EPiC Series in Computing

Volume 103, 2024, Pages 153–162

Proceedings of the 11th Int. Workshop on Applied Verification for Continuous and Hybrid Systems

Randomized Generation of Arbitrarily Difficult Verification Benchmarks for Linear Time-Invariant Systems

Mark Wetzlinger and Matthias Althoff

Technische Universität München, Boltzmannstr. 3, 85748 Garching b. München, Germany
{m.wetzlinger,althoff}@tum.de

Abstract

Benchmark proposal: The verification of uncertain linear systems is a fundamental building block for the analysis of complex cyber-physical systems. While there exist many advanced tools for numerical analysis, their evaluation is to date limited by selected benchmarks. To better understand the strengths and weaknesses of formal verification algorithms for linear systems, we propose a randomized generation of verification benchmarks in this paper. To this end, we leverage a reachability algorithm that can compute reachable sets with arbitrary precision. By placing unsafe sets exactly at the boundary of the computed outer and inner approximations, we are able to generate random verification tasks of arbitrary difficulty by changing the precision of the reachability analysis. We validate our approach by verifying and falsifying increasingly complex generated benchmarks using a state-of-the-art verification algorithm.

1 Introduction

Formal verification of reach-avoid problems proves whether a dynamical system, subject to uncertainties in the initial state, the input, and potentially parameters, can avoid unsafe sets over a given time horizon. To this end, a wide variety of different approaches have been developed, e.g., simulation-based techniques [1, 2, 3], set-propagation methods [4, 5], barrier certificates [6], reformulations to optimization problems [7, 8], and theorem-proving approaches [9]. The annual ARCH competition provides a level playing field for comparing these approaches over various categories of different system classes. Tools such as SpaceEx [10], CORA [11], JuliaReach [12], HyDRA [13], C2E2 [14], HyLAA [15], and XSpeed [16] have participated in recent editions of the linear systems category [17], at which this benchmark proposal is addressed.

Currently, analysis tools are evaluated on selected state-of-the-art benchmarks. While this is beneficial regarding the comparability between approaches, it incentivizes tool developers to tune their approaches toward specific benchmarks. However, this casts doubt on the generalization of the proposed approaches to larger audiences and industrial practitioners, as expert knowledge is required to obtain optimal results. In fact, many current approaches depend on the tuning of algorithm parameters, which heavily influence the result[1]—bad parameter tuning can lead to spurious counterexamples, and, consequently, unsuccessful verification.

In some cases, manual tuning might not even be possible, as linear systems may be generated automatically or on-the-fly, e.g., due to simplifications of more complex system dynamics. Also, the system dynamics may change during operation due to unforeseen events, such as hardware failures. In this case, an algorithm that can adapt to different dynamics is preferable over manually-tuned counterparts. In summary, an enhanced generalization of current approaches and increased adaptability constitute our core motivation for the benchmark proposal presented in this paper.

In Section 2, we introduce the notation and fundamentals about linear systems and reachability analysis to formulate the verification task. Next, we show how to randomly generate verification benchmarks in Section 3, where we first generate a random linear system, then compute the reachable set, and finally place the unsafe sets, resulting in verification benchmarks that are verifiable or falsifiable by construction. In Section 4, we showcase the entire procedure on two selected randomly generated benchmarks and validate the correct generation over increasingly complex benchmarks using a state-of-the-art verification algorithm. We draw conclusions in Section 5.

2 Preliminaries

2.1 Notation

The nonnegative natural, positive natural, real, positive real, and complex numbers are denoted by $\mathbb{N}$, $\mathbb{N}_+$, $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{C}$, respectively. The real and imaginary part of a complex number $z = a + bi \in \mathbb{C}$ are returned by the operators $\Re(z) = a$ and $\Im(z) = b$. The floor operator $\lfloor x \rfloor$ returns the next lower integer number of $x \in \mathbb{R}$. Given a vector $v \in \mathbb{R}^n$, $v_{(i)}$ denotes the ith entry. For a matrix $M \in \mathbb{R}^{n_1 \times n_2}$, $M_{(i,\cdot)}$ returns the ith row and $M_{(\cdot,j)}$ returns the jth column. The eigenvalues of a square matrix $M \in \mathbb{R}^{n \times n}$ are denoted by $\lambda_1(M), \dots, \lambda_n(M) \in \mathbb{C}$. For two matrices M_1, M_2, $\texttt{diag}(M_1, M_2)$ returns a blockdiagonal matrix with the matrices concatenated diagonally.

