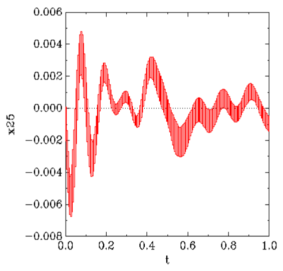
(d) Hylaa.



(e) JuliaReach.



(f) SpaceEx.



(g) XSpeed

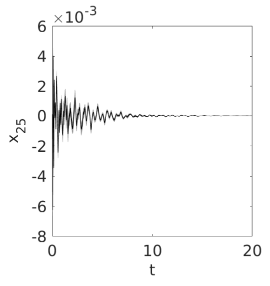
Figure 11: Building (BLDC01): Reachable sets of x_{25} plotted over time up to time 1. Some tools additionally show possible trajectories.

PLANxy (loss at nondeterministic times) The loss of communication occurs at any time $t \in [t_b, t_c]$ in Fig. 13(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.

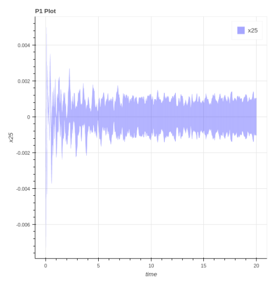
PLAN01: $t_b = 10$, $t_c = 20$, $t_r = 20$.

The models are available in SpaceEx, KeYmaera, and MATLAB/Simulink format on the ARCH website[7]. Discrete-time analysis for the platoon system should use a step size of 0.1.

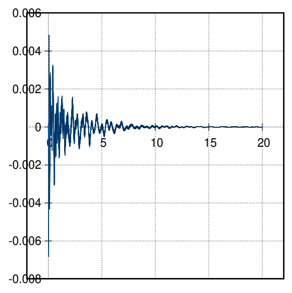
[7]cps-vo.org/node/15096



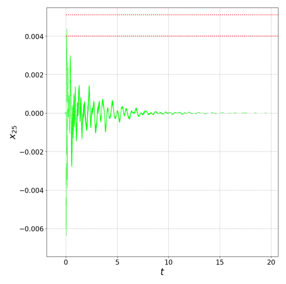
(a) CORA.



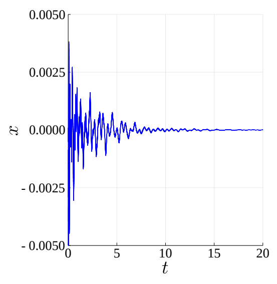
(b) C2E2.



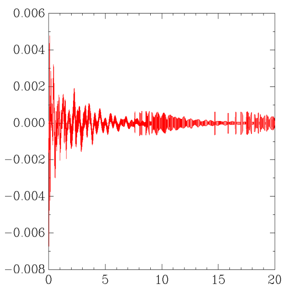
(c) HyDRA.



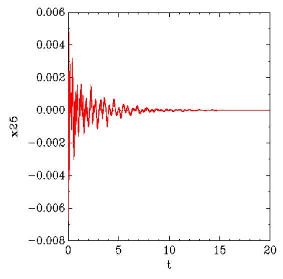
(d) Hylaa.



(e) JuliaReach.



(f) SpaceEx.



(g) XSpeed.

Figure 12: Building (BLDC01): Reachable sets of x_{25} plotted over time up to time 20. Some tools additionally show possible trajectories.

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

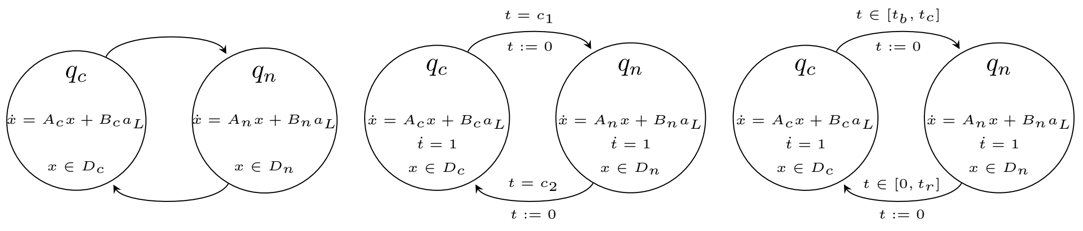
3.6.2 Specifications

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: $d_{min} = 42$.



