EPiC Series in Computing

Volume 74, 2020, Pages 1–15

ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)

ARCH-COMP20 Category Report: Hybrid Systems with Piecewise Constant Dynamics and Bounded Model Checking

Lei Bu[1], Alessandro Abate[2], Dieky Adzkiya[3], Muhammad Syifa'ul Mufid[2], Rajarshi Ray[4], Yuming Wu[1], and Enea Zaffanella[5]

[1] State Key Laboratory of Novel Software Techniques, Nanjing University, Nanjing, Jiangsu, P.R. China
bulei@nju.edu.cn
[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] Indian Association for the Cultivation of Science, Kolkata, India
rajarshi.ray@iacs.res.in
[5] Department of Mathematical, Physical and Computer Sciences University of Parma, Italy
enea.zaffanella@unipr.it

Abstract

This report presents the 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 2020. In this fourth edition, five tools have been applied to solve six different benchmark problems in the category for piecewise constant dynamics: BACH, PHAVerLite, PHAVer/SX, TROPICAL, and XSpeed. Compared to last year, we combine the HBMC and HPWC categories of ARCH-COMP 2019 to a new category PCDB (hybrid systems with Piecewise Constant bounds on the Dynamics (HPCD) and Bounded model checking (BMC) of HPCD systems). 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* and *bounded model checking* 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 2020 friendly competition of the ARCH workshop[1] in PCDB category for two types of problems: for verifying hybrid systems with piecewise constant bounds on the dynamics (HPCD) and for bounded model checking (BMC) of HPCD systems.

The PCDB category concerns hybrid systems where 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 $\mathbb{R}^n$. Specifically, the BMC task concerns the bounded model checking of HPCD systems where the bound is described as the depth of the discrete jump of the system.

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 *PCDB: Continuous and Hybrid Systems with Piecewise Constant Dynamics and Bounded Model Checking* are introduced below in alphabetical order.

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

BACH BACH [13, 12] 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 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 [27]. Such infeasible path segments can also be used to derive complete state arguments under certain conditions [28].

PHAVer/SX PHAVer [16] 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 over-approximation 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. This plugin is the tool actually used for the competition: for clarity, in the following we refer to it under the name PHAVer/SX.

PHAVerLite (and PHAVer-lite/SX) PHAVerLite is a variant of the stand alone tool PHAVer, sharing the same capabilities and formal soundness guarantees. The main differences with respect to PHAVer/SX are:

- the use of the modeling syntax of PHAVer (rather than the SpaceEx syntax);
- the adoption of the new polyhedra library PPLite [7, 8], achieving significant efficiency improvements with respect to the PPL [6] implementation used in PHAVer/SX;
- the development of a few novel algorithms and the revisiting of some of the design/implementation trade-offs [9];
- the possibility to experiment with Cartesian factoring techniques [19].

In order to better highlight the efficiency improvements obtained, in the experimental evaluations we will pair the results obtained by PHAVerLite with those that were obtained (in the 2018 edition) by its first prototype PHAVer-lite/SX, which was developed as a SpaceEx plugin.

TROPICAL TROPICAL is an abstraction and formal verification tool for systems defined over tropical algebra such max-plus linear (MPL) systems, min-plus linear (MiPL) systems and interval max-plus linear (IMPL) systems. It manages two types of data structures, Difference-Bound Matrix (DBM) [15] and Difference Logic [14], to express the continuous state-space of the underlying system. The tool allows to solve reachability analysis (to check whether an unsafe set is reachable from a given set of initial conditions) and model checking over time-difference specifications. The tool grows out of the previous software VeriSiMPL [1, 4] and is presently under development, based on the work in [23, 24]. In this report we focus on bounded-horizon properties, however we expect the tool to be usable over infinite-horizon verification problem.

XSpeed *XSpeed* implements algorithms for reachability analysis of continuous and hybrid systems with affine dynamics. The focus of the tool is to exploit the modern multicore architectures to enhance the performance through parallel computations. The algorithms in XSpeed are based on symbolic states represented using support functions. The tool can analyze hybrid automata models in the *SpaceEx* input format. It allows to compute the reachability in bounded depth as well as reachability till fixed point. 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 [25]. 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, which is called the AGJH algorithm [18]. In addition, a task parallel and an asynchronous variant of AGJH are also implemented in the tool. The tool is available at `https://gitlab.com/raj.ray84/XSpeed-plan`.