Sets are denoted by calligraphic letters $\mathcal{S} \subseteq \mathbb{R}^n$. For a matrix $M \in \mathbb{R}^{n \times n}$ and a set $\mathcal{S} \subset \mathbb{R}^n$, the linear map is defined by $M\mathcal{S} \coloneqq \{Ms | s \in \mathcal{S}\}$. The operation $\texttt{cen}(\mathcal{S})$ returns the center of a set $\mathcal{S}$. The support function in a direction $\ell \in \mathbb{R}^n$ is defined by $\rho(\mathcal{S}, \ell) = \max_{x \in \mathcal{S}} \ell^\top x$ and the corresponding support vector is defined by $\nu(\mathcal{S}, \ell) = \arg\max_{x \in \mathcal{S}} \ell^\top x$. For two sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^n$, their Minkowski sum is defined by $\mathcal{S}_1 \oplus \mathcal{S}_2 \coloneqq \{s_1 + s_2 | s_1 \in \mathcal{S}_1, s_2 \in \mathcal{S}_2\}$, their Minkowski difference is defined by $\mathcal{S}_1 \ominus \mathcal{S}_2 \coloneqq \{s | s \oplus \mathcal{S}_2 \in \mathcal{S}_1\}$, and the Hausdorff distance between them by

$$d_H(\mathcal{S}_1, \mathcal{S}_2) = \max\left\{ \max_{s_1 \in \mathcal{S}_1} \left(\min_{s_2 \in \mathcal{S}_2} \|s_1 - s_2\|_2 \right), \max_{s_2 \in \mathcal{S}_2} \left(\min_{s_1 \in \mathcal{S}_1} \|s_1 - s_2\|_2 \right) \right\}. \tag{1}$$

[1] The ARCH reports for the various systems classes provide many examples for the intricate tuning efforts required to solve certain benchmarks.

Using a hyperball $\mathcal{B}_\varepsilon$ of radius ε, an alternative definition is

$$d_H(\mathcal{S}_1, \mathcal{S}_2) = \varepsilon \quad \Leftrightarrow \quad \mathcal{S}_2 \subseteq \mathcal{S}_1 \oplus \mathcal{B}_\varepsilon \wedge \mathcal{S}_1 \subseteq \mathcal{S}_2 \oplus \mathcal{B}_\varepsilon. \tag{2}$$

An interval $\mathcal{I} \subset \mathbb{R}^n$ is written as $[a, b] = \{x \in \mathbb{R}^n | a \leq x \leq b\}$, where the inequality holds elementwise.

Let $\mathtt{P}_\mathcal{I}$ be a probability distribution defined on the domain $\mathcal{I} \subseteq \mathbb{R}$, where taking a random value x is written as $x \sim \mathtt{P}_\mathcal{I}$. The uniform distribution of all integer numbers within the interval $\mathcal{I}$ is represented by $\mathtt{U}_\mathcal{I}$.

2.2 Linear Systems, Reachable Sets, and Verification Tasks

We consider linear time-invariant systems of the form

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3a}$$
$$y(t) = Cx(t) + v(t), \tag{3b}$$

where $x \in \mathbb{R}^n$ is the state vector, $A \in \mathbb{R}^{n \times n}$ is the state matrix, $u \in \mathbb{R}^m$ is the input vector, $B \in \mathbb{R}^{n \times m}$ is the input matrix, $y \in \mathbb{R}^r$ is the output vector, $C \in \mathbb{R}^{r \times n}$ is the output matrix, and $v \in \mathbb{R}^r$ is the sensor noise. To analyze a linear system under the influence of uncertainties, we compute the reachable set and the output set as defined subsequently.

