EPiC Series in Computing
Volume 54, 2018, Pages 1–13
ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems

ARCH-COMP18 Category Report: Hybrid Systems with Piecewise Constant Dynamics

Goran Frehse[1], Alessandro Abate[2], Dieky Adzkiya[3], Lei Bu[4], Mirco Giacobbe[5], Muhammad Syifa'ul Mufid[2], and Enea Zaffanella[6]

[1] Univ. Grenoble Alpes, Grenoble, France
goran.frehse@univ-grenble-alpes.fr
[2] University of Oxford, Oxford, U.K.
{aabate,muhammad.syifaul.mufid}@cs.ox.ac.uk
[3] Department of Mathematics, Institut Teknologi Sepuluh Nopember, Indonesia
dieky@matematika.its.ac.id
[4] State Key Laboratory of Novel Software Techniques, Nanjing University, Nanjing, Jiangsu, P.R. China
bulei@nju.edu.cn
[5] IST Austria, Klosterneuburg, Austria
mgiacobbe@ist.ac.at
[6] Department of Mathematical, Physical and Computer Sciences University of Parma, Italy
enea.zaffanella@unipr.it

Abstract

This report presents results of a friendly competition for formal verification of continuous and hybrid systems with piecewise constant dynamics. The friendly competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in 2018. In this second edition, five tools have been applied to solve five different benchmark problems in the category for piecewise constant dynamics: BACH, Lyse, PHAVer, PHAVer-lite, and VeriSiMPL. Compared to last year, a new tool has participated (PHAVer-lite) and a benchmark has been made more complex (Dutch Railway Network). 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 piecewise constant dynamics up to this date.

1 Introduction

Disclaimer The presented report of the ARCH friendly competition for *continuous and hybrid systems with piecewise constant 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 piecewise constant bounds on the dynamics. In each location (mode, piece of the hybrid state space), the dynamics are given by a differential inclusion of the form

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

where $\mathcal{U}$ is a convex subset of $\mathbb{R}^n$. Tool developers run their tools summarized in Sec. 2 on different benchmark problems presented in Sec. 3 and report the results obtained from their own machines also in Sec. 3.

The results reported by each participant have not been checked by an independent authority and are obtained on the machines of the tool developers. Thus, one has to factor in the computational power of the used processors summarized in Sec. A as well as the efficiency of the programming language of the tools. It is not the goal of the friendly competition to rank the results, the goal is to present the landscape of existing solutions in a breadth that is not possible by scientific publications in classical venues. Those would require the presentation of novel techniques, while this report showcases the current state of the art.

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 Piecewise Constant Dynamics* are introduced below in alphabetical order.

BACH BACH [12, 11] is a bounded reachability checker for Linear Hybrid Automata (LHA) model, Hybrid Systems with Piecewise Constant Dynamics (HPWC) in the term of ARCH competition. The tool provides GUI for LHA modeling and also bounded reachability checkers for both single automaton and automata network. Be different from classical bounded checkers of LHA, which encodes the "complete" bounded state space of the system into a huge SMT

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

problem, BACH conducts the bounded checking in a "path-oriented" layered style. It finds potential paths which can reach the target location on the graph structure first, then encodes the feasibility of such path into a linear programming problem and solve it afterwards. In this way, as the number of paths in the discrete graph structure of an LHA under a given bound is finite, all candidate paths can be enumerated and checked one by one to tackle the bounded reachability analysis of LHA. Furthermore, the memory usage is well controlled as it only encodes and solves one path at a time. Meanwhile, BACH provides an efficient way to locate the infeasible path segment core when a path is reported as infeasible to guide the backtracking in the graph structure traversing to achieve good performance [20]. Such infeasible path segments can also be used to derive complete state arguments under certain conditions [21].

