EPiC Series in Computing

Volume 54, 2018, Pages 23–52

ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems

ARCH-COMP18 Category Report: Continuous and Hybrid Systems with Linear Continuous Dynamics

Matthias Althoff[1], Stanley Bak[2], Xin Chen[3], Chuchu Fan[9], Marcelo Forets[7], Goran Frehse[4], Niklas Kochdumper[1], Yangge Li[9], Sayan Mitra[9], Rajarshi Ray[5], Christian Schilling[8], and Stefan Schupp[6]

[1] Technische Universität München, Department of Informatics, Munich, Germany
althoff@in.tum.de
[2] Safe Sky Analytics, Manlius, NY, United States
stanleybak@gmail.com
[3] University of Dayton, Dayton, OH, United States
xchen4@udayton.edu
[4] Univ. Grenoble Alpes, Grenoble, France
goran.frehse@imag.fr
[5] National Institute of Technology Meghalaya, Shillong, India.
rajarshi.ray@nitm.ac.in
[6] RWTH Aachen University, Theory of hybrid systems, Aachen, Germany
stefan.schupp@cs.rwth-aachen.de
[7] UTEC Universidad Tecnológica, Uruguay
marcelo.forets@utec.edu.uy
[8] University of Freiburg, Freiburg, Germany
schillic@informatik.uni-freiburg.de
[9] University of Illinois at Urbana-Champaign, Champaign, IL, United States
{mitras,cfan10,li213}@illinois.edu

Abstract

This report presents the results of a friendly competition for formal verification of continuous and hybrid systems with linear continuous dynamics. The friendly competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2018. In its second edition, 9 tools have been applied to solve six different benchmark problems in the category for linear continuous dynamics (in alphabetical order): CORA, CORA/SX, C2E2, Flow*, 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 probably provide the most complete assessment 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. The obtained results have been verified by an independent repeatability evaluation. To establish further trustworthiness of the results, the code with which the results have been obtained is publicly available at gitlab.com/goranf/ARCH-COMP.

This report summarizes results obtained in the 2018 friendly competition of the ARCH workshop[1] for verifying hybrid 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 $u \in \mathbb{R}^m$. Participating tools are summarized in Sec. 2. The results of our selected benchmark problems are shown in Sec. 3 and are obtained on the tool developers' own machines. Thus, one has to factor in the computational power of the processors used, summarized in Appendix A, as well as the efficiency of the programming language of the tools.

The goal of the friendly competition is not to rank the results, but rather to present the landscape of existing solutions in a breadth that is not possible with scientific publications in classical venues. Such publications would typically require the presentation of novel techniques, while this report showcases the current state-of-the-art tools. For all results reported by each participant, we have run an independent repeatability evaluation.

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 for 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.

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) [1, 3, 4] 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

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

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 http://www6.in.tum.de/Main/SoftwareCORA.

CORA/SX CORA/SX is a port of the basic zonotope reachability algorithm from the CORA MATLAB toolbox to SpaceEx. It varies slightly in that some matrix computations (which approximate the input over one time step) use SpaceEx code instead of an over-approximation that is based on intervals.

C2E2 C2E2 (Compare-Execute-Check-Engine) [21, 22] is a tool for verifying bounded-time invariant properties for hybrid system with both linear or nonlinear dynamics, and discrete transitions with guards and resets. The tool implements a *simulation-based approach* for over-approximating the reachable states. The input hybrid automata and the unsafe set has to be represented in an XML format. The new version of C2E2 used for these experiments (to be released in Fall 2018) comes with a model editor that can compose hybrid automata and a built-in plotter. C2E2 and related publications are available from https://publish.illinois.edu/c2e2-tool/.

Flow* The tool Flow* [18, 17] computes *Taylor model (TM) flowpipes* as overapproximations for continuous and hybrid system reachable sets. For systems defined by Linear Time-Invariant (LTI) and Linear Time-Varying (LTV) ODEs which could be uncertain, Flow* computes *symbolic flowpipes* which are essentially TM overapproximations for the exact flow mappings from initial sets to the reachable sets in different time intervals. The overapproximation error is only proportional to δ^{k+1} where δ is the time stepsize and k is the TM order in use. Unlike convex set representations, symbolic flowpipes are usually more time-costly to obtain, however, they are only ODE related and can be directly reused in a safety verification task, i.e., with a different initial set or unsafe condition. Besides, symbolic flowpipes can be used in generating relational abstractions [30, 19] and real-time monitoring [20] for dynamical systems. In the current version, Flow* simply treats all real numbers as intervals in order to take roundoff errors into account in all computational tasks. However it sometimes leads to serious issues in numerical stability especially in the operations on large-scale matrices. This year, we are going to release the first version of Flow* API such that the tool can be used in a more flexible way. For example, various integration, intersection and aggregation algorithms are exposed to the users, the roundoff errors produced in a function may be ignored when the inputs are rational numbers. In the future, we also plan to design more sophisticated methods to track roundoff errors in matrix computation. Flow* is available at flowstar.org.

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 [31] 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 state 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 [9, 10] computes the *simulation-equivalent* reachable set of states for a hybrid system with linear ODEs. That is, for a given model, Hylaa can compute all the states reached by any fixed-step simulation. This is a bit different than full reachability as it does not reason between time steps (it checks safety at discrete times), and furthermore time-varying inputs are considered to be constant between time steps (not varying at any point in time) [11]. If an unsafe state is reachable, however, Hylaa can produce a counter-example trace with an initial point and set of inputs to apply at each time step in order to reach an unsafe state. Hylaa uses numerical simulations to compute the matrix exponential, and LP solving to check for intersections with guards and unsafe states. Hylaa is a Python-based tool (with core computational components being libraries written in other languages), which can produce live plots during computation, as well as images and video files of projections of the reachable set. Hylaa's website is `http://stanleybak.com/hylaa`. The version of Hylaa used in this year's competition is available at `https://github.com/stanleybak/hylaa/releases/tag/v1.1`.

This year we also developed a version of Hylaa tailored to continuous affine systems (without discrete switches) [12]. This version, which we refer to as Hylaa-Continuous, uses explicit initial and output spaces to reduce the number of simulations needed to compute linear projections of the matrix exponential. It also supports Krylov subspace methods when systems are extremely large, but this feature was not used in this year's competition as the benchmarks are of modest size. The implementation used is available in the continuous branch of the Hylaa repository, `https://github.com/stanleybak/hylaa/releases/tag/continuous-May18`.