Definition 1 (Reachable set and output set)**.** *For a given initial state $x(0)$ and an input trajectory $u(\cdot)$, let us denote the solution of (3a) at time t by $\xi(t; x(0), u(\cdot))$. The reachable set at time $t \geq 0$ for all initial states within the initial set $\mathcal{X}_0 \subset \mathbb{R}^n$ and all input trajectories bounded by the input set $\mathcal{U} \subset \mathbb{R}^m$ is*

$$\mathcal{R}(t) \coloneqq \{\xi(t; x(0), u(\cdot)) \,|\, x(0) \in \mathcal{X}_0, \forall \theta \in [0, t] \colon u(\theta) \in \mathcal{U}\}. \tag{4}$$

We write $\mathcal{R}(t_k)$ for the time-point reachable set at time t_k and $\mathcal{R}(\tau_k)$ for the time-interval reachable set for $t \in \tau_k = [t_k, t_{k+1}]$. The output set follows from inserting (4) in (3b), i.e.,

$$\mathcal{Y}(t) \coloneqq \{C\xi(t; x(0), u(\cdot)) + v(t) \,|\, x(0) \in \mathcal{X}_0, \forall \theta \in [0, t] \colon u(\theta) \in \mathcal{U}, v(\theta) \in \mathcal{V}\} = C\mathcal{R}(t) \oplus \mathcal{V}, \tag{5}$$

where the sensor noise v is bounded by $\mathcal{V} \subset \mathbb{R}^r$. Analogously to the reachable set, we write $\mathcal{Y}(t_k)$ and $\mathcal{Y}(\tau_k)$ for the time-point output set and time-interval output set, respectively.

Please note that the reachable set and output set over longer time horizons $[0, t_\text{end}]$ are given by the unions $\mathcal{R}([0, t_\text{end}]) = \bigcup_{k=0}^{\omega-1} \mathcal{R}(\tau_k)$, $\mathcal{Y}([0, t_\text{end}]) = \bigcup_{k=0}^{\omega-1} \mathcal{Y}(\tau_k)$, where $\omega \in \mathbb{N}_+$ is the number of time steps.

For further use, let us refer to the tuple $\Theta = (t_\text{end}, \mathcal{X}_0, \mathcal{U}, \mathcal{V})$ as the *model parameters*. It is well known that the reachable set and, consequently, the output set can only be computed exactly for special system classes [18]. In the context of safety verification, we thus compute outer approximations $\widehat{\mathcal{Y}}(t) \supseteq \mathcal{Y}(t)$ to verify safety and inner approximations $\check{\mathcal{Y}}(t) \subseteq \mathcal{Y}(t)$ to falsify safety. Recently, an automated reachability algorithm has been developed that, given the model parameters Θ, returns arbitrarily tight outer approximations [19], i.e.,

$$\forall t \in [0, t_\text{end}] \colon d_\text{H}\big(\mathcal{Y}(t), \widehat{\mathcal{Y}}(t)\big) \leq \varepsilon, \tag{6}$$

where $\varepsilon \in \mathbb{R}, \varepsilon > 0$ is a user-defined value for the admissible error. Using (2), we obtain

$$\check{\mathcal{Y}}(t) = \widehat{\mathcal{Y}}(t) \ominus \mathcal{B}_\varepsilon \subseteq \mathcal{Y}(t), \tag{7}$$

which we will use for the generation of falsifiable benchmarks. Finally, let us formally introduce the type of verification task considered in this work.

Algorithm 1 Random generation of linear systems

Require: State dimension n, input dimension m, output dimension r, interval $\mathcal{I}_{\text{real}}$ bounding $\Re(\lambda(A))$, interval $\mathcal{I}_{\text{imag}}$ bounding $\Im(\lambda(A))$, random distribution $\mathtt{P}$

Ensure: State matrix A, input matrix B, output matrix C

1: $\lambda_{\text{cc}} \sim \mathtt{U}_{[0,\lfloor n/2 \rfloor]}$, $\lambda_{\text{real}} \leftarrow n - \lambda_{\text{cc}}$
2: $\forall i \in \{1, \dots, \lambda_{\text{real}}\}\colon A_{\text{real}(i,i)} \sim \mathtt{P}_{\mathcal{I}_{\text{real}}}$
3: $\forall i \in \{1, 3, \dots, 2\lambda_{\text{cc}} - 1\}\colon \tilde{\lambda} \sim \mathtt{P}_{\mathcal{I}_{\text{real}}}, A_{\text{cc}(i,i)} \leftarrow \tilde{\lambda}, A_{\text{cc}(i+1,i+1)} \leftarrow \tilde{\lambda}$
4: $\forall i \in \{1, 3, \dots, 2\lambda_{\text{cc}} - 1\}\colon \tilde{\lambda} \sim \mathtt{P}_{\mathcal{I}_{\text{imag}}}, A_{\text{cc}(i+1,i)} \leftarrow \tilde{\lambda}, A_{\text{cc}(i,i+1)} \leftarrow -\tilde{\lambda}$
5: $\forall i \in \{1, \dots, n\}\ \forall j \in \{1, \dots, n\}\colon Q_{(i,j)} \sim \mathtt{P}_{[-\infty,\infty]}$, $A \leftarrow Q\,\mathtt{diag}(A_{\text{real}}, A_{\text{cc}})\,Q^{-1}$
6: $\forall i \in \{1, \dots, n\}\ \forall j \in \{1, \dots, m\}\colon B_{(i,j)} \sim \mathtt{P}_{[-\infty,\infty]}$
7: $\forall i \in \{1, \dots, r\}\ \forall j \in \{1, \dots, n\}\colon C_{(i,j)} \sim \mathtt{P}_{[-\infty,\infty]}$
8: **return** A, B, C