Lyse Lyse is a tool for the reachability analysis of convex hybrid automata, namely hybrid automata with piecewise constant dynamics, whose constraints are possibly non-linear but required to be convex. In this class are HPWC whose flow is contrained in rectangles, polyhedra, but also ellipses and parabolae. Linear hybrid automata are a special case. Lyse performs forward reachability analysis by means of template-polyhedra, whose directions are incrementally extracted from spurious counterexamples. The extraction is performed by a novel technique that generates interpolants by means of convex programming [8].

PHAVer PHAVer [14] is a formal verification tool for computing reachability and equivalence (simulation relation) of hybrid systems. It can handle the class of Linear Hybrid Automata (LHA), whose continuous dynamics is characterized by piecewise constant bounds on the derivatives and whose discrete jumps can be a convex linear predicate over the variables before and after the jump. PHAVer uses standard operations on polyhedra for the reachability computation over an infinite time horizon (similar to those used in HyTech), and the algorithm for computing simulation relations is a straightforward extension of these. Using unbounded integer arithmetic, the computations are exact and formally sound. While termination of LHA is undecidable, PHAVer provides formally sound, precise overapproximation and widening operators that can force termination at the cost of reduced precision. These operators also simplify the computed continuous sets and dynamics of the system, and may result in a considerable speed-up without much loss in precision. Since 2011, PHAVer is continued as a plugin to the tool platform SpaceEx.

PHAVer-lite PHAVer-lite is a variant of the verification tool PHAVer, sharing the same capabilities and formal soundness guarantees, as well as providing the same high level interface as a plugin to the SpaceEx platform. The main difference with respect to PHAVer is the adoption of the new polyhedra library PPLite [7]: thanks to a novel representation and conversion algorithm [6] for NNC (Not Necessarily Closed) polyhedra, PPLite is able to obtain significant efficiency improvements with respect to the classical polyhedra implementation used in PHAVer (which is based on the Parma Polyhedra Library [5]). Besides changing the polyhedra backend, PHAVer-lite also applies a couple of more general improvements to the reachability algorithm (e.g., in the implementation of the continuous post operator), leading to other remarkable efficiency gains for some of the benchmarks.

VeriSiMPL This toolbox [1, 4] is used to generate finite abstractions and reachability of max-plus-linear (MPL) systems. VeriSiMPL leverages the piecewise affine (PWA) dynamics generated from an MPL system and some operations over difference-bound matrices (DBM) [13]. Abstractions are characterized as finite-state labeled transition systems (LTS). The finite

LTS abstractions are shown to either simulate or to bisimulate the original MPL system [2]. The resulting LTS are to be verified against given specifications expressed as formulae in linear temporal logic (LTL) and computation tree logic (CTL). The toolbox intends to leverage the SPIN and NuSMV model checkers. With regards to the reachability of MPL systems, VeriSiMPL is able to compute the forward and backward reach sets of MPL systems exactly [3]. The initial and final states are expressed as a union of finitely many DBM. The reachability algorithm uses the PWA dynamics associated with an MPL system and some operations on DBM.

3 Verification of Benchmarks

3.1 Adaptive Cruise Controller

Model The adaptive cruise controller is a distributed system for assuring that safety distances in a platoon of cars are satisfied [9]. It is inspired by a related benchmark in [17]. For n cars, the number of discrete states is 2^n and the number of continuous variables is n. Each variable x_i encodes the relative position of the i-th car, for $i = 0, \ldots, n-1$. The car i-th car is considered to be in front of the $i+1$-th car. The relative velocity of each car has a drift $|\dot{x}_i - \dot{x}_{i+1}| \leq 1$ when cruising and $|\dot{x}_i - \dot{x}_{i+1} - \varepsilon| \leq 1$ when recovering, where ε is the slow-down parameter. The cars can stay in cruise mode as long as the distance to the preceding vehicle is greater 1. The can go into recovery mode when the distance is smaller than 2.

ACCS*nn* The model with *nn* cars, $\varepsilon = 2$. This model is considered safe with respect to specification UBS*nn* (no collisions).

ACCU*nn* The model with *nn* cars, $\varepsilon = 0.9$. This model is considered unsafe with respect to specification UBS*nn* (collisions are possible).