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 [10]. It is inspired by a related benchmark in [21]. 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 UB*nn* and BD*nn* (no collisions).

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

Specification The distance between adjacent cars should be positive:

$$x_{\texttt{ldr}} - x > 0,$$

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

For unbounded state space, we have UB*nn*. For bounded state space, we have BD*nn*.

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

Table 1: Computation Times of the Adaptive Cruise Controller.

instance	ACCS05	ACCU05	ACCS06	ACCU06	ACCS07	ACCU07	ACCS08	ACCU08
safety	safe	unsafe	safe	unsafe	safe	unsafe	safe	unsafe
# vars	5	5	6	6	7	7	8	8
# locs	32	32	64	64	128	128	256	256
tool	computation time in [s]							
unbounded spec.	UB05	UB05	UB06	UB06	UB07	UB07	UB08	UB08
PHAVer/SX	9.4	13.7	461	13430	–	–	–	–
PHAVer-lite/SX	1.0	0.9	38.1	22.4	–	–	–	–
PHAVerLite	0.08	0.05	0.50	0.23	4.28	1.38	49.81	7.86
bounded spec. [2]	BD05	BD05	BD06	BD06	BD07	BD07	BD08	BD08
BACH	4.9	0.1	6.4	0.1	11.8	0.2	11.8	0.2

BD*nn* For $i = 0, \ldots, n-1$: $x_i - x_{i+1} > 0$ within default discrete search depth 30.

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 [20], 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 $4 \times (1+n) \times 4^n$ locations, $2n+1$ continuous variables and $2n$ discrete variables. Note that some tools, such as all tools based on 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 UB*nn*.

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

For unbounded state space, we have UB*nn*. For bounded state space, we have BD*nn*.

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

BD*nn* All *nn* sensors are not in location `send` within default discrete search depth 30.

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

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

Table 2: Computation Times of the Distributed Controller.

instance	DISC02	DISC03	DISC04	DISC05
safety	safe	safe	safe	safe
# vars	9	13	17	21
# locs	192	1024	5120	24976
tool	computation time in [s]			
unbounded spec.	UB02	UB03	UB04	UB05
PHAVer/SX	1.1	–	–	–
PHAVer-lite/SX	0.1	548.0	–	–
PHAVerLite	0.04	0.35	2.59	27.99
bounded spec.	BD02	BD03	BD04	BD05
BACH	0.3	0.6	1.1	3.1

Note on PHAVerLite The DISC benchmark is an example where the application of Cartesian factoring techniques makes a significant difference: in the 2019 edition of the competition, where Cartesian factoring was not available, about 78 seconds were spent to solve instance DISC04-UB04, while instance DISC05-UB05 was timing out.

3.3 Fischer's Protocol

Model Fischer's protocol is a time based protocol of mutual exclusion between processes, originally from [22]. The flow constraints are given by $\frac{1}{2} \leq \dot{x}_1 \leq \frac{3}{2}, \ldots, \frac{1}{2} \leq \dot{x}_{\mathtt{m}} \leq \frac{3}{2}$, where $x_{\mathtt{i}}$ is the clock of the $\mathtt{i}$-th process. The product automaton has $(n+1) \times 4^n$ locations and n variables.

FISCSnn protocol with nn processes, considered safe with respect to specification UBnn and BDnn.

FISCUnn protocol with nn processes, considered unsafe with respect to specification UBnn and BDnn.

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

For unbounded state space, we have UBnn. For bounded state space, we have BDnn.

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

BDnn There are no two processes such that both are in location `cs` (critical section) at the same time within default discrete search depth 30.