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

Figure 13: Three options adapted from the original benchmark proposal [16]. 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.

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

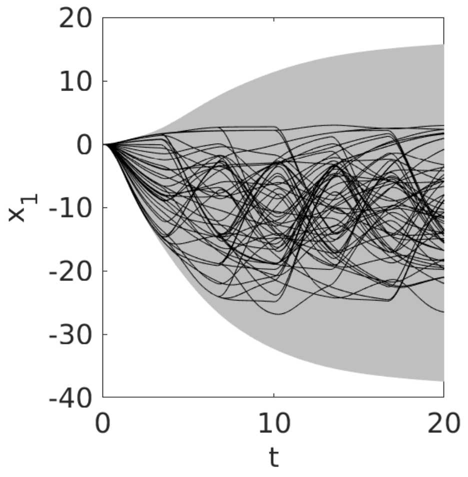
Results of the platoon benchmark for state x_1 over time are shown in Fig. 14-16. The computation times of various tools for the platoon benchmark are listed in Tab. 6.

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* [7, 8] to rewrite the hybrid automaton as a purely continuous system with uncertain parameters.

Note HyDRA As HyDRA cannot handle uncertain time-varying inputs, we use an instance of the platoon benchmark in which inputs are constant (similar to the building benchmark). We use a step size of 0.25 and support functions with an octagonal template for instance DBND30 and a box-shaped template for instance DBND42.

Note JuliaReach For PLAD01-BND42, the step size used (in dense time) is 0.01. For PLAD01-BND30, the step size used (in dense time) is 0.03. For the BND30 instances, intersections with the guard are taken lazily using `LGG09` with an octagonal template.

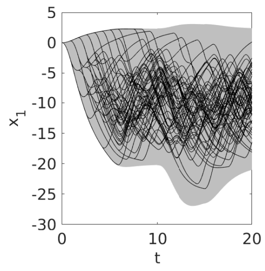


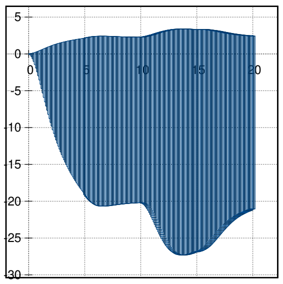
(a) CORA.

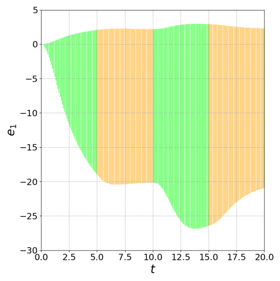
Figure 14: PLAA01: Reachable sets of x_1 plotted over time. CORA additionally shows possible trajectories.

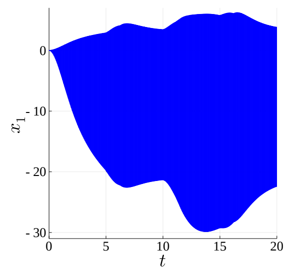
Table 6: Computation Times for the Platoon Benchmark in [s].

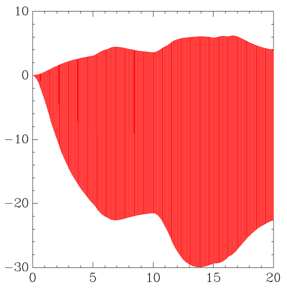
tool	**PLAA01 BND50**	**PLAA01 BND42**	**PLAD01 BND42**	**PLAD01 BND30**	**PLAN01 UNB50**	**language**
CORA	11.3	38.2	1.07	3.02	102	MATLAB
HyDRA	–	–	0.41	18.81	–	C++
JuliaReach	–	–	0.13	40.6	–	Julia
SpaceEx	–	–	0.37	9.73	109.63	C++
XSpeed	–	–	5.36	1588	37.33	C++
	discrete-time tools					
Hylaa	–	–	0.97	0.97	–	Python
JuliaReach	–	–	0.94	9.7	–	Julia


(a) CORA (BND30).