Specification The distance between adjacent cars should be positive:

$$\dot{x}_{\texttt{ldr}} - \dot{x} > 0,$$

where x and $x_{\texttt{ldr}}$ are the positions of the car and the car in front, respectively.

UBD*nn* For $i = 0, \ldots, n-1$: $x_i - x_{i+1} > 0$.

Results The computation times of various tools are listed in Tab. 1.

3.2 Distributed controller

Model The benchmark is an extension of the benchmarks presented in [16], to which multiple sensors with multiple priorities have been added. It models the distributed controller for a robot that reads and processes data from different sensors. A scheduler component determines what sensor data must be read according to the priority of the sensor. The controller has 1 continuous and n discrete variables, the scheduler has n continuous and n discrete variables, and each sensor has 1 continuous variable. The controller has 4 locations, the scheduler has $1 + n$, and each sensor has 4 locations. The product automaton has $4 \times (1 + n) \times 4$ locations, $n + 2$ continuous variables and $2n$ discrete variables. Note that some tools, such as PHAVer, do not support discrete variables and may model the discrete variables as continuous variables.

DISC*nn* The model with *nn* sensors. This model is considered safe with respect to specification UBS*nn*.

Table 1: Computation Times of the Adaptive Cruise Controller.

instance	ACCS05 UBD01	ACCU05 UBD01	ACCS06 UBD01	ACCU06 UBD01	ACCS07 UBD01	ACCU07 UBD01		
safety	safe	unsafe	safe	unsafe	safe	unsafe		
#vars.	5	5	6	6	7	7		
#locs.	32	32	64	64	128	128		
tool	computation time in [s]						lang.	machine
Lyse	1.08	≈ 0	–	–	573.35	0.233	C++	M_{Lyse}
PHAVer	9.4	13.7	461	13430	∞	∞	C++	M_{PHAVer}
PHAVer-lite	1.0	0.9	38.1	22.4	–	–	C++	$M_{PHAVer-lite}$

Table 2: Computation Times of the Distributed Controller.

instance	DISC02 UBS02	DISC03 UBS03	DISC04 UBS04	DISC05 UBS05		
safety	safe	safe	safe	safe		
#vars.	8	11	14	17		
#locs.	48	64	80	96		
tool	computation time in [s]				lang.	machine
PHAVer	1.1	∞	∞	∞	C++	M_{PHAVer}
PHAVer-lite	0.1	548.0	–	–	C++	$M_{PHAVer-lite}$
	bounded-depth tools[2]					
BACH	–	–	$0.1(B:10)$	$0.2(B:10)$	C++	M_{BACH}

Specification The system is considered safe if at no point in time all sensors send data simultaneously.

UBSnn It is never the case that all nn sensors are in location `send`.

Results The computation times of various tools are listed in Tab. 2.

3.3 Dutch Railway Network

We consider a finite-horizon safety problem over max-plus-linear (MPL) systems. More precisely, given a PWA system generated from an MPL system, a time horizon N, a set of initial conditions X_0 expressed as a difference-bound matrix (DBM) [13], an unsafe set S described as a DBM, we wanted to know whether the system can reach the unsafe set within the given time horizon.

[2]The search depth p is indicated as $(B:p)$, and counted as the number of discrete transitions taken.

Model In [19, p. 30], a subset of Dutch railway networks is modeled as a max-plus-linear (MPL) system. That model has 14 state variables $x_1(k), \dots, x_{14}(k)$ representing the k-th departure time of trains at the following stations:

- $x_1(k)$: Den Haag CS to Utrecht,
- $x_2(k)$: Utrecht to Deventer (via Amersfoort and Apeldoorn),
- $x_3(k)$: Deventer to Enschede (via Almelo and Hengelo),
- $x_4(k)$: Enschede to Deventer (via Hengelo and Almelo),
- $x_5(k)$: Deventer to Utrecht (via Apeldoorn and Amersfoort),
- $x_6(k)$: Utrecht to Rotterdam CS and to Den Haag CS,
- $x_7(k)$: Den Haag CS to Amersfoort (via Utrecht),
- $x_8(k)$: Rotterdam CS to Amersfoort (via Utrecht),
- $x_9(k)$: Amersfoort to Zwolle,
- $x_{10}(k)$: Zwolle to Leeuwarden and to Groningen,
- $x_{11}(k)$: Leeuwarden to Amersfoort (via Zwolle),
- $x_{12}(k)$: Groningen to Amersfoort (via Zwolle),
- $x_{13}(k)$: Amersfoort to Utrecht,
- $x_{14}(k)$: Utrecht to Den Haag CS and to Rotterdam CS.

Every MPL system can be transformed into a discrete-time piecewise affine (PWA) system in event domain [15]. The PWA system corresponding to the above MPL system has 12 regions:

- Region 1, given by $\{x : 40 + x_1 \geq 72 + x_6, 55 + x_7 \geq 54 + x_8, 55 + x_7 \geq 37 + x_5, 90 + x_{11} \geq 93 + x_{12}\}$, with dynamics

$$\begin{array}{lll} x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\ x_2(k) = 40 + x_1(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\ x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 90 + x_{11}(k-1), \\ x_4(k) = 41 + x_3(k-1), & x_9(k) = 55 + x_7(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\ x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), & \end{array}$$

- Region 2, given by $\{x : 40 + x_1 \geq 72 + x_6, 55 + x_7 \geq 54 + x_8, 55 + x_7 \geq 37 + x_5, 93 + x_{12} \geq 90 + x_{11}\}$, with dynamics

$$\begin{array}{lll} x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\ x_2(k) = 40 + x_1(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\ x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 93 + x_{12}(k-1), \\ x_4(k) = 41 + x_3(k-1), & x_9(k) = 55 + x_7(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\ x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), & \end{array}$$

- Region 3, given by $\{x : 72+x_6 \geq 40+x_1, 55+x_7 \geq 54+x_8, 55+x_7 \geq 37+x_5, 90+x_{11} \geq 93+x_{12}\}$, with dynamics

$$\begin{array}{lll}
x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\
x_2(k) = 72 + x_6(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\
x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 90 + x_{11}(k-1), \\
x_4(k) = 41 + x_3(k-1), & x_9(k) = 55 + x_7(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\
x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), &
\end{array}$$

- Region 4, given by $\{x : 72+x_6 \geq 40+x_1, 55+x_7 \geq 54+x_8, 55+x_7 \geq 37+x_5, 93+x_{12} \geq 90+x_{11}\}$, with dynamics

$$\begin{array}{lll}
x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\
x_2(k) = 72 + x_6(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\
x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 93 + x_{12}(k-1), \\
x_4(k) = 41 + x_3(k-1), & x_9(k) = 55 + x_7(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\
x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), &
\end{array}$$

- Region 5, given by $\{x : 40+x_1 \geq 72+x_6, 54+x_8 \geq 55+x_7, 54+x_8 \geq 37+x_5, 90+x_{11} \geq 93+x_{12}\}$, with dynamics

$$\begin{array}{lll}
x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\
x_2(k) = 40 + x_1(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\
x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 90 + x_{11}(k-1), \\
x_4(k) = 41 + x_3(k-1), & x_9(k) = 54 + x_8(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\
x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), &
\end{array}$$

- Region 6, given by $\{x : 40+x_1 \geq 72+x_6, 54+x_8 \geq 55+x_7, 54+x_8 \geq 37+x_5, 93+x_{12} \geq 90+x_{11}\}$, with dynamics

$$\begin{array}{lll}
x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\
x_2(k) = 40 + x_1(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\
x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 93 + x_{12}(k-1), \\
x_4(k) = 41 + x_3(k-1), & x_9(k) = 54 + x_8(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\
x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), &
\end{array}$$

