EPiC Series in Computing

Volume 48, 2017, Pages 143–159

ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems

ARCH-COMP17 Category Report: Continuous and Hybrid Systems with Linear Continuous Dynamics

Matthias Althoff[1], Stanley Bak[2], Dario Cattaruzza[3], Xin Chen[4], Goran Frehse[5], Rajarshi Ray[6], and Stefan Schupp[7]

[1] Technische Universität München, Department of Informatics, Munich, Germany
althoff@in.tum.de
[2] Air Force Research Laboratory, Dayton, OH, United States
[3] University of Oxford, Oxford, UK
dario.cattaruzza@ox.ac.uk
[4] University of Colorado, Boulder, CO, United States
xinchen@colorado.edu
[5] Univ. Grenoble Alpes, Grenoble, France
goran.frehse@imag.fr
[6] National Institute of Technology Meghalaya, Shillong, India.
rajarshi.ray@nitm.ac.in
[7] RWTH Aachen University, Theory of hybrid systems, Aachen, Germany
stefan.schupp@cs.rwth-aachen.de

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 2017. In its first edition, seven tools have been applied to solve three different benchmark problems in the category for linear continuous dynamics (in alphabetical order): Axelerator, CORA, Flow*, HyDRA, Hylaa, SpaceEx, and XSpeed. The result 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.

This report summarizes results obtained in the 2017 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, which are applied to different benchmark problems presented in Sec. 3. The results are also 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 Sec Sec. 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.

Axelerator This tool analyzes linear time-invariant systems by means of *abstract acceleration* [9, 20]. The proposed method relies on the relaxation of the continuous dynamics to calculate a one-step evaluation of the reach tube for an infinite time horizon, which combines all possible states visited by the system at any time from a known initial set and arbitrarily varying

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

inputs. Currently, Axelerator only operates on discrete-time models. Continuous-time models are discretized by a user-defined sampling time. Axelerator operates in two modes with sound (-S) or unsound (-U) numeric abstractions. Counterexample-Guided Abstraction Refinement is used to maintain a high precision when required.

CORA The tool *COntinuous Reachability Analyzer* (CORA) [1, 2] 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 `http://www6.in.tum.de/Main/SoftwareCORA`.

Flow* The tool Flow* [13, 12] computes *Taylor model flowpipes* as overapproximations for continuous and hybrid system reachable sets. For the systems defined by Linear Time-Invariant (LTI) and Linear Time-Varying (LTV) ODEs which could be uncertain, Flow* computes *symbolic flowpipes* which essentially are higher-order overapproximations for the exact 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 Taylor model order in use. Unlike the 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 with a different initial set or unsafe condition. Besides, symbolic flowpipes can be easily extended to generate relational abstractions [23] for LTI and LTV systems. In order to guarantee the conservativeness in computations, Flow* treats all floating-point numbers as intervals, for example, a matrix exponential e^A is overapproximated by an interval matrix.

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 [19, 24], a C++ library for reachability analysis. HyPro provides different implementations of state 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 state 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 [5, 6] 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) [7]. 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 a reachability approach based on a version of the generalized star set representation [14], where

the star's predicates are restricted to be conjunctions of linear constraints. This allows the use of linear programming in order to determine if a guard is enabled, or if an unsafe state is reached. 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`.

SpaceEx *SpaceEx* is a tool for computing reachability of hybrid systems with complex, high-dimensional dynamics [15, 17, 16]. 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 modern multicore architectures to 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 [21, 22]. The performance of hybrid systems reachability analysis is enhanced using the adapted G.J. Holzmann's algorithm and the task parallel algorithm, both of which propose variants of parallel breadth-first exploration of the hybrid automaton [18].

3 Verification of Benchmarks

We have agreed on three benchmarks, each one of them having unique features. The *building benchmark* [25, No. 2] is a purely continuous linear system with the largest number of continuous state variables among all benchmark problems. Less continuous state variables are considered in the *platooning benchmark* [8], but one can arbitrarily switch between two discrete states: a normal operation mode and a communication-failure mode. Finally, the gearbox benchmark in [11] has the smallest number of continuous state variables, but the reachable set does not contract to a steady state and the reachable set for one point in time might intersect multiple guards at once. Next, let us briefly discuss some specificities of each benchmark problem.