JuliaReach *JuliaReach* is a software framework for reachability computations of dynamical systems, available at `http://juliareach.org`. It is written in Julia, a modern high-level language for scientific computing. Currently *JuliaReach* can handle continuous affine systems. The reachability algorithm uses a block decomposition technique presented in [14]. Here we partition the state space, project the initial states to subspaces, and propagate these low-dimensional sets in time. This allows us to perform otherwise expensive set operations in low dimensions. Furthermore, if the output does not depend on all dimensions, we can effectively skip the reach set computation for the respective dimensions. In the evaluation we used two-dimensional blocks, for which our implementation supports epsilon-close approximation; for box approximation we can handle arbitrary partitions. For the set computations we use the *LazySets* library, which is also part of the *JuliaReach* framework. *LazySets* exploits the principle of lazy (on-demand) evaluation and uses support functions to represent lazy sets. *JuliaReach* also comes with *SX*, a parser for SX (SpaceEx format) model files. For next year we plan to add support for hybrid dynamics, which will require a careful balance between low- and high-dimensional computations and adaptive choice of the partition.

SpaceEx *SpaceEx* is a tool for computing reachability of hybrid systems with complex, high-dimensional dynamics [24, 25, 23]. It can handle hybrid automata whose continuous and jump dynamics are piecewise affine with nondeterministic inputs. Nondeterministic inputs are particularly useful for modeling the approximation error when nonlinear systems are brought to piecewise affine form. SpaceEx comes with a web-based graphical user interface and a graphical model editor. 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 [28, 29]. 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 [27]. 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 2018 edition, we have decided to keep all benchmarks from our 2017 friendly competition [2], but solve them with our updated tools and tools that did not participate in 2017. In addition, we have added three new benchmarks for 2018: a model of the international space station, a spacecraft rendezvous, and an automotive powertrain. We first discuss special features of each benchmark, different treatment of uncertain inputs, and different paths to successfully verifying a benchmark. Afterwards, the verification of each benchmark is presented in detail.

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

- *Space station benchmark* from [32]: This is a purely continuous benchmark with 270 state variables and three inputs. This year, it is the benchmark with the largest amount of continuous state variables (in 2017, the building benchmark with 48 state variables was the benchmark with the most continuous state variables).

- *Spacecraft benchmark* from [15]: This benchmark has a 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 [5, 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 [32, No. 2]: A purely continuous linear system with a medium number of continuous state variables; the benchmark does not only have safety properties, but also ones that should be violated to check whether the reachable sets contain certain states.
- *Platooning benchmark* from [13]: 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 [16]: 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.

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

1. Fixed inputs, where $u(t)$ is precisely known. If in addition, $u(t) = \texttt{const}$, the linear system becomes an affine system $\dot{x}(t) = Ax(t) + b$. For instance, the gearbox benchmark has affine dynamics.
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.

3.1 International Space Station Benchmark

3.1.1 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 [32]. In particular, the considered system is a structural model of component 1R (Russian service module), which has 270 state variables with three inputs.

The specification deals with the possible range for output y_3, which is a linear combination of the state variables ($y = Cx$, $C \in \mathbb{R}^{3\times 270}$). 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, with a suggested step size of 0.005. 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

3.1.2 Specifications

The verification goal is to check the ranges reachable by the output y_3, which is a linear combination of the state variables. In addition to the safety specification, for each version there is an UNSAT instance that serves 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 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.

3.1.3 Results

Results of the international space station benchmark for state y_3 over time are shown in Fig. 1 and Fig. 2. The computation times of various tools for the building benchmark are listed in Tab. 1.

Note CORA CORA was run with a step size of 0.01 and with a zonotope order of 30 for benchmark version ISSF01. For version ISSC01 a step size of 0.02 and a zonotope order of 10 was used.

Note C2E2 C2E2 can solve ISS02 when the initial sets for some variables are slightly reduced. With the original initial set, the computation never seems to stop. Due to the reduced initial set, ISU02 cannot be properly verified by C2E2. C2E2 solve the model using time step 0.005 and K value 2000.

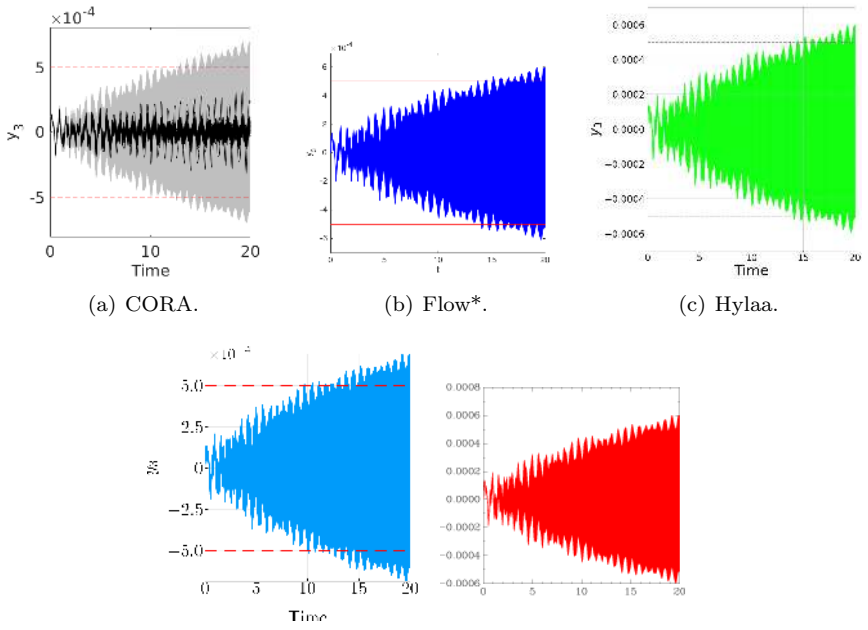
Note Flow* Both of the safe and unsafe properties on the time-varying and time invariant models are solved using the new C++ API in Flow*. Due to the numerical instability in handling large-scale interval matrices, we ignore the roundoff errors only in computing the product of two matrices whose entries are all rationals. For the model ISSC01, we use the time stepsize 0.02 and the TM order 3, while the matrix exponential is firstly overapproximated by a TM of order 10 and then conservatively truncated to order 3. For the model ISSF01, we use a smaller stepsize which is 0.005 along with a lower TM order which is 2. The matrix exponential is firstly overapproximated by a TM of order 6 and then conservatively truncated to order 2. In all of the tests, we use the precision 200 in MPFR library. It is worth mentioning that the flowpipes are initial set independent, which means that they can be reused for any other initial sets.

Note Hylaa Hylaa and Hylaa-continuous were run with a step size of 0.005. The plots for Hylaa are at discrete points in time. They look continuous since the time-step used is fairly small.

Note JuliaReach The algorithm was run with a step size of 0.005 for all scenarios but ISS01, where a step size of 0.0006 was used. There is a *check* mode for property checking that makes the minimum support function evaluations (the whole reach pipe is built with the *reach* mode). Moreover, a specialized option was designed for this benchmark and used to verify ISS01; it adds less over-approximation error by keeping the inputs lazy over time.

Note SpaceEx SpaceEx was run with the LGG scenario. The sampling was chosen as 0.005 for ISSF01 and 0.05 for ISSC01. The template directions were taken to be $\pm y_3$, so only two directions. 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.