- Region 7, given by $\{x : 72+x_6 \geq 40+x_1, 54+x_8 \geq 55+x_7, 54+x_8 \geq 37+x_5, 90+x_{11} \geq 93+x_{12}\}$, with dynamics

$$\begin{array}{lll}
x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\
x_2(k) = 72 + x_6(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\
x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 90 + x_{11}(k-1), \\
x_4(k) = 41 + x_3(k-1), & x_9(k) = 54 + x_8(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\
x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), &
\end{array}$$

- Region 8, given by $\{x : 72+x_6 \geq 40+x_1, 54+x_8 \geq 55+x_7, 54+x_8 \geq 37+x_5, 93+x_{12} \geq 90+x_{11}\}$, with dynamics

$$\begin{array}{lll}
x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\
x_2(k) = 72 + x_6(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\
x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 93 + x_{12}(k-1), \\
x_4(k) = 41 + x_3(k-1), & x_9(k) = 54 + x_8(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\
x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), &
\end{array}$$

- Region 9, given by $\{x : 40+x_1 \geq 72+x_6, 37+x_5 \geq 55+x_7, 37+x_5 \geq 54+x_8, 90+x_{11} \geq 93+x_{12}\}$, with dynamics

$$\begin{array}{lll} x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\ x_2(k) = 40 + x_1(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\ x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 90 + x_{11}(k-1), \\ x_4(k) = 41 + x_3(k-1), & x_9(k) = 37 + x_5(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\ x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), & \end{array}$$

- Region 10, given by $\{x : 40+x_1 \geq 72+x_6, 37+x_5 \geq 55+x_7, 37+x_5 \geq 54+x_8, 93+x_{12} \geq 90+x_{11}\}$, with dynamics

$$\begin{array}{lll} x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\ x_2(k) = 40 + x_1(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\ x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 93 + x_{12}(k-1), \\ x_4(k) = 41 + x_3(k-1), & x_9(k) = 37 + x_5(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\ x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), & \end{array}$$

- Region 11, given by $\{x : 72+x_6 \geq 40+x_1, 37+x_5 \geq 55+x_7, 37+x_5 \geq 54+x_8, 90+x_{11} \geq 93+x_{12}\}$, with dynamics

$$\begin{array}{lll} x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\ x_2(k) = 72 + x_6(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\ x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 90 + x_{11}(k-1), \\ x_4(k) = 41 + x_3(k-1), & x_9(k) = 37 + x_5(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\ x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), & \end{array}$$

- Region 12, given by $\{x : 72+x_6 \geq 40+x_1, 37+x_5 \geq 55+x_7, 37+x_5 \geq 54+x_8, 93+x_{12} \geq 90+x_{11}\}$, with dynamics

$$\begin{array}{lll} x_1(k) = 38 + x_6(k-1), & x_6(k) = 53 + x_5(k-1), & x_{11}(k) = 54 + x_{10}(k-1), \\ x_2(k) = 72 + x_6(k-1), & x_7(k) = 38 + x_{14}(k-1), & x_{12}(k) = 58 + x_{10}(k-1), \\ x_3(k) = 50 + x_2(k-1), & x_8(k) = 36 + x_{14}(k-1), & x_{13}(k) = 93 + x_{12}(k-1), \\ x_4(k) = 41 + x_3(k-1), & x_9(k) = 37 + x_5(k-1), & x_{14}(k) = 16 + x_{13}(k-1). \\ x_5(k) = 41 + x_4(k-1), & x_{10}(k) = 35 + x_9(k-1), & \end{array}$$

The model instance is defined formally as follows:

DRNW02 initial condition $X_0 = \{x : 0 \leq x_i \leq 5, \text{ for all } i = 1, \ldots, 14\}$

The model is easily embedded in a hybrid automaton with a single location, where the time derivative of all variables is zero, and a self-loop transition that models the discrete dynamics for each region.