Types of Inputs Generally, we distinguish between three types of inputs: a) Fixed inputs, where $u(t)$ is precisely known. If in addition, $u(t) = \texttt{const}$, the linear system becomes an affine system $\dot{x}(t) = Ax(t) + b$; the gearbox benchmark has affine dynamics. b) 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: $u(t) = \texttt{const}$. This case is not considered in the building benchmark.

c) 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. The benchmark problems *building* and *platoon* consider this kind of uncertain input. 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. In this edition, we have two different ways of solving the platooning example, which are discussed in Sec. 3.2.

3.1 Building Benchmark

3.1.1 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 by the attachment of the benchmark on the ARCH website[2]. The initial set and the uncertain input $\mathcal{U}$ are provided in [25, Tab. 2.2]. Please note that the input to the system is allowed to change arbitrarily over time within the specified bounds.

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.

3.1.2 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, underapproximation, 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 overapproximation is not too coarse, and the UNSAT instance demonstrates that the overapproximation is indeed conservative, at least in the narrow sense of the specification.

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.

[2] cps-vo.org/group/ARCH

Remark The proposed benchmark is derived from another benchmark in [10]. The original file attachment of [25] used truncated floating point numbers from [10]. To remove any possible confusion, a new file attachment[3] has been uploaded, which matches the original benchmark.

3.1.3 Results

Results of the building benchmark for state x_{25} over time are shown in Fig. 1 and Fig. 2.

Note Hylaa The plots for Hylaa are at discrete points in time. It looks continuous since the time-step used (0.005) is fairly small.

The computation times of various tools for the building benchmark are listed in Tab. 1.

Table 1: Computation Times for the Building Benchmark

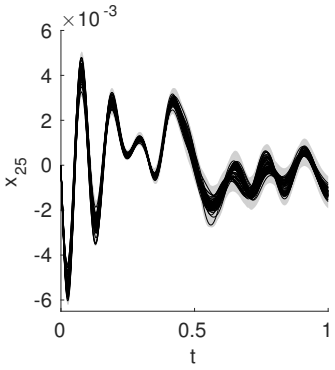
	computation time in [s]		**platform**	
tool	**BLDC01 BDS01**	**BLDF01 BDS01**	**language**	**machine (Sec. A)**
CORA	6.5	6.5	MATLAB	M_{CORA}
Flow*	122	145	C++	$\mathrm{M}_{\mathrm{Flow*}}$
HyDRA	3.5	–	C++	M_{HyDRA}
SpaceEx	1.9	2.2	C++	$\mathrm{M}_{SpaceEx}$
	discrete-time tools[5]			
Axelerator-U	–	18	C++	$\mathrm{M}_{Axelerator}$
Axelerator-S	–	563	C++	$\mathrm{M}_{Axelerator}$
Hylaa	1.7	2.7	Python	M_{Hylaa}

Setting for Flow*. For the building benchmark, the results from Flow* are computed based on the time step size 0.008, a Taylor Model (TM) order 25, and a precision for floating-point numbers of 100. Flow* computes a symbolic flowpipe overapproximation for the reachable set in a time step. The flowpipe is represented by a TM of order 25 over the variables $x_1, \ldots, x_{48}$ which are representing the initial set, and t which is the time variable. The Taylor model is globally valid; that is, one may simply change the initial set and the flowpipe immediately gives you a reachable set overapproximation from the new initial set. Notice that the overapproximations in the plot are further enlarged compared to the results of Flow* for plotting reasons. For example, the region defined by $x_{25} \geq 0.0048$ is reached in the figure; however, it is not reachable as proved by Flow*. Besides, all round-off errors are included by the overapproximations.

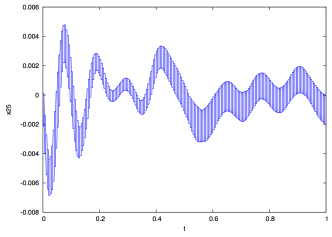
[3] cps-vo.org/node/26533

[4] 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.

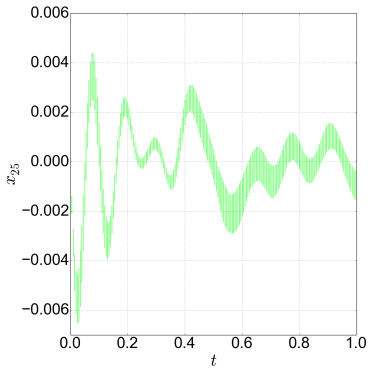
[5] Reachable sets computed in discrete-time are not generally conservative when embedded in continuous time.