(b) HyDRA (BND30).


(c) Hylaa (BND30).


(d) JuliaReach (BND30).


(e) SpaceEx (BND30).

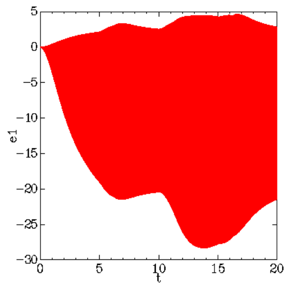
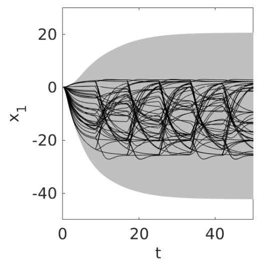
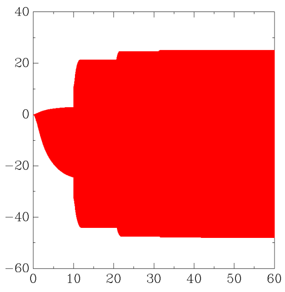

(f) XSpeed (BND30).

Figure 15: PLAD01: Reachable sets of x_1 plotted over time. Some tools additionally show possible trajectories.


(a) CORA (UNB50).


(b) SpaceEx (UNB50).

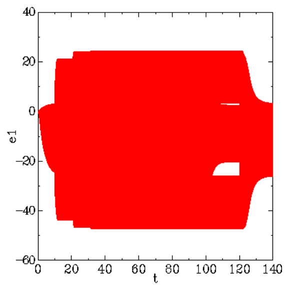

(c) XSpeed (UNB50).

Figure 16: 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[8] and as a Simulink model[9]. 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 [20]. 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 $\mathcal{X}_0 = 0 \times 0 \times [-0.01675, -0.01665] \times [0.00285, 0.00315] \times 0$.

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. 17. 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 [30]. In order to find suitable orthogonal directions for the method in [30], 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.

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

Note C2E2 C2E2 uses the discrepancy-function-based algorithm in [25]. The time step used for both instances of this benchmark is $1e-6$.

Note Hylaa Hylaa used resets to project the system state exactly onto the guard when transitioning from the *free* mode.

Note XSpeed To prove safety, XSpeed uses convex-hull set aggregation in discrete jump computation, octagonal template directions, and a time-step of 0.0001.

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

tool	GRBX01-MES01	GRBX02-MES01	language
CORA	0.62	0.76	MATLAB
C2E2	259.99	256.80	C++
SpaceEx	0.07	0.07	C++
XSpeed	3327	3280.15	C++
	discrete-time tools		
Hylaa	5.5	5.6	Python

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 [?]. 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[10]. Here, we consider a linearized version with parameters.

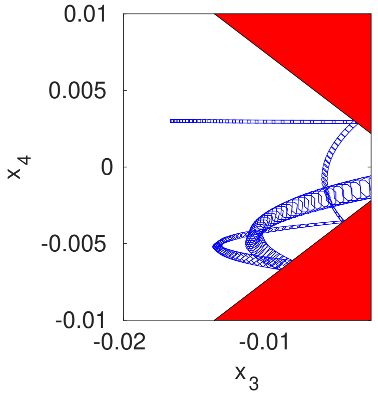
The model is a hybrid automaton (see Fig. 18) with four state variables (the motor current I, the brake position x, and two auxiliary linearization variables) and a clock variable T. The automaton consists of a single mode and a self-loop transition. The transition is time-triggered, i.e., it only depends on the value of the clock variable.