Problem 1 (Verification task)**.** *Given a linear system of the form (3), the model parameters $\Theta = (t_{end}, \mathcal{X}_0, \mathcal{U}, \mathcal{V})$ of proper dimensions, and a set of unsafe sets $\mathcal{L}_1, \dots, \mathcal{L}_w \subset \mathbb{R}^r$, decide whether*

$$\forall t \in [0, t_{end}]\colon \mathcal{Y}(t) \cap \bigcup_{j=1}^{w} \mathcal{L}_j = \emptyset, \tag{8}$$

that is, whether the output set $\mathcal{Y}(t)$ intersects any unsafe set $\mathcal{L}_1, \dots, \mathcal{L}_w$ at any point in time.

3 Randomized Generation of Verification Benchmarks

In this section, we show how to generate random verification benchmarks for linear systems using a three-step process: First, we generate a random linear system; second, we compute an outer approximation of the reachable set; and third, we place unsafe sets that are known to be verifiable or falsifiable.

3.1 Randomized Generation of Linear Systems

The randomized generation of a linear system, determined by its state matrix A, input matrix B, and output matrix C, is summarized in Algorithm 1. The required input arguments are the state dimension $n \in \mathbb{N}_+$, the number of inputs $m \in \mathbb{N}$, the number of outputs $r \in \mathbb{N}_+$, as well as the bounds $\mathcal{I}_{\text{real}} \subset \mathbb{R}$, $\mathcal{I}_{\text{imag}} \subset \mathbb{R}$ for the real and imaginary parts of the eigenvalues of A, respectively, and a distribution $\mathtt{P}$ for sampling from the respective domains. Please note that the interval $\mathcal{I}_{\text{imag}}$ needs to be symmetric around zero. After determining the number of purely real eigenvalues and complex-conjugate eigenvalues, we place all real eigenvalues on the diagonal of the matrix A_{real} and the complex-conjugate eigenvalues $a \pm bi \in \mathbb{C}$ into the matrix A_{cc} as blocks of the form

$$\begin{bmatrix} a & -b \\ b & a \end{bmatrix}.$$

Consequently, we construct the blockdiagonal matrix $\mathtt{diag}(A_{\text{real}}, A_{\text{cc}}) \in \mathbb{R}^{n \times n}$ and rotate it by a random matrix, see line 5. Finally, in lines 6 and 7, we sample the entries of the input matrix B and the output matrix C. Please note that one can also choose different distributions for each sampling procedure in lines 2-7.

3.2 Computation of Reachable Sets

After randomly generating a linear system using Algorithm 1, we require to set the model parameters Θ. For the initial set $\mathcal{X}_0 \subset \mathbb{R}^n$, the input set $\mathcal{U} \subset \mathbb{R}^m$, and the set $\mathcal{V} \subset \mathbb{R}^r$, we may use a variety of different set representations, e.g., intervals, zonotopes, ellipsoids, or polytopes. For the center $c \in \mathbb{R}^n$, we sample from a given distribution $\mathtt{P}_{[-\infty,\infty]}$ in each coordinate, i.e.,

$$\forall i \in \{1, \dots, n\} \colon c_{(i)} \sim \mathtt{P}_{[-\infty,\infty]}.$$

Let us briefly showcase the random generation for intervals $\mathcal{I} \subset \mathbb{R}^n$ and zonotopes $\mathcal{Z} \subset \mathbb{R}^n$. For intervals, we randomly sample the width of each dimension:

$$\mathcal{I} = c \oplus [-a, a], \quad \forall i \in \{1, \dots, n\} \colon a_{(i)} \sim \mathtt{P}_{[-1,1]}. \tag{9}$$