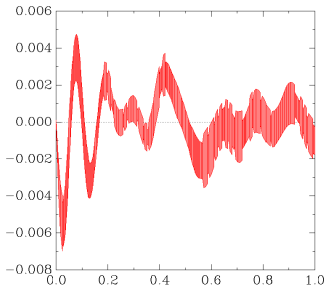
(a) CORA.



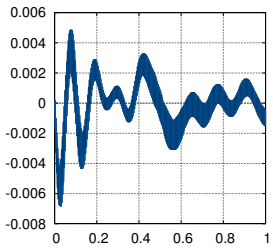
(b) Flow*.



(c) Hylaa.



(d) SpaceEx (BLDF01-BDS01).



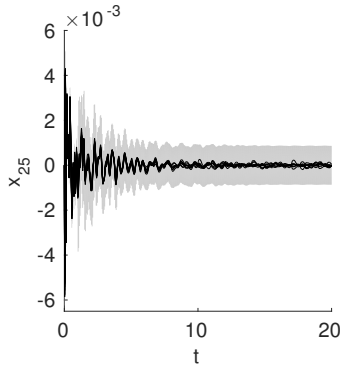
(e) HyDRA (affine adaption, $\delta = 0.004$).

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

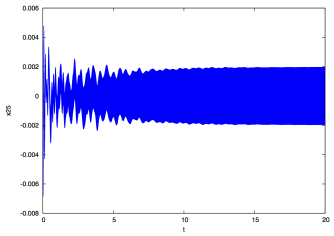
3.2 Platooning Benchmark

3.2.1 Model

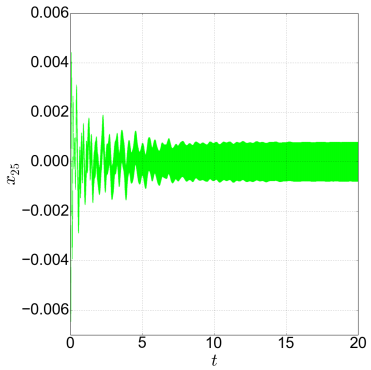
The platooning benchmark considers a platoon of three vehicles following each other. This benchmark considers loss of communication between vehicles. The initial discrete state is q_c. Three scenarios are considered for the loss of communication:



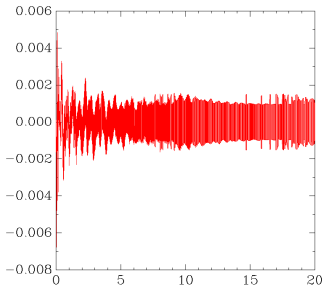
(a) CORA.



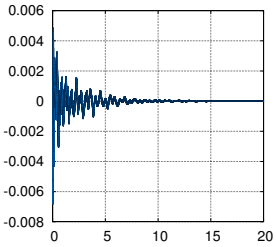
(b) Flow*.



(c) Hylaa.



(d) SpaceEx (BLDF01-BDS01).[4]



(e) HyDRA (affine adaption, $\delta = 0.004$).

Figure 2: Reachable sets of x_{25} plotted over time up to time 20. Some tools additionally show possible trajectories.

PLAA01 (arbitrary loss) The loss of communication can occur at any time, see Fig. 4(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. 4(b). Note that the transitions have must-semantics, i.e., they take place as soon as possible.

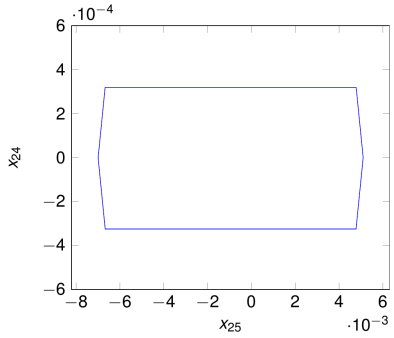


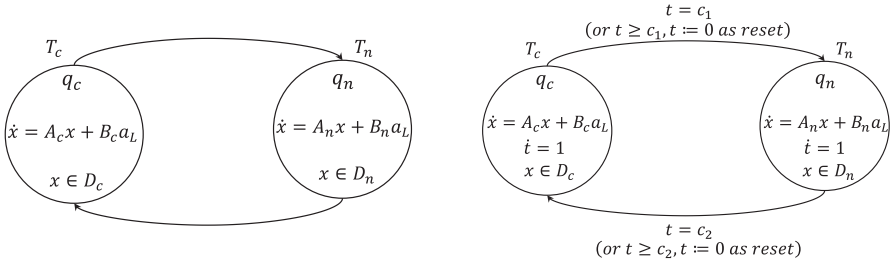
Figure 3: Reachable values of x24 and x25 over an infinite time horizon ($t_{min} = 0, t_{max} \geq 1$, ($ie$ $k_{min} = 0, k_{max} \geq 200$ samples))

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

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 4: Two options presented in the original benchmark proposal [8]. 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.