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

Table 3: Computation Times of the Fischer Benchmark.

instance	FISCS04	FISCU04	FISCS05	FISCU05	FISCS06	FISCU06
safety	safe	unsafe	safe	unsafe	safe	unsafe
# vars	4	4	5	5	6	6
# locs	1280	1280	6144	6144	28672	28672
tool	computation time in [s]					
unbounded spec.	UB04	UB04	UB05	UB05	UB06	UB06
PHAVer/SX	90.5	579	–	–	–	–
PHAVer-lite/SX	12.3	102.2	14722.2	–	–	–
PHAVerLite	0.18	0.17	1.06	0.76	8.83	3.88
bounded spec.	BD04	BD04	BD05	BD05	BD06	BD06
BACH	32.3($B:10$)	0.3	166.3($B:10$)	0.5	655.6($B:10$)	1.1

3.4 Navigation (NAV)

3.4.1 Model

The navigation example is also a single automaton. It models the motion of a point robot in a n-dimensional cube. The cube is partitioned into m^n rectangular regions and each such region is associated with a vector field described by the flow equations. We use a generalization method introduced in [17] to generate such a navigation mode, NAV_m_n. Similar with the motorcade model, in order to generate a not too complex model, we set m as 3 and n as 2, 3, and 4 respectively. As the model is too large to put in the paper, we will omit the graphical presentation here.

3.4.2 Specification

The specification is to check whether there is a behavior of the system which can reach the specific state in the farthest corner. In the benchmark model, Whether $l_{\underbrace{(m-1)\dots(m-1)}_{n}}$ is reachable.

For unbounded state space, we have UBnn. For bounded state space, we have BDnn.

UB_m_n It is never the case that the system is in location $l_{\underbrace{(m-1)\dots(m-1)}_{n}}$.

BD_m_n The system is not in location $l_{\underbrace{(m-1)\dots(m-1)}_{n}}$ within default discrete search depth 30.

3.4.3 Result

Computation Times The computation times of various tools for the NAV benchmark are listed in Tab. 4.

Table 4: Computation Times on the NAV Benchmark

instance	NAV_3_2	NAV_3_3	NAV_3_4
safety	safe	safe	safe
# vars	3	4	5
# locs	9	27	81
tool	computation time in [s]		
unbounded spec.	UB_3_2	UB_3_3	UB_3_4
PHAVerLite	0.01	0.49	0.50
bounded spec.	BD_3_2	BD_3_3	BD_3_4
BACH	0.1	35.8	673.1($B:20$)
XSpeed	67.7	574.28($B:25$)	980.29($B:15$)

Note on PHAVerLite Safety of the unbounded instances is proved using exact reachability for instances UB_3_2 and UB_3_3, while resorting to over-approximation for instance UB_3_4 (in this case, the computation of the exact reachable set diverges).

Note on XSpeed The tool proves the safety of NAV_3_3: BD_3_3 and NAV_3_4: BD_3_4 in a bounded depth of 25 and 15 respectively. Since XSpeed expects flow equations, the flow expressions of the form $x' \in [x_l, x_h]$ is converted to $x' = xc$, where xc is an introduced variable. The constraint $x_l \leq xc \leq x_h$ is set in the initial condition.

3.5 TTEthernet

Model The TTEthernet protocol is a protocol for the remote synchronization of possibly drifted clocks distributed over multiple components, taken from [11]. The system consists of two compression masters (CM) and k synchronization 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.

TTESn protocol with n SM. This model is considered safe with respect to specification UBn and BDn. The global time horizon is limited to 3000 ms.

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

For unbounded state space, we have UBnn. For bounded state space, we have BDnn.

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

BDn For all i, j, $sm_i - sm_j \leq 2max\_drift$ within default discrete search depth 30, where $max\_drift = 0.001$ ms.

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

Table 5: Computation Times of the TTEthernet Benchmark

instance	TTES05	TTES07	TTES09
safety	safe	safe	safe
# vars	9	11	13
# locs	16384	262144	4194304
tool	computation time in [s]		
unbounded spec.	UB05	UB07	UB09
PHAVer/SX	25.2	113	–
PHAVer-lite/SX	1.9	7.7	–
PHAVerLite	0.35	1.40	9.13
bounded spec.	BD05	BD07	BD09
BACH	1.2	1.5	1.8

3.6 Dutch Railway Network

We consider a finite-horizon safety problem over max-plus linear (MPL) systems. An MPL system is described by recurrence equation

$$\mathbf{x}(k+1) = A \otimes \mathbf{x}(k), k = 0, 1, \ldots, \tag{1}$$

where $\mathbf{x}(k) = [x_1(k) \ldots x_n(k)] \in \mathbb{R}^n$ is the state variables representing the time stamps of the discrete events at time horizon k and A is $n \times n$ max-plus algebraic matrix representing model under consideration. It should be noted that the matrix operation on (1) is defined over max-plus algebra: see [5] for more detailed descriptions about max-plus algebra and its operations, and to [2, 3] for more details on formal verification of MPL systems.