Specification We have four specifications of interest:

BDR01 there exists a $k = 0, \ldots, 100$ such that $50 \leq x_2(k) - x_7(k) \leq 60$ (satisfied)

BDR02 there exists a $k = 0, \ldots, 100$ such that $70 \leq x_2(k) - x_7(k) \leq 80$ (satisfied)

BDR03 there exists a $k = 0, \ldots, 100$ such that $90 \leq x_2(k) - x_7(k) \leq 100$ (not satisfied)

BDR04 there exists a $k = 0, \ldots, 100$ such that $10 \leq x_2(k) - x_7(k) \leq 20$ (satisfied)

In the sense of a safety specification, the above specifications specify unsafe states. If the unsafe are reachable, the corresponding specification BDR01,. . . ,BDR04 is satisfied.

Table 3: Computation Times of the Dutch Railway Benchmark.

instance	DRNW02 BDR01	DRNW02 BDR02	DRNW02 BDR03	DRNW02 BDR04		
safety	unsafe	unsafe	safe	unsafe		
# vars.	14	14	14	14		
# locs.	1	1	1	1		
tool	computation time in [s]				lang.	machine
VeriSiMPL	0.057	0.030	6.081	0.033	MATLAB	$\text{M}_{VeriSiMPL}$
PHAVer	528.1	528.1	528.1	528.1	C++	M_{PHAVer}
PHAVer-lite	113.0	113.0	113.0	113.0	C++	$\text{M}_{PHAVer-lite}$
SpaceEx[3]	1.2	1.2	1.2	1.2	C++	M_{PHAVer}

Results The computation times of various tools are listed in Tab. 3.

Note PHAVer/SpaceEx Since the iteration count in PHAVer/SpaceEx does not guarantee the actual search depth, we added a counter automaton that models each value of k with a discrete location. The counter is limited to 100 transitions, after which it deadlocks. The tool is then run until a fixed point is found, which guarantees that all values up to $k = 100$ are explored. The flow predicate in the hybrid automaton model was set to `false`, which means that there is no computation of time elapse in the reachability. We therefore expect that the overhead of embedding the discrete-time model in a continuous-time model is minimal. Note that PHAVer/SpaceEx computes the full reach set before checking whether the unsafe states are reachable. This explains why all instances of the specification take the same time. To compare the performance of exact polyhedral computations with that of template polyhedra, we also include the results of SpaceEx running the LGG scenario and box directions. This gives the same variable ranges as PHAVer, but it should be noted that the result is not formally sound due to the double precision floating point used by LGG.

3.4 Fischer's Protocol

Model Fischer's protocol is a time based protocol of mutual exclusion between processes, originally from [18]. The flow constraints are given by $\frac{1}{2} \leq \dot{x}_{\mathtt{1}} \leq \frac{3}{2}, \ldots, \frac{1}{2} \leq \dot{x}_{\mathtt{m}} \leq \frac{3}{2}$, where $x_{\mathtt{i}}$ is the clock of the `i`-th process. The product automaton has $(n+1) \times 4^n$ locations and n variables.

FISCS*nn* protocol with *nn* processes, considered safe with respect to specification UBD*nn*.

FISCU*nn* protocol with *nn* processes, considered unsafe with respect to specification UBD*nn*.

Specification The protocol is correct if no two processes are ever in the critical section at the same time.

UBD*nn* There are no two processes such that both are in location `cs` (critical section) at the same time.

[3]In contrast to the other tools, the results of SpaceEx given here are numerically unsound due to the use of double precision floating point arithmetic.

Table 4: Computation Times of the Fischer Benchmark.

instance	FISCS04 UBD01	FISCU04 UBD01	FISCS05 UBD01	FISCU05 UBD01		
safety	safe	unsafe	safe	unsafe		
# vars.	4	4	5	5		
# locs.	1280	1280	6144	6144		
tool	computation time in [s]				lang.	machine
Lyse	33.59	0.06	859.84	0.16	C++	M_{Lyse}
PHAVer	90.5	579	∞	∞	C++	M_{PHAVer}
PHAVer-lite	12.3	102.2	14722.2	–	C++	$M_{PHAVer-lite}$

Results The computation times of various tools are listed in Tab. 4.