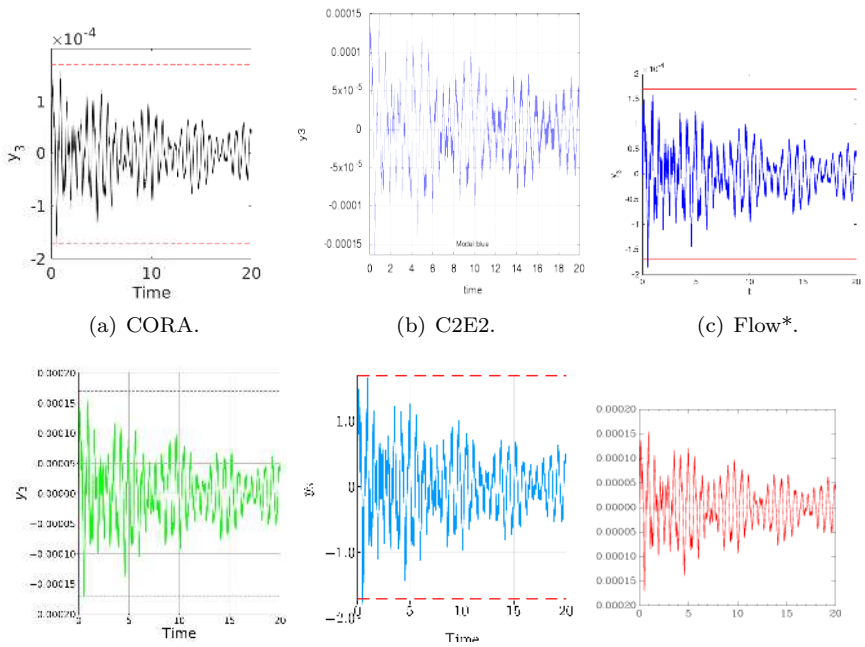
Since SpaceEx does not currently support the plotting of algebraic variables, we used the following trick to plot y_3 over time: we introduced a state variable z with dynamics $\dot{z} = -1000(z - y_3)$. Since the time constant for z is about two orders of magnitude below that of y_3, we expect the plots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



(a) CORA. (b) Flow*. (c) Hylaa.

(d) JuliaReach. (e) SpaceEx.

Figure 1: ISS: Reachable sets of y_3 plotted over time for the uncertain input case.



(a) CORA. (b) C2E2. (c) Flow*.

(d) Hylaa. (e) JuliaReach. (f) SpaceEx.

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

Table 1: Computation Times for the International Space Station Benchmark

	computation time in [s]				**platform**	
	ISSF01		**ISSC01**			**machine**
tool	**ISS01**	**ISU01**	**ISS02**	**ISU02**	**language**	**(Sec. A)**
CORA	370	200	6.46	0.29	MATLAB	M_{CORA}
C2E2	–	–	29.86	–	C++	M_{C2E2}
Flow*	212	154	99.3	55.7	C++	$M_{\text{Flow*}}$
SpaceEx	37.6	40.1	29.5	32.9	C++	$M_{SpaceEx}$
JuliaReach	2.82	1.17	0.23	0.17	Julia	$M_{JuliaReach}$
	discrete-time tools[4]					
Hylaa	34.6	25.2	28.2	0.9	Python	M_{Hylaa}
Hylaa-Continuous	3.3	2.3	2.1	0.06	Python	M_{Hylaa}
JuliaReach	0.40	1.35	0.22	0.16	Julia	$M_{JuliaReach}$

[4]Reachable sets computed in discrete-time are not generally conservative when embedded in continuous time.

3.2 Spacecraft Rendezvous Benchmark

3.2.1 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 [15]; its original continuous dynamics is nonlinear, and the original system is verified in the ARCH-COMP category *Continuous and Hybrid Systems with Nonlinear Dynamics*. However, we find it interesting to see the difference in computation time and verification results if we also investigate the linearized dynamics. 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*. The model is available in C2E2, SDVTool, and SpaceEx format on the ARCH website[5]. The set of initial states is

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

There are two versions of this benchmark:

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 $t = 120min$.

3.2.2 Specifications

Given the thrust constraints of the specified model, in mode *rendezvous attempt*, the absolute velocity must stay below 0.055m/s. In the *aborting* mode, the vehicle must avoid the target, which is modeled as a box $\mathcal{B}$ with 0.2m edge length and the center placed as the origin. In the *rendezvous attempt* the spacecraft must remain within the line-of-sight cone $\mathcal{L} = \{[x, y]^T | (x \geq -100m) \wedge (y \geq x \tan(30°)) \wedge (-y \geq x \tan(30°))\}$. It is sufficient to check these parameters for a time horizon of 200 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 positions 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

SR01 $\forall t \in [0, 200min], \forall x(0) \in \mathcal{X}_0 : (z(t) = z_2) \implies \Big(\sqrt{v_x^2 + v_y^2} \leq 0.055m/s \wedge [s_x, s_y]^T \in \mathcal{L}\Big)$.

SR02 $\forall t \in [0, 200min], \forall x(0) \in \mathcal{X}_0 : (z(t) = z_2) \implies \Big(\sqrt{v_x^2 + v_y^2} \leq 0.055m/s \wedge [s_x, s_y]^T \in \mathcal{L}\Big) \wedge (z(t) = z_3) \implies ([s_x, s_y]^T \notin \mathcal{B})$.

To solve the above specifications, all tools under-approximate the nonlinear constraint $\sqrt{v_x^2 + v_y^2} \leq 0.055m/s$ by an octagon as shown in Fig. 3.

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

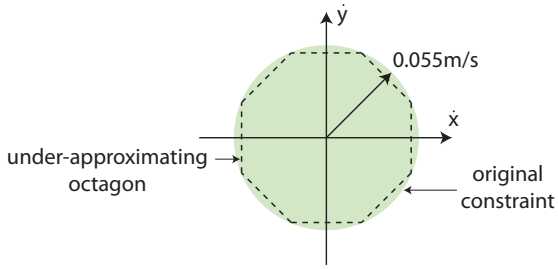


Figure 3: Under-approximation of the nonlinear velocity constraint by an octagon.

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.

3.2.3 Results

Results of the spacecraft rendezvous benchmark for the s_x-s_y-plane are shown in Fig. 4 for the version SRNA01 and in Fig. 5 for the version SRA01. The computation times of various tools for the spacecraft rendezvous benchmark are listed in Tab. 2.