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

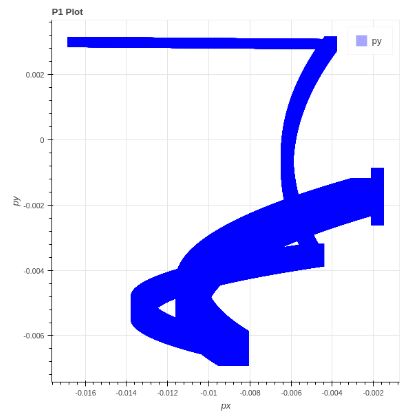
3.8.2 Specification

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:

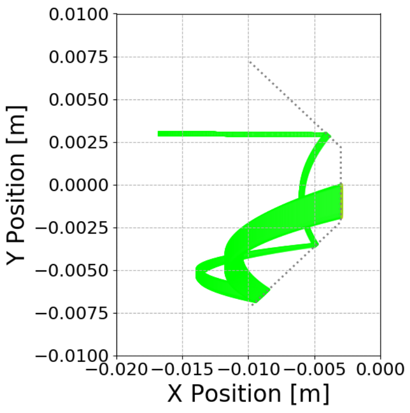
[10]cps-vo.org/node/20289



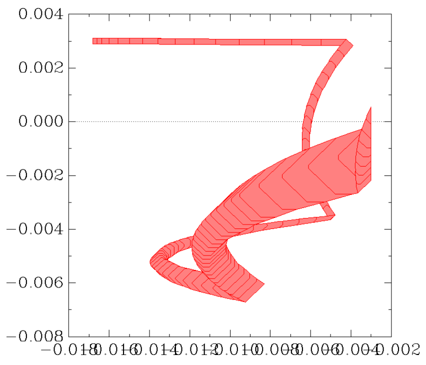
(a) CORA.



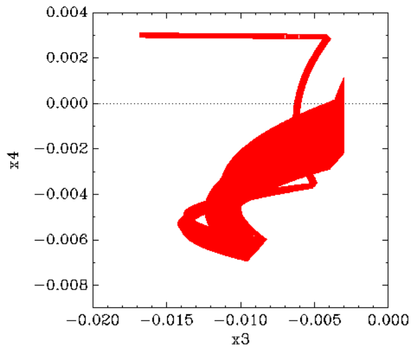
(b) C2E2.



(c) Hylaa.



(d) SpaceEx.



(e) XSpeed.

Figure 17: Gearbox (GRBX01): Reachable sets of x_3 and x_4.

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

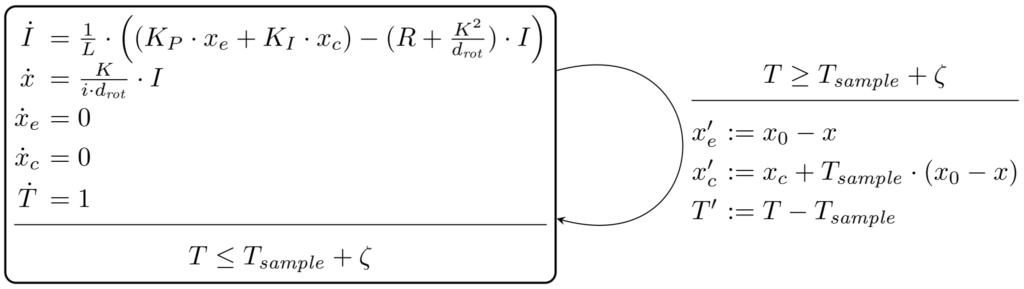

$\dot{I} = \frac{1}{L} \cdot \Big((K_P \cdot x_e + K_I \cdot x_c) - (R + \frac{K^2}{d_{rot}}) \cdot I \Big)$
$\dot{x} = \frac{K}{i \cdot d_{rot}} \cdot I$
$\dot{x}_e = 0$
$\dot{x}_c = 0$
$\dot{T} = 1$
$T \leq T_{sample} + \zeta$
$T \geq T_{sample} + \zeta$
$x_e' := x_0 - x$
$x_c' := x_c + T_{sample}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

Figure 18: Hybrid automaton of the electro-mechanical brake with periodic discrete-time PI controller and sampling jitter.

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).