3.5 TTEthernet

Model The TTEthernet protocol is a protocol for the remote synchronization of possibly drifted clocks distributed over multiple components, taken from [10]. The system consists of two compression masters (CM) and k synchronisation masters (SM). Each CM has two clocks cm_i, each SM has one clock sm_i. Both CM and SM are modeled by a hybrid automaton with 4 locations each. The product automaton has $4 + k$ variables and 4^{k+2} locations.

TTESnn protocol with nn SM. This model is considered safe with respect to specification UBDnn. The global time horizon is limited to 3000 ms.

Specification The difference between the clocks of the SM should not exceed a threshold of $2max\_drift$.

UBDnn For all i, j, $sm_i - sm_j \leq 2max\_drift$, where $max\_drift = 0.001$ ms.

Results The computation times of various tools are listed in Tab. 5.

4 Conclusions and Outlook

This report presents the results of the second edition of a 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.

In the spirit of a friendly competition, this report does not provide any ranking of tools. A few casual observations can be made nonetheless. For the reported instances, PHAVer computes the exact set of reachable states and can therefore be regarded as a base line. PHAVer-lite is a variation of PHAVer that uses a different polyhedra library and includes some algorithmic improvements that lead to a considerable speedup. PHAVer is vastly outperformed by Lyse,

[4]The search depth p is indicated as $(B : p)$, and counted as the number of discrete transitions taken.

Table 5: Computation Times of the TTEthernet Benchmark.

instance	TTES05 UBS01	TTES07 UBS01		
safety	safe	safe		
# vars.	9	11		
# locs.	15384	262144		
tool	computation time in [s]		lang.	machine
PHAVer	25.2	113	C++	M_{PHAVer}
PHAVer-lite	1.9	7.7	C++	$\mathrm{M}_{PHAVer-lite}$
	bounded-depth tools[4]			
BACH	$0.15(B:11)$	$0.2(B:11)$	C++	M_{BACH}

which relies on abstraction refinement. VeriSiMPL also vastly outperforms PHAVer in the specific category for which it was developed. The bounded model checker BACH was included for rough comparison and to create a link to the ARCH-COMP category on bounded model checking (HBMC). For the reported depths, BACH performed very well.

More benchmarks and more tools are necessary to obtain a clearer picture of which methods work for which types of systems. The code with which the results have been obtained is publicly available at gitlab.com/goranf/ARCH-COMP. New benchmarks for inclusion in next year's edition will be solicited in a public call.

5 Acknowledgments

The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921.

References

[1] D. Adzkiya and A. Abate. VeriSiMPL: Verification via biSimulations of MPL models. In K. Joshi, M. Siegle, M. Stoelinga, and P.R. D'Argenio, editors, *International Conference on Quantitative Evaluation of Systems*, volume 8054 of *Lecture Notes in Computer Science*, pages 253–256. Springer, Heidelberg, September 2013. http://sourceforge.net/projects/verisimpl/.

[2] D. Adzkiya, B. De Schutter, and A. Abate. Finite abstractions of max-plus-linear systems. *IEEE Transactions on Automatic Control*, 58(12):3039–3053, December 2013.

[3] D. Adzkiya, B. De Schutter, and A. Abate. Computational techniques for reachability analysis of max-plus-linear systems. *Automatica*, 53(0):293–302, 2015.

[4] D. Adzkiya, Y. Zhang, and A. Abate. VeriSiMPL 2: An open-source software for the verification of max-plus-linear systems. *Discrete Event Dynamic Systems*, 26(1):109–145, 2016.

[5] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. Applications of polyhedral computations to the analysis and verification of hardware and software systems. *Theor. Comput. Sci.*, 410(46):4672–4691, 2009.

[6] Anna Becchi and Enea Zaffanella. A direct encoding for NNC polyhedra. In *Computer Aided Verification - 30th International Conference, CAV 2018, Oxford, UK*, 2018. To appear.