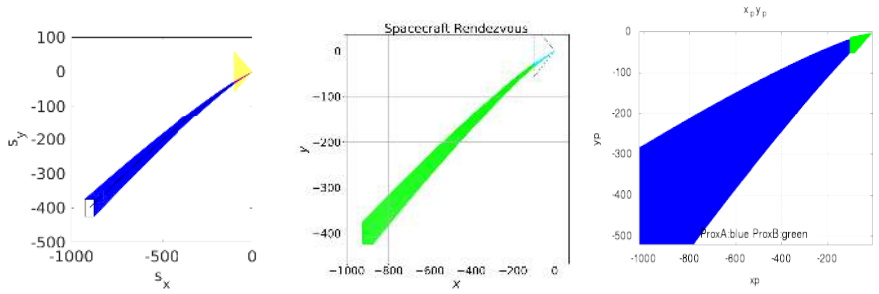
Note CORA For both benchmark versions CORA was run with a zonotope order of 10 and with a step size of $0.2min$ for the mode *approaching*, $0.02min$ for the mode *rendezvous attempt*, and $0.2min$ for the mode *aborting* (does not exist for version SRNA01). The intersections with the guard sets are calculated with the method of Girard and Le Guernic in [26]. In order to find suitable orthogonal directions for the method in [26], 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 C2E2 The run time reported for verifying the model is the sum of the run time for verifying each constraint of the under-approximating octagon (see Fig. 3). For all cases, C2E2 runs with a step size of 0.1. Note that the result for C2E2 is not optimal since C2E2 is currently being updated.

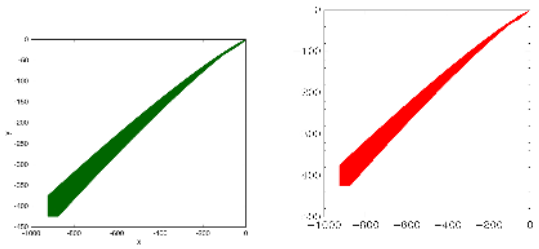
Note Flow* The flowpipes are computed using the user interface of Flow*. For the model SRNA01, we use a stepsize of 0.05, a fixed TM order of 4 for the mode *approaching*, a fixed TM order of 6 for the mode *rendezvous attempt*, and a cutoff threshold $[-10^{-8}, 10^{-8}]$. For the model SRA01, we use the stepsize of 0.01 for the mode *approaching* and mode *aborting*, and the stepsize of 0.005 for the mode *rendezvous attempt*. The TM order is fixed at 3, and we use the same cutoff threshold. All roundoff errors in the reachability computation are kept in the results.

Note Hylaa Hylaa was run with a step size of 0.1. The plots for Hylaa are at discrete points in time. They look continuous since the time-step used is fairly small.

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



(a) CORA. (b) Hylaa. (c) C2E2.



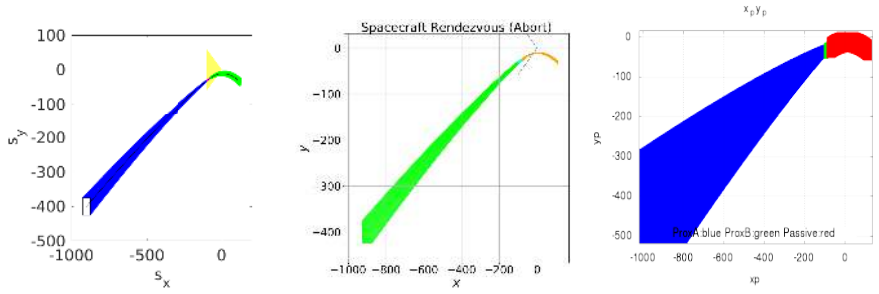
(d) Flow*. (e) SpaceEx.

Figure 4: Reachable sets for the spacecraft rendezvous benchmark in the s_x-s_y-plane for the benchmark variant without maneuver abortion (SRNA01)

Table 2: Computation Times for the Spacecraft Rendezvous Benchmark

	computation time in [s]		**platform**	
tool	**SRNA01 SR01**	**SRA01 SR02**	**language**	**machine (Sec. A)**
CORA	2.1	0.92	MATLAB	M_{CORA}
C2E2	21.95	25.51	C++	M_{C2E2}
Flow*	4.2	16.5	C++	$M_{\text{Flow*}}$
SpaceEx	0.41	0.38	C++	M_{SpaceEx}
	discrete-time tools[6]			
Hylaa	32.7	10.8	Python	M_{Hylaa}

[6]Reachable sets computed in discrete-time are not generally conservative when embedded in continuous time.



(a) CORA. (b) Hylaa. (c) C2E2.

(d) Flow*. (e) SpaceEx.

Figure 5: Reachable sets for the spacecraft rendezvous benchmark in the s_x-s_y-plane for the benchmark variant with maneuver abortion at $t = 120min$ (SRA01)

3.3 Powertrain with Backlash

3.3.1 Model

The powertrain benchmark is an extensible benchmark for hybrid systems with linear continuous dynamics taken from [5, Sec. 6] and [8, 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, clutch, among others, as illustrated in Fig. 6. 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[7].

The set of initial states is

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

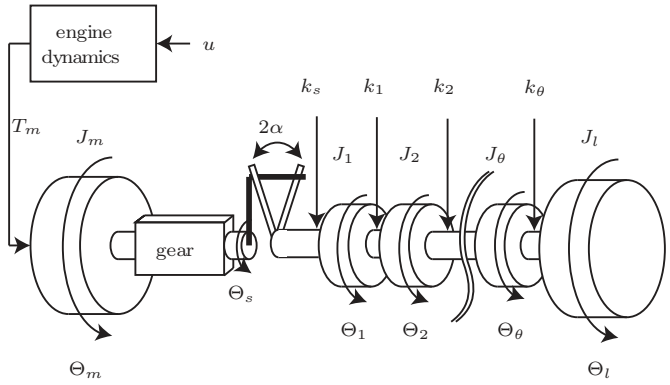


Figure 6: Powertrain model.

3.3.2 Specifications

We analyze an extreme maneuver from an assumed 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.

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

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 $\mathcal{X}_0$.

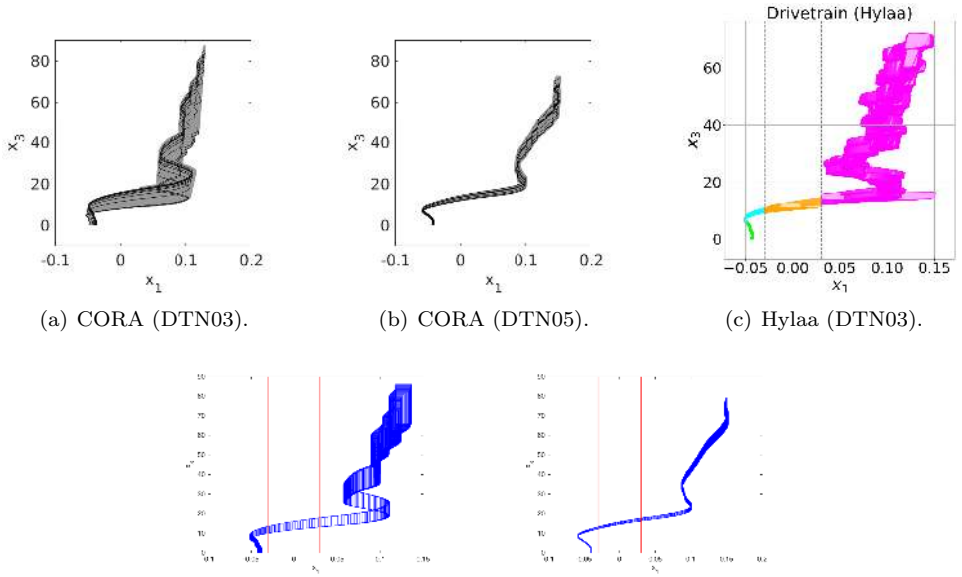
3.3.3 Results

Results of the powertrain benchmark in the x_1-x_3-plane are shown in Fig. 7. The computation times of various tools for the powertrain benchmark are listed in Tab. 3