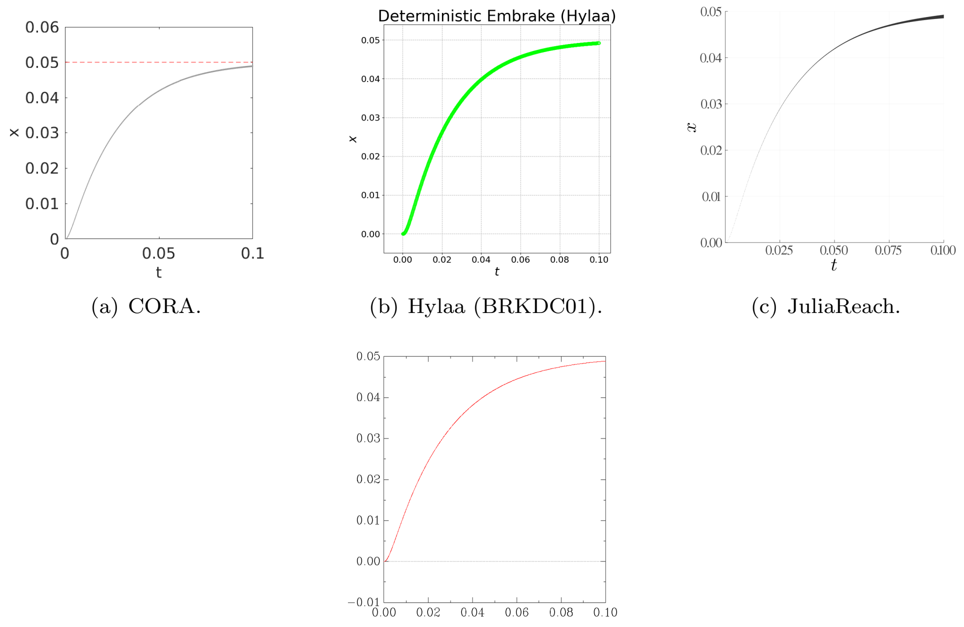
3.8.3 Results

Results of the benchmark are shown in Fig. 19. 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 [40].

Note JuliaReach For the BRKDC01 and BRKNC01 scenario, we use the `GLGM06` algorithm with a fixed step size of 10^{-8} and a forward approximation model. For the BRKNP01 scenario, we use the `ASB07` algorithm with a step size of 10^{-8} and an order 10 correction-hull-approximation model. In all scenarios we use a maximum zonotope order of one. The discrete-time instances use the same step sizes as the dense-time ones. We use a custom analysis for dealing with the time-triggered transition efficiently, considering intersections with the guard separately from the flowpipe computation. The largest time horizon for which $x < x_0$ holds for BRKNP01 is 0.0823s and 0.0825s for dense and continuous time, respectively.

Note SpaceEx We use the STC algorithm, which here is significantly faster than the LGG algorithm despite using sophisticated algorithms for containment checking, convexification, and redundancy reduction of polyhedra. Since SpaceEx is a model checker, it checks after each jump whether the successor states have already been visited. In this benchmark, all states are in the same location. At the k-th jump, this leads to a pairwise containment check with all $k - 1$ previous states. This consumes about 90% of the runtime. For the non-deterministic instance BRKNC01, the required precision means that polyhedra are much more complex than in the deterministic instance (more faces). The containment checking therefore leads to an excessive runtime.



(a) CORA. (b) Hylaa (BRKDC01). (c) JuliaReach.

(d) SpaceEx.

Figure 19: 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	4.84	427	496	MATLAB
HyDRA	–	–	–	C++
JuliaReach	0.82	0.99	12.2	Julia
SpaceEx	19.22	–	–	C++
XSpeed	–	–	–	C++
	discrete-time tools			
Hylaa	230	–	–	Python
JuliaReach	0.65	0.97	12.0	Julia

4 Conclusion and Outlook

This report presents the results of the fourth friendly competition for the formal verification of continuous and hybrid systems with linear continuous dynamics as part of the ARCH'20 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. Also, one can now execute the Dockerfile of all tools on gitlab.com/goranf/ARCH-COMP using the command `measure_all`. Information about the competition in 2021 will be announced on the ARCH website.

5 Acknowledgments

The authors gratefully acknowledge financial support by the European Commission project justITSELF under grant number 817629, by the Austrian Science Fund (FWF) under grant Z211-N23 (Wittgenstein Award), by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411, and by the Science and Engineering Research Board (SERB) project with file number IMP/2018/000523. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-19-1-0288. Any opinions, finding, 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.