For zonotopes, we sample each individual entry of the generator matrix $G \in \mathbb{R}^{n \times \gamma}$ and scale the resulting generators to another randomly sampled length:

$$\mathcal{Z} = \left\{ c + \sum_{j=1}^{\gamma} G_{(\cdot,j)} \beta_{(j)} \,\middle|\, \|\beta\|_\infty \leq 1 \right\}, \quad \forall j \in \{1, \dots, \gamma\} \colon G_{(\cdot,j)} \sim \mathtt{P}_{\mathcal{S}_n}, \tag{10}$$

where $\mathtt{P}_{\mathcal{S}_n}$ is a distribution over the n-dimensional unit sphere $\mathcal{S}_n \coloneqq \{x \in \mathbb{R}^n \,|\, \|x\|_2 = 1\}$. Similar ideas can be used to generate random ellipsoids and polytopes.

We determine the time horizon t_{end} using the largest real part $\lambda_{\max}(A) \in \mathbb{R}$ of all eigenvalues of the state matrix A and the convergence criterion

$$e^{\lambda_{\max}(A) t_{\text{end}}} = \eta \quad \Rightarrow \quad t_{\text{end}} = \frac{\ln \eta}{\lambda_{\max}(A)}, \tag{11}$$

where $\eta \in \mathbb{R}, 0 < \eta \ll 1$. Note that this requires that $\lambda_{\max}(A) < 0$, which is a sensible requirement in practice to ensure stability.

Given the randomly generated set of model parameters $\Theta = (t_{\text{end}}, \mathcal{X}_0, \mathcal{U}, \mathcal{V})$, we use the algorithm from [19, Alg. 2] with tightness $\varepsilon \in \mathbb{R}_+$ to compute an outer approximation of the reachable set $\widehat{\mathcal{Y}}([0, t_{\text{end}}]) \supseteq \mathcal{Y}([0, t_{\text{end}}])$.

3.3 Placement of Unsafe Sets

For the purpose of benchmark generation, unsafe sets can be categorized into two groups depending on their satisfiability. To obtain a satisfiable safety specification, one must place the unsafe set such that it does not intersect the computed outer approximation. In contrast, an unsafe set that intersects an inner approximation of the reachable set constitutes an unsatisfiable safety specification. Our main idea consists in generating sets, e.g., using (9)-(10) and re-positioning the center afterward.

The procedure is illustrated in Figure 1: Given a random direction $\ell \in \mathbb{R}^r, \|\ell\|_2 = 1$, we compute the support vector $\widehat{\eta}(\ell)$ associated to the support function of the computed outer approximation $\widehat{\mathcal{Y}}([0, t_{\text{end}}])$, i.e.,

$$\widehat{y}(\ell) = \rho\big(\widehat{\mathcal{Y}}([0, t_{\text{end}}]), \ell\big) = \max_{k \in \{0, \dots, \omega - 1\}} \rho\big(\widehat{\mathcal{Y}}(\tau_k), \ell\big).$$

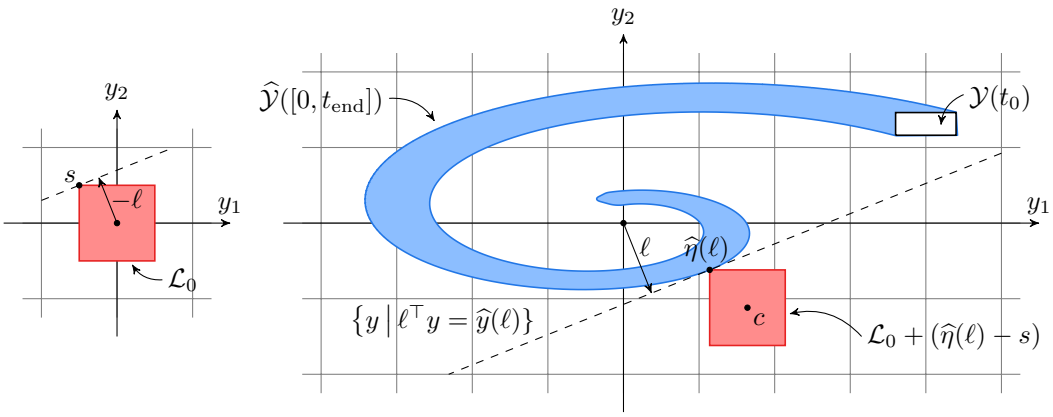

Figure 1: The support vectors of the zero-centered unsafe set $\mathcal{L}_0$ and the output set $\widehat{\mathcal{Y}}([0, t_{\text{end}}])$ are used for placing unsafe sets.