Note Flow* The flowpipes for all of the 6 models are computed using the new C++ API in Flow*. We use a more sophisticated strategy to aggregate flowpipe/guard intersections while they are very few and consecutive. While there are consecutive flowpipes $F_1, \ldots, F_n$ intersecting a guard, we then compute the hitting as well as the leaving time point and compute a aggregated flowpipe over this time interval. For the cases DTN01, DTN02, and DTN03, we use a fixed stepsize of 0.005 and a fixed TM order 25. The cutoff threshold is $[-10^{-10}, 10^{-10}]$ and the precision is 100. For the cases DTN04, DTN05, and DTN06, we use the same setting except that the stepsize is reduced to 0.0025. All roundoff errors are included in all results.

Note Hylaa Hylaa aggregates sets across discrete transitions, and then de-aggregates them if the aggregated set were to reach a spurious transition. Since sets are aggregated, plots show a certain amount of over-approximation. The resulting, intentional lack of accuracy shows in the plot. Also, the plot for Hylaa is also for the $\theta = 2$ system, since plotting for the $\theta = 22$ system caused GLPK[8] (the LP solver library Hylaa uses) to enter an infinite loop (the bug has been reported). Hylaa was run with a step size of 0.0005. The plots for Hylaa are at discrete points in time. They look continuous since the time-step used is fairly small.

[8] https://www.gnu.org/software/glpk



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

(d) Flow* (DTN03). (e) Flow* (DTN05).

Figure 7: Reachable sets in the x_1-x_3-plane.

Table 3: Computation Times for the Powertrain Benchmark

	computation time in [s]						**platform**	
tool	**DTN01**	**DTN02**	**DTN03**	**DTN04**	**DTN05**	**DTN06**	**language**	**machine (Sec. A)**
CORA	3.28	3.17	3.31	17.8	17.9	18.3	MATLAB	M_{CORA}
Hylaa	1.61	2.69	8.89	7.76	23.3	90.9	Python	M_{Hylaa}
Flow*	2.51	2.51	2.52	103	114	116	C++	$M_{\text{Flow*}}$

3.4 Building Benchmark

3.4.1 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[9]. The initial set and the uncertain input $\mathcal{U}$ are provided in [32, Tab. 2.2]. There are two versions of this benchmark:

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.

3.4.2 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.

3.4.3 Results

Results of the building benchmark for state x_{25} over time are shown in Fig. 8 and Fig. 9. The computation times of various tools for the building benchmark are listed in Tab. 4.

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.

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

Table 4: Computation Times for the Building Benchmark

	computation time in [s]		**platform**	
tool	**BLDC01 BDS01**	**BLDF01 BDS01**	**language**	**machine (Sec. A)**
CORA	1.64	1.85	MATLAB	M_{CORA}
CORA/SX	1.01	1.04	C++	$M_{SpaceEx}$
C2E2	16.051	–	C++	M_{C2E2}
Flow*	14.7	14.3	C++	$M_{\text{Flow*}}$
HyDRA	0.47	–	C++	M_{HyDRA}
SpaceEx	2.0	2.2	C++	$M_{SpaceEx}$
JuliaReach	1.76	1.88	Julia	$M_{JuliaReach}$
	discrete-time tools[10]			
Hylaa	1.8	2.7	Python	M_{Hylaa}
Hylaa-Continuous	0.8	1.2	Python	M_{Hylaa}
JuliaReach	1.76	1.84	Julia	$M_{JuliaReach}$

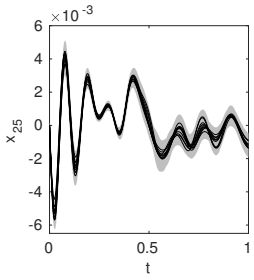
Note C2E2 C2E2 is able to solve BLDC01. It cannot solve BLDF01 because arbitrarily changing bounded inputs is currently not supported by the tool. The step size set in C2E2 is 0.001. Note that the result for C2E2 is not optimal since C2E2 is currently been updated.

Note Flow* The results from Flow* are computed using the C++ API while we ignore the roundoff errors only in computing the product of two matrices with rational entries. We use the same setting as that used in the last year except that the precision is set to be 200 which is much higher. The time stepsize is 0.008, we firstly overapproximate the matrix exponential by an order 25 TM, and then conservatively truncate it to order 3. The computed flowpipes are initial set independent.

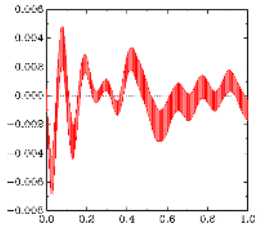
Note Hylaa Hylaa and Hylaa-continuous were run with a step size of 0.005. The plots for Hylaa are at discrete points in time. They look continuous since the time-step used is fairly small.

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.

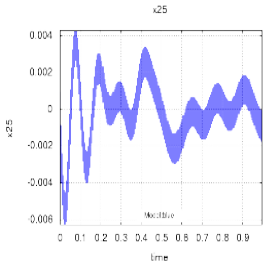
[10] Reachable sets computed in discrete-time are not generally conservative when embedded in continuous time.



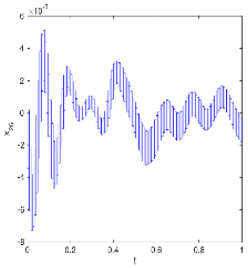
(a) CORA.



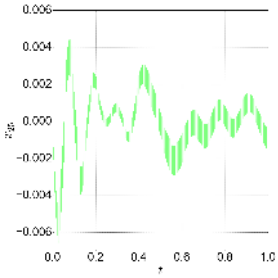
(b) CORA/SX.



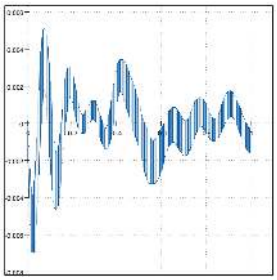
(c) C2E2.