[7] Anna Becchi and Enea Zaffanella. An efficient abstract domain for not necessarily closed polyhedra. In *Static Analysis - 25th International Symposium, SAS 2018, Freiburg im Breisgau, Germany*, 2018. To appear.

[8] Sergiy Bogomolov, Goran Frehse, Mirco Giacobbe, and Thomas A Henzinger. Counterexample-guided refinement of template polyhedra. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 589–606. Springer, 2017.

[9] Sergiy Bogomolov, Goran Frehse, Mirco Giacobbe, and Thomas A. Henzinger. Counterexample-guided refinement of template polyhedra. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I*, volume 10205 of *Lecture Notes in Computer Science*, pages 589–606, 2017.

[10] Sergiy Bogomolov, Christian Herrera, and Wilfried Steiner. Benchmark for verification of fault-tolerant clock synchronization algorithms. In *ARCH (2016)*.

[11] Lei Bu, You Li, Linzhang Wang, Xin Chen, and Xuandong Li. BACH 2 : Bounded reachability checker for compositional linear hybrid systems. In *Design, Automation and Test in Europe, DATE 2010, Dresden, Germany, March 8-12, 2010*, pages 1512–1517, 2010.

[12] Lei Bu, You Li, Linzhang Wang, and Xuandong Li. BACH : Bounded reachability checker for linear hybrid automata. In *Formal Methods in Computer-Aided Design, FMCAD 2008, Portland, Oregon, USA, 17-20 November 2008*, pages 1–4, 2008.

[13] D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, chapter 17, pages 197–212. Springer, Heidelberg, 1990.

[14] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. *International Journal on Software Tools for Technology Transfer*, 10:263–279, 2008.

[15] W. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.

[16] Thomas A. Henzinger and Pei-Hsin Ho. HYTECH: the cornell hybrid technology tool. In Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 265–293. Springer, 1994.

[17] Sumit Kumar Jha, Bruce H. Krogh, James E. Weimer, and Edmund M. Clarke. Reachability for linear hybrid automata using iterative relaxation abstraction. In *HSCC*, 2007.

[18] Leslie Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems (TOCS)*, 5(1):1–11, 1987.

[19] Subiono. *On classes of min-max-plus systems and their applications*. PhD thesis, Delft University of Technology, 2000. http://resolver.tudelft.nl/uuid:e5181e99-fb8f-4588-a37a-46ff2007c5c7.

[20] Dingbao Xie, Lei Bu, Jianhua Zhao, and Xuandong Li. SAT-LP-IIS joint-directed path-oriented bounded reachability analysis of linear hybrid automata. *Formal Methods in System Design*, 45(1):42–62, 2014.

[21] Dingbao Xie, Wen Xiong, Lei Bu, and Xuandong Li. Deriving unbounded reachability proof of linear hybrid automata during bounded checking procedure. *IEEE Trans. Computers*, 66(3):416–430, 2017.

A Specification of Used Machines

A.1 M$_{BACH}$

- Processor: Intel(R) Core(TM)2 Quad CPU Q9500 @ 2.83GHz x 4
- Memory: 4 GB

- Average CPU Mark on `www.cpubenchmark.net`: 3636 (full), 1203 (single thread)

A.2 M$_{Lyse}$

- Processor: Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz x 2
- Memory: 4 GB
- Average CPU Mark on `www.cpubenchmark.net`: 3818 (full), 1521 (single thread)

A.3 M$_{PHAVer}$

- 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.4 M$_{PHAVer-lite}$

- Processor: Intel Core i7-3632QM CPU @ 2.20GHz x 4
- Memory: 15.5 GB
- Average CPU Mark on `www.cpubenchmark.net`: 6939 (full), 1566 (single thread)

A.5 M$_{VeriSiMPL}$

- Processor: Intel Core i7-4720HQ CPU @ 2.6GHz x 4
- Memory: 4 GB
- Average CPU Mark on `www.cpubenchmark.net`: 8010 (full), 1912 (single thread)