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*, pages 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 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.

[5] 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*, pages 145–173, 2018.

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

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

[8] 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.

[9] 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*, pages 726–732, 2007.

[10] Matthias Althoff, Stanley Bak, Marcelo Forets, Goran Frehse, Niklas Kochdumper, Rajarshi Ray, Christian Schilling, and Stefan Schupp. ARCH-COMP19 category report: Continuous and hybrid systems with linear continuous dynamics. In Goran Frehse and Matthias Althoff, editors, *Proc. of the 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 61 of *EPiC Series in Computing*, pages 14–40, 2019.

[11] 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*, pages 133–150, 2017.

[12] S. Bak and P. S. Duggirala. Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 173–178. ACM, 2017.

[13] S. Bak and P. S. Duggirala. Rigorous simulation-based analysis of linear hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 555–572. Springer, 2017.

[14] S. Bak and P. S. Duggirala. Simulation-equivalent reachability of large linear systems with inputs. In *International Conference on Computer Aided Verification*, pages 401–420. Springer, 2017.

[15] 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.

[16] 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*, pages 37–42, 2015.

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

[18] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling. Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices. In *HSCC*, pages 41–50, 2018.

[19] 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*, pages 20–32, 2017.

[20] 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*, pages 9–18, 2015.

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

[22] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2E2: A verification tool for stateflow models. In *TACAS*, 2015.

[23] Parasara Sridhar Duggirala and Mahesh Viswanathan. Parsimonious, simulation based verification of linear systems. In *Computer Aided Verification*, pages 477–494. Springer International Publishing, 2016.

[24] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala. Automatic reachability analysis for nonlinear hybrid models with C2E2. In *Computer Aided Verification*, pages 531–538, Cham, 2016. Springer International Publishing.

[25] Chuchu Fan and Sayan Mitra. Bounded verification with on-the-fly discrepancy computation. In *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015, Proceedings*, pages 446–463, 2015.

[26] G. Frehse. Reachability of hybrid systems in space-time. In Alain Girault and Nan Guan, editors, *Proc. Int. Conf. Embedded Software, EMSOFT, Amsterdam, Netherlands, October 4-9, 2015*, pages 41–50. IEEE, 2015.

[27] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *Proc. of the 23rd International Conference on Computer Aided Verification*, LNCS 6806, pages 379–395. Springer, 2011.

[28] G. Frehse, R. Kateja, and C. Le Guernic. Flowpipe approximation and clustering in space-time. In C. Belta and F. Ivancic, editors, *Proc. Int. Conf. Hybrid systems: computation and control, HSCC, April 8-11, 2013, Philadelphia, PA, USA*, pages 203–212. ACM, 2013.

[29] A. Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, LNCS 3414, pages 291–305. Springer, 2005.

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

[31] 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.

[32] A. Gurung, A. Deka, E. Bartocci, S. Bogomolov, R. Grosu, and R. Ray. Parallel reachability analysis for hybrid systems. In *ACM/IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE*, pages 12–22. IEEE, 2016.

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

[34] 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.

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

[36] 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. Exported from https://app.dimensions.ai on 2019/05/05.

[37] R. Ray and A. Gurung. Poster: Parallel state space exploration of linear systems with inputs using xspeed. In *Proc. of HSCC'15*, pages 285–286. ACM, 2015.

[38] R. Ray, A. Gurung, B. Das, E. Bartocci, S. Bogomolov, and R. Grosu. XSpeed: Accelerating reachability analysis on multi-core processors. In *Proc. of HVC 2015*, volume 9434 of *LNCS*, pages 3–18, 2015.

[39] S. Schupp, E. Abraham, I. Ben Makhlouf, and S. Kowalewski. HyPro: A C++ library for state set representations for hybrid systems reachability analysis. In *Proc. NFM'17*, volume 10227 of *LNCS*, pages 288–294. Springer, 2017.

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

[41] 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*, pages 60–67, 2017.