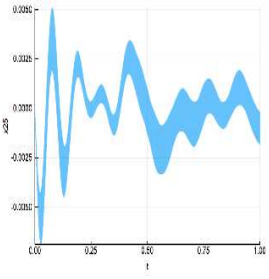
(d) Flow*.



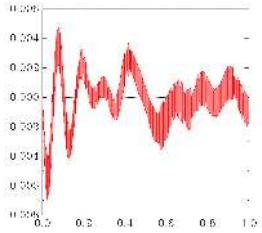
(e) Hylaa.



(f) HyDRA.

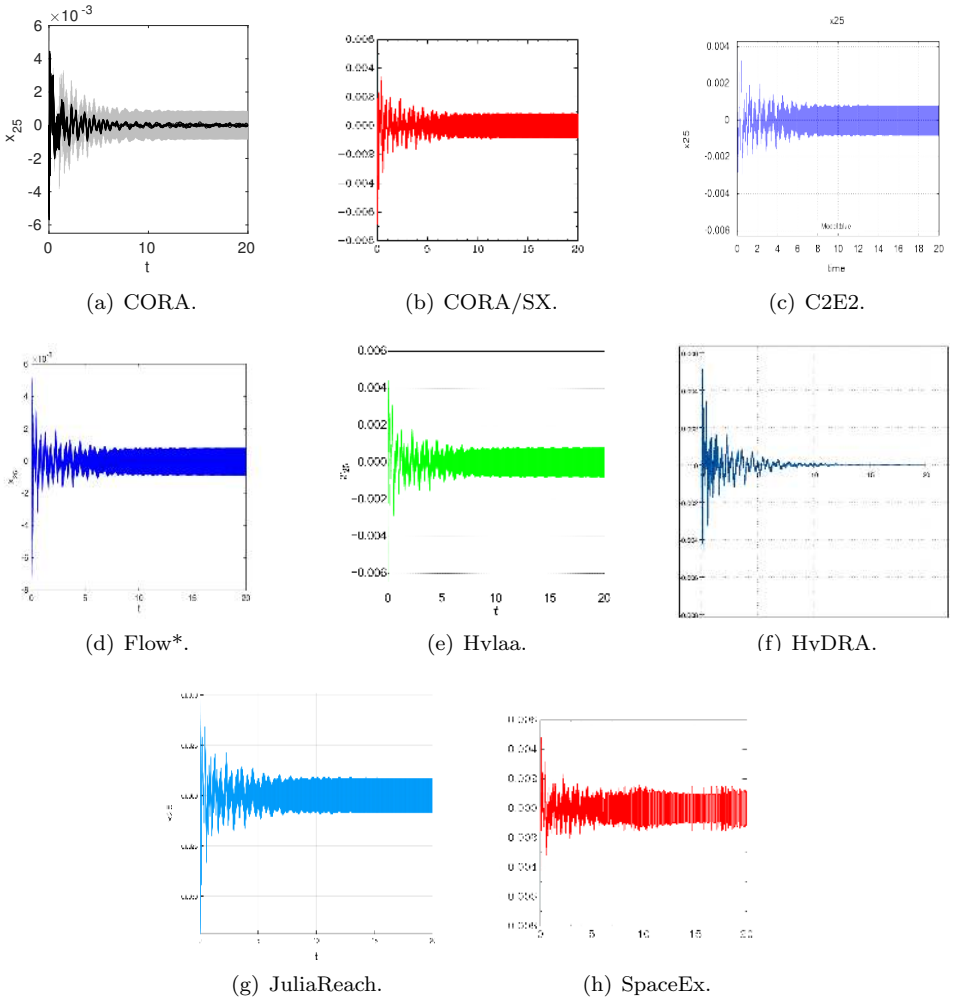


(g) JuliaReach.



(h) SpaceEx.

Figure 8: Building: Reachable sets of x_{25} plotted over time up to time 1. Some tools additionally show possible trajectories.



(a) CORA. (b) CORA/SX. (c) C2E2.

(d) Flow*. (e) Hvlaa. (f) HyDRA.

(g) JuliaReach. (h) SpaceEx.

Figure 9: Building: Reachable sets of x_{25} plotted over time up to time 20. Some tools additionally show possible trajectories.

3.5 Platooning Benchmark

3.5.1 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. 10(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. 10(b). Note that the transitions have must-semantics, i.e., they take place as soon as possible.

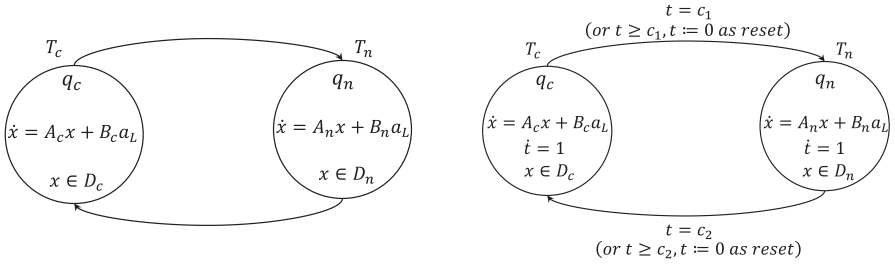
PLAD01: $c_1 = c_2 = 5$.

PLANxy (loss at nondeterministic times) The loss of communication occurs at any time $t \in [t_b, t_c]$. The clock t is reset when communication is lost, and 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[11].

Discussion The arbitrary-loss scenario (PLAA) subsumes the other two instances (PLAD, PLAN). There seems to be no reason why the upper bound t_c should be greater than t_b. Loss of communication after a time longer than t_b is included by reestablishing communication in zero time. Tools could possibly exploit this by minimizing the model before the analysis.



(a) Arbitrary switching.

(b) Controlled switching.

Figure 10: Two options presented in the original benchmark proposal [13]. On the left, the system can switch arbitrarily between the modes, while on the right, mode switches are only possible at given points in time.

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

3.5.2 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], $x_7(t) \geq -d_{min}$ [m], where d_{min} =xy [m].

BND50: $d_{min} = 50$.

BND42: $d_{min} = 42$.

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], $x_7(t) \geq -d_{min}$ [m], where d_{min} =xy [m].

UNB50: $d_{min} = 50$.

UNB42: $d_{min} = 42$.

UNB30: $d_{min} = 30$.

3.5.3 Different Paths to Success

CORA CORA can re-write the hybrid automaton as a purely continuous system with uncertain parameters. This idea is also known as *continuization* [6, 7]. After introducing the uncertain matrix

$$\mathcal{A} = \{\alpha A_c + (1 - \alpha A_n) | \alpha \in [0, 1]\}$$

we can abstract Fig. 10(a) by

$$\dot{x}(t) \in \mathcal{A}x \oplus \tilde{\mathcal{U}},$$

where $\oplus$ denotes the Minkowski addition and $\tilde{\mathcal{U}} = B_c\mathcal{U}$ ($B_c = B_n$). The tool *CORA* uses the continuization approach to solve the system with arbitrary switching. Please note that the exact reachable set of Fig. 10(a) encloses the one of Fig. 10(b), which is a special case.