3.2.2 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: $d_{min} = 30$.

3.2.3 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* [3, 4]. After introducing the uncertain matrix

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

we can abstract Fig. 4(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. 4(a) encloses the one of Fig. 4(b), which is a special case.

3.2.4 Results

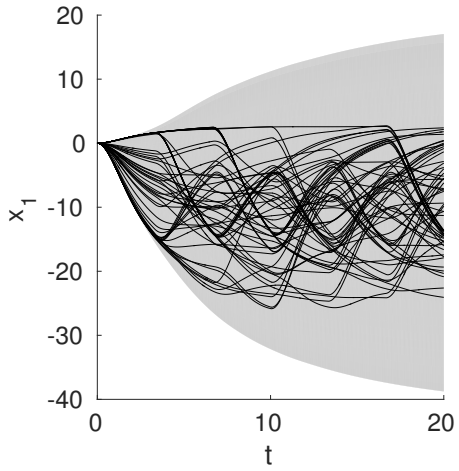
Results of the platoon benchmark for state x_1 over time are shown in Fig. 5, Fig. 6, and Fig. 7.

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.

The computation times of various tools for the platoon benchmark are listed in Tab. 2.

Table 2: 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	7	61	1.5	3	178	MATLAB	M_{CORA}
Flow*	–	–	4.1	–	–	C++	$M_{\text{Flow*}}$
SpaceEx	–	–	0.79	12.1	177	C++	$M_{SpaceEx}$
XSpeed	–	–	3.53	–	–	C++	M_{XSpeed}



(a) CORA.

Figure 5: PLAA01: Reachable sets of x_1 plotted over time. CORA additionally shows possible trajectories.

Setting for Flow*. Since Flow* requires a bounded number of jumps during the reachability computation, only the benchmark PLAD01-BND42 is considered. The computational parameters are given as follows: We use the step size 0.01, the TM order 3, the cutoff threshold 10^{-12}, and the precision 100 for floating-point numbers. For both of the jumps, we aggregate the flowpipe/guard intersections by an interval, which causes additional overestimation. This, however, is unnecessary in this example since at most one flowpipe intersects a guard so that one only needs to change the continuous dynamics at the time when a jump is enabled. Using the API of Flow*, one can easily obtain a better result.

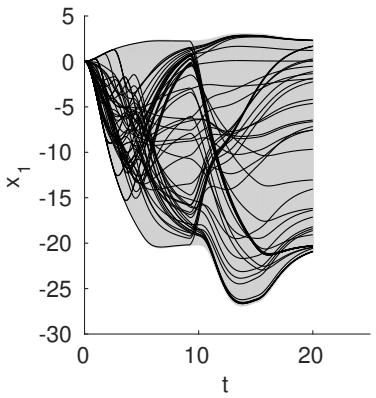
3.3 Gearbox Benchmark

3.3.1 Model

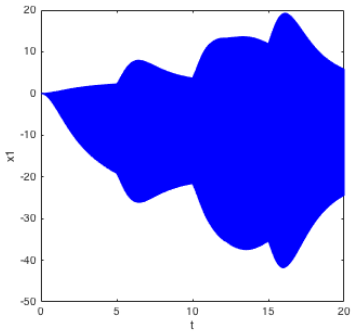
The gearbox benchmark models the motion of two meshing gears. When the gears collide, an elastic impact takes place. As soon as the gears are close enough, the gear is considered *meshed*. The model includes a monitor state that checks whether the gears are meshed or free. Once the monitor reaches the 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 [11]. For some initial continuous states, the target region is reached without any discrete transition, while for other initial states, several discrete transitions are required.