Given an MPL system (1), a time horizon N, a set of initial conditions X_0 and an unsafe set S, a finite-horizon safety problem is an instance problem to check whether the system can reach the unsafe set within the given time horizon.

Model In [26, Appendix B], the high-scale Dutch railway networks are modeled as a max-plus-linear(MPL) system. For details on MPL system, please refer to [3] That model has 214 state variables $x_1(k), \ldots, x_{214}(k)$ representing the k-th departure from the selected stations. For ARCH-COMP this year, we use a subset of that model by considering only the first 21 state variables $x_1(k), \ldots, x_{21}(k)$. The model instance is defined formally as follows:

DRNW03 initial condition $X_0 = \{\mathbf{x} \in \mathbb{R}^{21} : 0 \leq x_1 \leq x_2 \leq \ldots \leq x_{21} \leq 1\}$.

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 for bounded state space:

BD01 $\exists k = 0, \ldots, 30$ such that $x_1(k) > x_2(k) > \ldots > x_{21}(k)$ (not satisfied)

Table 6: Computation Times of the Dutch Railway Network Benchmark

instance	DRNW03	DRNW03	DRNW03	DRNW03
safety	safe	unsafe	unsafe	safe
# vars	21	21	21	21
# locs	1	1	1	1
tool	computation time in [s]			
bounded spec.	BD01	BD02	BD03	BD04
TROPICAL	0.28	0.009	0.013	0.28
BACH	0.1	0.3	0.4	366.1($B:20$)
PHAVerLite	0.80	0.29	0.31	0.82
unbounded spec.	UB01	UB02	UB03	UB04
PHAVerLite	11.32	0.27	0.29	–

BD02 $\exists k = 0, \ldots, 30$ such that $x_1(k) - x_2(k) > 20$ or $x_1(k) - x_2(k) < -20$ (satisfied)

BD03 $\exists k = 0, \ldots, 30$ such that $x_9(k) - x_{13}(k) > 40$ or $x_9(k) - x_{13}(k) < -40$ (satisfied)

BD04 $\exists k = 0, \ldots, 30$ such that $x_{17}(k) - x_{21}(k) > 60$ or $x_{17}(k) - x_{21}(k) < -60$ (not satisfied)

For unbounded state space, we have UB01, ..., UB04 by removing the upper bound from $k \in \mathbb{N}$. In the sense of a safety specification, the above specifications specify unsafe states. If the unsafe sets are reachable, the corresponding specification BD01, ..., BD04 is satisfied.

Results The computation times of various tools are listed in Tab. 6.

Note on the model Given an MPL system (1), it is possible that there exist $k_0, c \in \mathbb{N}$, $\lambda \in \mathbb{R}$ such that for all $\mathbf{x}(0) \in \mathbb{R}^n$ the trajectory of (1) starting from $\mathbf{x}(0)$ satisfies

$$x_i(k+c) = (\lambda \times c) + x_i(k),\ i = 1, \ldots, n,\ k \geq k_0.$$

The smallest such k_0 and c are called the transient and the cyclicity of A, respectively. Furthermore, the scalar λ corresponds to the max-plus eigenvalue of the state matrix A. We refer the interested readers to [5, Section 3.7] about this periodic behavior and how to compute transient and cyclicity.

For the underlying DRNW03 model, the corresponding transient and cyclicity is $k_0 = 26$ and $c = 5$, respectively. As a result, $x_i(k+5) - x_j(k+5) = x_i(k) - x_j(k)$ for all $i, j \in \{1, \ldots, 21\}$ and $k \geq 26$. Since, the unsafe sets BD01, ..., BD04 can be expressed as the conjunction or disjunction of inequalities $x_i(k) - x_j(k) > c$ for $c \in \mathbb{R}$, it is suffice to check the reachability problems BD01, ..., BD04 up to time horizon $N = 26 + 5 - 1 = 30$.