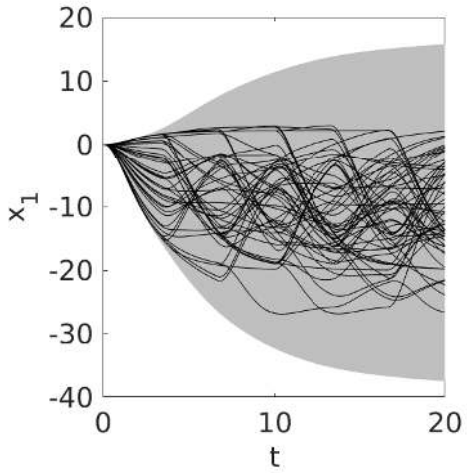
3.5.4 Results

Results of the platoon benchmark for state x_1 over time are shown in Fig. 11, Fig. 12, and Fig. 13. The computation times of various tools for the platoon benchmark are listed in Tab. 5.

Note CORA For the unbounded case, the reachable set at $t = 50$ is increased by 1% and it is checked when this set is re-entered.

Note Flow* The results are computed using the C++ API of Flow* such that time-triggered jumps are much better handled by a new feature. In the last year, Flow* was not able to prove the property BND30 on the model PLAD01, as we mentioned that much better results could be obtained using the API. Therefore, we present much better results this year using the C++ API of Flow*. All approximation errors as well as roundoff errors are included. For the property BND42, we use the stepsize of 0.1, the TM order of 4, the cutof threshold $[-10^{-12}, 10^{-12}]$, and the precision of 100. The TM remainders of the flowpipes are kept every the other jump. While for the property BND30, we use the same setting except that the TM remainders are kept in all jumps.

Note HyDRA As HyDRA implements CEGAR-based path refinement, we can observe erroneous runs in the plot (see Figure 12(d)) which intersect the set of bad states (bottom, red). The refined set of reachable states is depicted in a darker blue.

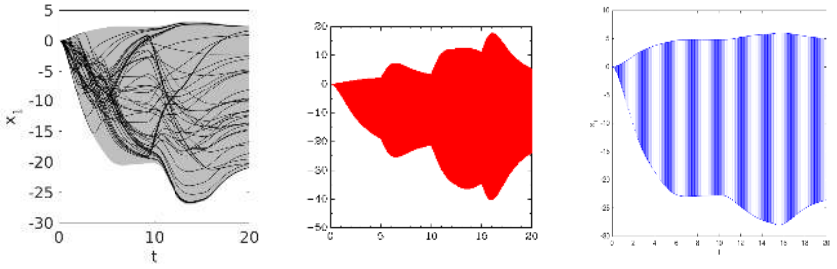


(a) CORA.

Figure 11: PLAA01: Reachable sets of x_1 plotted over time. CORA additionally shows possible trajectories.

Table 5: Computation Times for the Platoon Benchmark

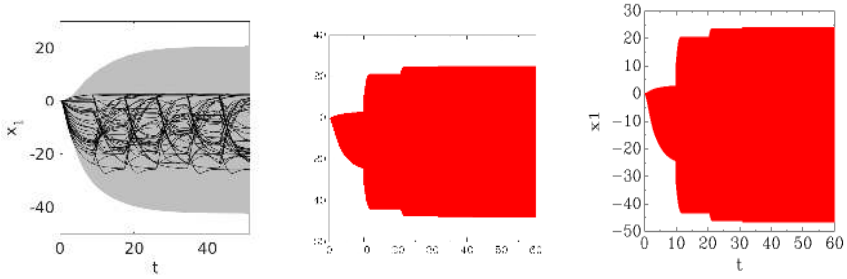
	computation time in [s]					**platform**	
tool	**PLAA01 BND50**	**PLAA01 BND42**	**PLAD01 BND42**	**PLAD01 BND30**	**PLAN01 UNB50**	**language**	**machine (Sec. A)**
CORA	4.17	13.7	0.57	1.42	85.4	MATLAB	M_{CORA}
CORA/SX	–	–	0.41	–	–	C++	$M_{SpaceEx}$
Flow*	–	–	0.91	5.20	–	C++	$M_{\text{Flow*}}$
HyDRA	–	–	3.6	–	–	C++	M_{HyDRA}
SpaceEx	–	–	0.8	9.94	159.1	C++	$M_{SpaceEx}$
XSpeed	–	–	0.98	8.89	29.41	C++	M_{XSpeed}



(a) CORA (BND30). (b) CORA/SX (BND42). (c) Flow* (BND30).

(d) HyDRA (BND42). (e) SpaceEx (BND30). (f) XSpeed (BND30).

Figure 12: 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 13: PLAN01: Reachable sets of x_1 plotted over time.

3.6 Gearbox Benchmark

3.6.1 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[12] and as a Simulink model[13]. 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 [16]. 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 a smaller initial set. The full uncertainty in x-direction could be considered by splitting the uncertainty in x-direction and aggregating the individual results.

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

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

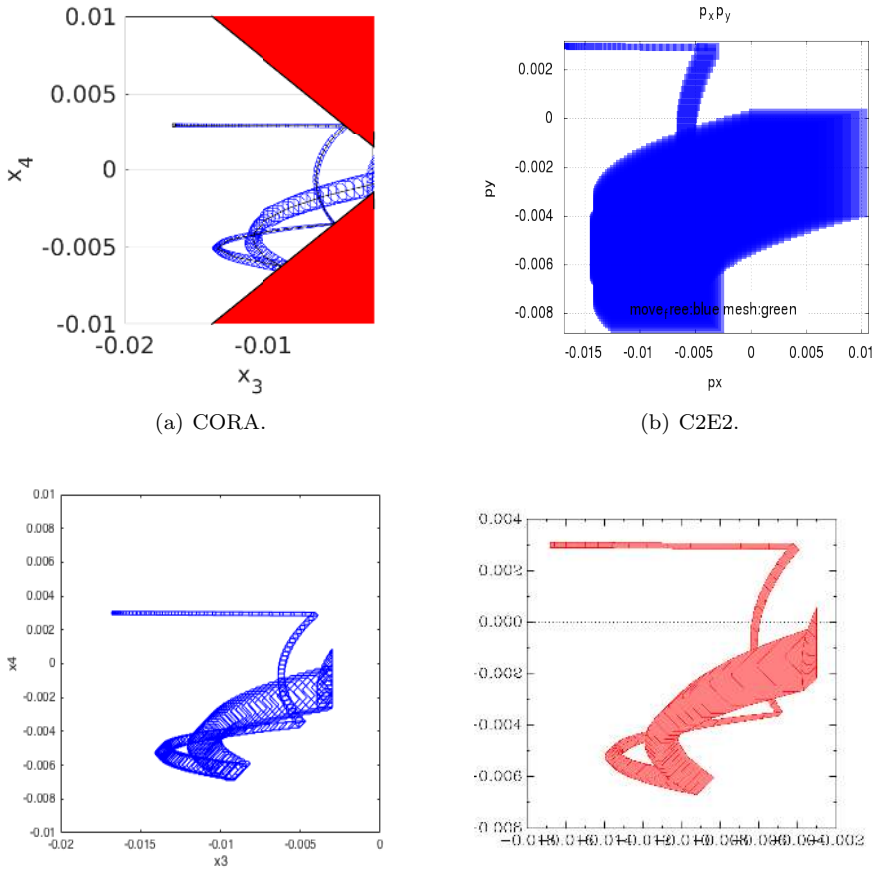
3.6.3 Results

Results of the platoon benchmark for state x_3 and x_4 are shown in Fig. 14. The computation times of various tools for the gearbox benchmark are listed in Tab. 6.