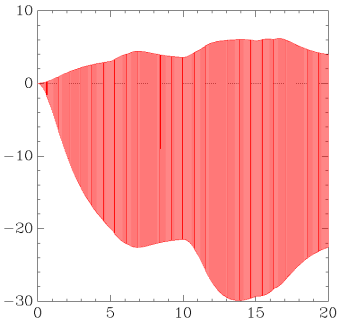
In the original benchmark, the position uncertainty in the direction of the velocity vector of the gear teeth (x-direction) is across the full width of the gear spline. Uncertainties of the position and velocity in y-direction, which is perpendicular to the x-direction, are considered to be smaller. 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.



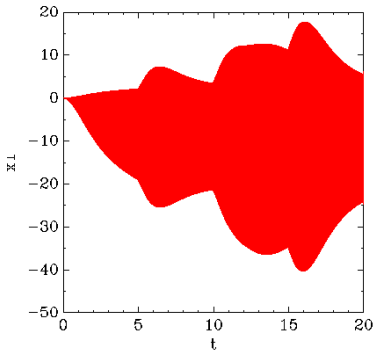
(a) CORA (BND30).



(b) Flow* (BND42).



(c) SpaceEx (BND30).



(d) XSpeed (BND42).

Figure 6: PLAD01: Reachable sets of x_1 plotted over time. Some tools additionally show possible trajectories.

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

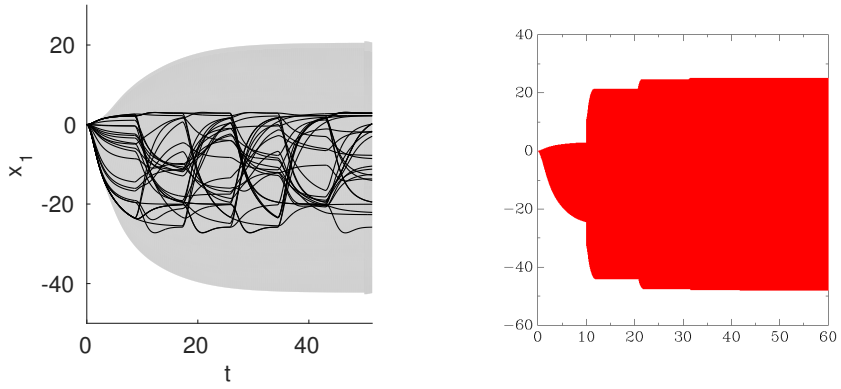
3.3.2 Specification

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: $\mathit{free} \wedge t \geq 0.2$ or $x_5 \geq 20$

3.3.3 Results

Results of the platoon benchmark for state x_3 and x_4 are shown in Fig. 8.



(a) CORA (UNB50). (b) SpaceEx (UNB50).

Figure 7: PLAN01: Reachable sets of x_1 plotted over time.

Computation Times The computation times of various tools for the gearbox benchmark are listed in Tab. 3.

Table 3: Computation Times of the Gearbox Benchmark

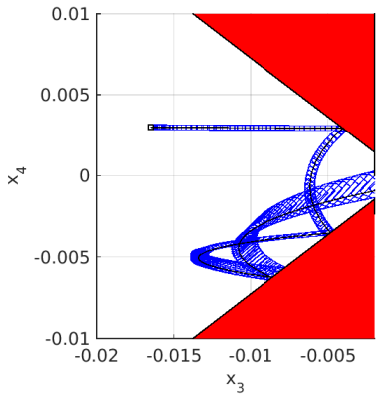
	computation time in [s]	**platform**	
tool	**GRBX01-MES01**	**language**	**machine (Sec. A)**
CORA	6	MATLAB	M_{CORA}
Flow*	0.23	C++	$M_{\text{Flow*}}$
SpaceEx	0.14	C++	$M_{SpaceEx}$

Setting for Flow*. Flow* uses the step size 0.001, TM order 3, and the precision 100 for floating-point numbers. Octagon overapproximations for the flowpipes are plotted; notice that they are much coarser than the exact overapproixmations computed by Flow* and are only used for plotting.

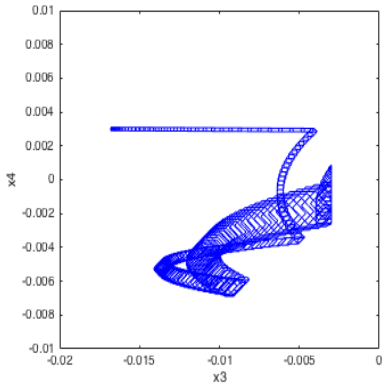
4 Conclusion and Outlook

This report presents the results on a first friendly competition for the formal verification of continuous and hybrid systems with linear continuous dynamics as part of the ARCH'17 workshop. The reports of other categories can be found in the proceedings and on the ARCH website: cps-vo.org/group/ARCH.