Note on PHAVerLite Since the iteration count in PHAVerLite does not guarantee the actual search depth, in the bounded-depth case we added an additional variable (k) to track the number of discrete transitions taken: this is initialized to 0, it is incremented at each discrete transition and it is limited from above by the invariant $k \leq 30$. The tool is then run

until either a fixed point is found (for specifications BD01 and BD04) or a reachable unsafe state is found (for BD02 and BD03). In the lower part of Table 6 we consider *unbounded* variants UB01, ..., UB04 (we also omit the additional variable from the hybrid model). The goal, in this case, is to try and prove/disprove the properties *without* exploiting the cyclicity of the original MPL system. By resorting to over-approximations (starting from a larger initial region and approximating set unions using constraint hulls), PHAVerLite is able to prove that UB01 is not satisfied (i.e., the unsafe region is unreachable); UB02 and UB03 are disproved similarly to BD02 and BD03, using exact reachability (in this case, the missing upper bound for k makes no difference); on the other hand, the tool is not able to prove that UB04 is not satisfied: a reachability analysis using over-approximations yields a false positive, whereas the exact reachability analysis does not terminate.

Note on TROPICAL Instead of translating the MPL system explicitly into an equivalent piecewise-affine system, TROPICAL uses difference logic to express the dynamic of MPL system [24]. By doing so, TROPICAL can solve the safety problem for the *full version* of Dutch Railway Network with 214 variables. The setups for model and specifications are the following.

DRNW03* initial condition $X_0 = \{\mathbf{x} \in \mathbb{R}^{214} : 0 \leq x_1 \leq x_2 \leq \ldots \leq x_{214} \leq 1\}$.

BDR01* $\exists k = 0, \ldots, 88$ such that $x_1(k) > x_2(k) > \ldots > x_{214}(k)$ (not satisfied)

BDR02* $\exists k = 0, \ldots, 88$ such that $x_1(k) - x_2(k) > 20$ or $x_1(k) - x_2(k) < -20$ (satisfied)

BDR03* $\exists k = 0, \ldots, 88$ such that $x_9(k) - x_{13}(k) > 40$ or $x_9(k) - x_{13}(k) < -40$ (satisfied)

BDR04* $\exists k = 0, \ldots, 88$ such that $x_{17}(k) - x_{21}(k) > 60$ or $x_{17}(k) - x_{21}(k) < -60$ (not satisfied)

This new MPL system is also periodic with transient $k_0 = 84$ and cyclicity $c = 5$. Consequently, we only need to check the reachability up to bound $N = 88$ [23]. The computation times are shown in Tab. 7.

Table 7: Computation Times of the Dutch Railway Network Benchmark (full)

instance	DRNW03*	DRNW03*	DRNW03*	DRNW03*
safety	safe	unsafe	unsafe	safe
# vars	214	214	214	214
# locs	1	1	1	1
tool	computation time in [s]			
bounded spec.	BDR01*	BDR02*	BDR03*	BDR04*
TROPICAL	235.31	10.02	17.90	219.51

4 Conclusions and Outlook

This report presents the results of the fourth edition of a friendly competition for the formal verification of continuous and hybrid systems of the ARCH'20 workshop, in the category on PCDB: piecewise constant dynamics and BMC of such systems. The reports of other categories can be found in the proceedings and on the ARCH website: cps-vo.org/group/ARCH. The code with which the results have been obtained is publicly available at gitlab.com/goranf/ARCH-COMP.

As this new PCDB category includes both unbounded checkers and bounded checkers of HPCD systems, we provide both unbounded and bounded specifications for each benchmark respectively. Then, each participator can choose to solve the corresponding specifications they are particularly suited for. In the spirit of a friendly competition, this report does not provide any ranking of tools. We report a few casual observations. For the reported instances, PHAVerLite and BACH solved almost all the cases in the respective category of unbounded and bounded problem efficiently. Meanwhile, TROPICAL scales very well on reachability and safety analysis of max-plus-linear systems.

5 Acknowledgments

Lei Bu and Yuming Wu are supported by the National Natural Science Foundation of China (No.61572249). Rajarshi Ray was supported by the Science and Engineering Research Board (SERB) project with File No. IMP/2018/000523.