Table 6: Computation Times of the Gearbox Benchmark

	computation time in [s]	**platform**	
tool	**GRBX01-MES01**	**language**	**machine (Sec. A)**
CORA	0.29	MATLAB	M_{CORA}
C2E2	0.30	C++	M_{C2E2}
Flow*	0.23	C++	$\mathrm{M}_{\mathrm{Flow*}}$
SpaceEx	0.14	C++	$\mathrm{M}_{SpaceEx}$

[12] cps-vo.org/node/34375
[13] cps-vo.org/node/34374



(a) CORA.

(b) C2E2.

(c) Flow*.

(d) SpaceEx.

Figure 14: Gearbox: Reachable sets of x_3 and x_4. Some tools additionally show possible trajectories.

Note CORA CORA was run with a time step size of $0.0011s$ and a zonotope order of 20. The intersections with the guard sets were calculated with the method of Girard and Le Guernic [26]. In order to find suitable orthogonal directions for the method in [26], 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 C2E2 C2E2 can only solve the model when the number of transition is limited to 4. As the number of transition increases, over-bloating happens to the variables. C2E2 uses step size 0.001 and K value 2000 while computing the reach tube. Note that the result for C2E2 is not optimal since C2E2 is currently being updated.

Note Flow* The flowpipes are computed using the user interface of Flow*. All computational parameters are same as those used in the last year: the step size is 0.001, the TM order is 3, and the precision is 100 for floating-point numbers. Octagon overapproximations for the flowpipes are plotted.

4 Conclusion and Outlook

This report presents the results on the second friendly competition for the formal verification of continuous and hybrid systems with linear continuous dynamics as part of the ARCH'18 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 a much more favorable scalability compared to the previous year. For instance, many tools could solve the international space station benchmark in a few seconds although it has 270 state variables, which has more than five times as many state variables as the largest benchmark from last year (building benchmark). Another example is the powertrain benchmark, which many tools could verify despite the fact that it is rather large (up to 51 state variables and hybrid). We expect that the development of all tools is continuing so that we can solve even more challenging benchmarks in the future.

We would also like to encourage other tool developers to consider participating next year. All authors agree that although the participation consumes time, we have all learned about new aspects that we would like to improve in the next releases. Also, we were able to fix small inconsistencies between benchmark descriptions and the files attached to the benchmarks due to their heavy use by several tools. Information about the competition in 2019 will be announced on the ARCH website.

5 Acknowledgments

The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921, and by DST-SERB, Government of India, under grant number YSS/2014/000623.

A Specification of Used Machines

A.1 $\mathbf{M}_{CORA}$

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

A.2 $\mathbf{M}_{C2E2}$

- Processor: Intel Core i5-3470 CPU @ 3.20GHz x 4
- Memory: 8 GB
- Average CPU Mark on `www.cpubenchmark.net`: 6680 (full), 1915 (single thread)

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

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

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

A.4 $\mathbf{M}_{HyDRA}$

- Processor: Intel Core i7-4790K CPU @ 4.00GHz x 8
- Memory: 15.9 GB
- Average CPU Mark on www.cpubenchmark.net: 11185

A.5 $\mathbf{M}_{Hylaa}$

- Processor: Intel Core i5-5300U @ 2.30GHz x 4
- Memory: 15.9 GB
- Average CPU Mark on www.cpubenchmark.net: 3755 (full), 1527 (single thread)

A.6 $\mathbf{M}_{JuliaReach}$

- Processor: Intel Core i5-7300HQ @ 3.50GHz x 4
- Memory: 15.4 GB
- Average CPU Mark on www.cpubenchmark.net: 6833 (full), 1879 (single thread)

A.7 $\mathbf{M}_{SpaceEx}$

- Processor: Intel Core i7-7920HQ CPU @ 3.1GHz x 4
- Memory: 16 GB
- Average CPU Mark on www.cpubenchmark.net: 10230 (full), 2161 (single thread)

A.8 $\mathbf{M}_{XSpeed}$

- Processor: Intel Core i7-4770 CPU @ 3.4GHz x 4
- Memory: 8 GB
- Average CPU Mark on www.cpubenchmark.net: 9806

References

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

[2] M. Althoff, S. Bak, D. Cattaruzza, X. Chen, G. Frehse, R. Ray, and S. Schupp. ARCH-COMP17 category report: Continuous and hybrid systems with linear continuous dynamics. In *Proc. of the 4th International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 143–159, 2017.

[3] 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.

[4] 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*, 2018.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] S. Bak, H.-D. Tran, and T. T. Johnson. Numerical verification of affine systems with up to a billion dimensions. *CoRR*, abs/1804.01583, 2018.

[13] 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.

[14] 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.

[15] N. Chan and S. Mitra. Verifying safety of an autonomous spacecraft rendezvous mission. In *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems, collocated with Cyber-Physical Systems Week (CPSWeek) on April 17, 2017 in Pittsburgh, PA, USA*, pages 20–32, 2017.

[16] 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.

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

[18] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of CAV'13*, volume 8044 of *LNCS*, pages 258–263. Springer, 2013.

[19] X. Chen, S. Mover, and S. Sankaranarayanan. Compositional relational abstraction for nonlinear hybrid systems. *ACM Trans. Embedded Comput. Syst.*, 16(5):187:1–187:19, 2017.

[20] X. Chen and S. Sankaranarayanan. Model predictive real-time monitoring of linear systems. In *Proc. of RTSS'17*, pages 297–306. IEEE Computer Society, 2017.

[21] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2e2: A verification tool for stateflow models. In *TACAS*, 2015.

[22] 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.

[23] 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.

[24] 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. Int. Conf. Computer Aided Verification, CAV*, LNCS 6806, pages 379–395. Springer, 2011.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] S. Sankaranarayanan and A. Tiwari. Relational abstractions for continuous and hybrid systems. In *Proc. of CAV'11*, volume 6806 of *LNCS*, pages 686–702. Springer, 2011.

[31] 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.

[32] 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*, 2017.