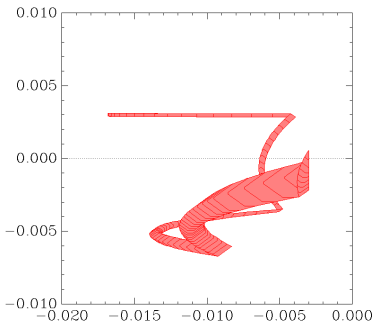
A major observation of the results is that all participating tools have a much more favorable scalability compared to the tools under development 10 years ago. All tools could solve the building benchmark consisting of 48 continuous state variables. We expect that the development of all tools is continuing so that more tools will be able to handle hybrid dynamics as well.



(a) CORA.



(b) Flow*.



(c) SpaceEx.

Figure 8: Reachable sets of x_3 and x_4. Some tools additionally show possible trajectories.

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 could 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 2018 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.

A Specification of Used Machines

A.1 $\mathbf{M_{Axelerator}}$

- Processor: Intel Core i7-6700HQ @ 2.60GHz x4
- Memory: 6GB
- Average CPU Mark on www.cpubenchmark.net: 8133 (full), 1803 (single thread)

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

- Processor: Intel Core i7-3520M CPU @ 2.90GHz x 4
- Memory: 7.6 GB
- Average CPU Mark on www.cpubenchmark.net: 4515 (full), 1785 (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 $\mathbf{M}_{SpaceEx}$

- Processor: Intel Core i7-4850HQ CPU @ 2.30GHz x 4
- Memory: 15.9 GB
- Average CPU Mark on www.cpubenchmark.net: 9057 (full), 1966 (single thread)

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

- 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 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.

[3] 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.

[4] 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.

[5] Stanley Bak and Parasara Sridhar Duggirala. Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In *International Conference on Hybrid Systems: Computation and Control*, 2017.

[6] Stanley Bak and Parasara Sridhar Duggirala. Rigorous simulation-based analysis of linear hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2017.

[7] Stanley Bak and Parasara Sridhar Duggirala. Simulation-equivalent reachability of large linear systems with inputs. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV'17)*, 2017.

[8] 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.

[9] Dario Cattaruzza, Alessandro Abate, Peter Schrammel, and Daniel Kroening. Unbounded-time analysis of guarded LTI systems with inputs by abstract acceleration. In *SAS*, volume 9291 of *LNCS*, pages 312–331. Springer, 2015.

[10] Y. Chahlaoui and P. Van Dooren. A collection of benchmark examples for model reduction of linear time invariant dynamical systems. Technical report, University of Manchester, 2002.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] Goran Frehse. Reachability of hybrid systems in space-time. In Alain Girault and Nan Guan, editors, *2015 International Conference on Embedded Software, EMSOFT 2015, Amsterdam, Netherlands, October 4-9, 2015*, pages 41–50. IEEE, 2015.

[17] Goran Frehse, Rajat Kateja, and Colas Le Guernic. Flowpipe approximation and clustering in space-time. In Calin Belta and Franjo Ivancic, editors, *Proceedings of the 16th international conference on Hybrid systems: computation and control, HSCC 2013, April 8-11, 2013, Philadelphia, PA, USA*, pages 203–212. ACM, 2013.

[18] Amit Gurung, Arup Deka, Ezio Bartocci, Sergiy Bogomolov, Radu Grosu, and Rajarshi 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.

[19] Hypro project website. Available at `http://ths.rwth-aachen.de/research/projects/hypro/`.

[20] Bertrand Jeannet, Peter Schrammel, and Sriram Sankaranarayanan. Abstract acceleration of general linear loops. In *POPL*, pages 529–540. ACM, 2014.

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

[22] Rajarshi Ray, Amit Gurung, Binayak Das, Ezio Bartocci, Sergiy Bogomolov, and Radu Grosu. XSpeed: Accelerating reachability analysis on multi-core processors. In *Proc. of HVC 2015*, volume 9434 of *LNCS*, pages 3–18, 2015.

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

[24] Stefan Schupp, Erika Abraham, Ibtissem Ben Makhlouf, and Stefan 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.

[25] 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.