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. Computational techniques for reachability analysis of max-plus-linear systems. *Automatica*, 53(0):293–302, 2015.

[3] D. Adzkiya, B. De Schutter, and A. Abate. Finite bisimulations of max-plus-linear systems. *IEEE Transactions on Automatic Control*, 58(12):3039–3054, 2012.

[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] François Baccelli, Guy Cohen, Geert Jan Olsder, and Jean-Pierre Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley & Sons Ltd, 1992.

[6] 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.

[7] A. Becchi and E. Zaffanella. A direct encoding for NNC polyhedra. In H. Chockler and G. Weissenbacher, editors, *Computer Aided Verification*, pages 230–248, Cham, 2018. Springer International Publishing.

[8] A. Becchi and E. Zaffanella. An efficient abstract domain for not necessarily closed polyhedra. In A. Podelski, editor, *Static Analysis - 25th International Symposium, SAS 2018, Freiburg, Germany, August 29-31, 2018, Proceedings*, volume 11002 of *Lecture Notes in Computer Science*, pages 146–165. Springer, 2018.

[9] A. Becchi and E. Zaffanella. Revisiting polyhedral analysis for hybrid systems. In Bor-Yuh Evan Chang, editor, *Static Analysis - 26th International Symposium, SAS 2019, Porto, Portugal, October 8-11, 2019, Proceedings*, volume 11822 of *Lecture Notes in Computer Science*, pages 183–202. Springer, 2019.

[10] 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.

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

[12] 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.

[13] 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.

[14] Scott Cotton, Eugene Asarin, Oded Maler, and Peter Niebert. Some progress in satisfiability checking for difference logic. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 263–276. Springer, 2004.

[15] 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.

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

[17] Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni, editors. *Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, volume 7737 of *Lecture Notes in Computer Science*. Springer, 2013.

[18] Amit Gurung, Arup Deka, Ezio Bartocci, Sergiy Bogomolov, Radu Grosu, and Rajarshi Ray. Parallel reachability analysis for hybrid systems. In *2016 ACM/IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE 2016, Kanpur, India, November 18-20, 2016*, pages 12–22. IEEE, 2016.

[19] N. Halbwachs, D. Merchat, and L. Gonnord. Some ways to reduce the space dimension in polyhedra computations. *Formal Methods in System Design*, 29(1):79–95, 2006.

[20] 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.

[21] 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.

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

[23] M.S. Mufid, D. Adzkiya, and A. Abate. Bounded model checking of max-plus linear systems via predicate abstractions. In *Proceedings of FORMATS, LNCS 11750*, pages 142–159, 2019.

[24] M.S. Mufid, D. Adzkiya, and A. Abate. Symbolic reachability analysis of high-dimensional max-plus linear systems. In *ArXiv: `https://arxiv.org/abs/2007.04510`. To appear in the Proceedings of WODES*, 2020.

[25] Rajarshi Ray, Amit Gurung, Binayak Das, Ezio Bartocci, Sergiy Bogomolov, and Radu Grosu. Xspeed: Accelerating reachability analysis on multi-core processors. In Nir Piterman, editor, *Hardware and Software: Verification and Testing - 11th International Haifa Verification Conference, HVC 2015, Haifa, Israel, November 17-19, 2015, Proceedings*, volume 9434 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.

[26] 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.

[27] 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.

[28] 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 Implementation Languages and Used Machines

A.1 BACH

- Implementation language: C++
- 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 PHAVer/SX

- Implementation language: C++
- 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.3 PHAVerLite (and PHAVer-lite/SX)

- Implementation language: C++
- 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.4 TROPICAL

- Implementation language: C++
- Processor: Intel Xeon CPU E5-1660 v3 @ 3.30GHz x 16
- Memory: 16 GB
- Average CPU Mark on www.cpubenchmark.net: 8388 (full), 1752 (single thread)

A.5 XSpeed

- Implementation language: C++
- Processor: Intel(R) Core(TM) i3-9100F CPU @ 3.60GHz x 4
- Memory: 8 GB
- Average CPU Mark on www.cpubenchmark.net: 6837 (full), 2519 (single thread)