Next, we compute the support vector $s = \nu(\mathcal{L}_0, -\ell)$ of the randomly generated set $\mathcal{L}_0$ centered at the origin in the opposite direction $-\ell$. Shifting the center of $\mathcal{L}_0$ to $(\widehat{\eta}(\ell) - s)$ then yields a satisfiable specification, as the hyperplane $\{y \mid \ell^\top y = \widehat{y}(\ell)\}$ separates the outer approximation $\widehat{\mathcal{Y}}([0, t_{\text{end}}])$ from the translated unsafe set:

$$\mathcal{Y}([0, t_{\text{end}}]) \cap \mathcal{L} = \emptyset$$

with $\mathcal{L} \coloneqq \mathcal{L}_0 + (\widehat{\eta}(\ell) - s)$.

To obtain an unsatisfiable specification, we re-use the computed outer approximation $\widehat{\mathcal{Y}}([0, t_{\text{end}}])$ and the unsafe set $\mathcal{L}$ from above. Enlarging the unsafe set $\mathcal{L}$ by a ball $\mathcal{B}_\varepsilon$ with ε as in (6) yields an unsatisfiable specification:

$$\widehat{\mathcal{Y}}([0, t_{\text{end}}]) \cap \mathcal{L} \neq \emptyset \overset{(6)}{\Rightarrow} \mathcal{Y}([0, t_{\text{end}}]) \cap \big(\mathcal{L} \oplus \mathcal{B}_\varepsilon\big) \neq \emptyset.$$

Hence, we repeat the procedure from above and then enlarge the obtained unsafe set $\mathcal{L}$ by the error ball $\mathcal{B}_\varepsilon$ to obtain an unsatisfiable specification.

The verification benchmarks can be made arbitrarily difficult via the tuning of the maximum Hausdorff distance ε in (6) between the exact output set and the computed outer and inner approximations. Obviously, the closer a specification is placed to the exact output set, the more difficult it is to obtain a correct answer to Problem 1. Since the reachability algorithm in [19, Alg. 2] is guaranteed to return a solution that respects the given Hausdorff distance ε, one can simply increase the difficulty by tuning this value closer to 0.

4 Numerical Evaluation

In this section, we evaluate the proposed random generation of verification benchmarks: First, we show two low-dimensional cases with satisfiable and unsatisfiable specifications, respectively. Then, we let a state-of-the-art verification algorithm solve a large number of randomly generated verification benchmarks with increasing complexity in order to validate our proposed procedure.

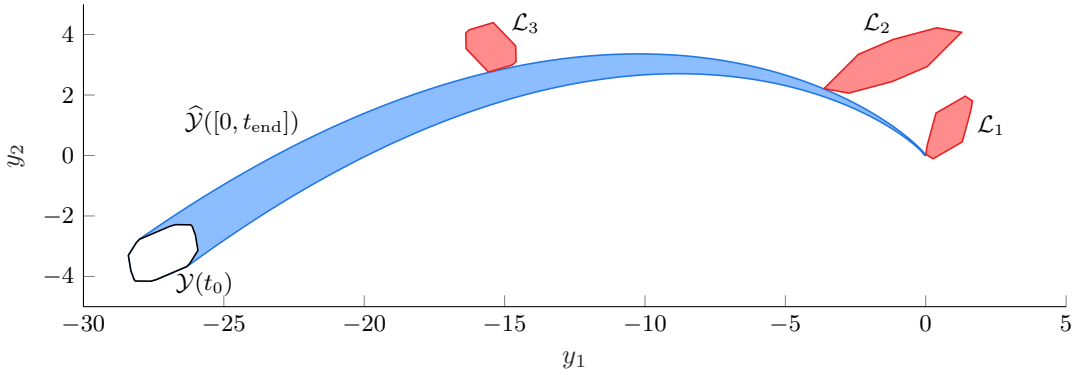

Figure 2: Verifiable verification benchmark: Initial output set $\mathcal{Y}(t_0) = C\mathcal{X}_0$, unsafe sets $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$, and outer approximation of the output set $\widehat{\mathcal{Y}}([0, t_{\text{end}}])$.

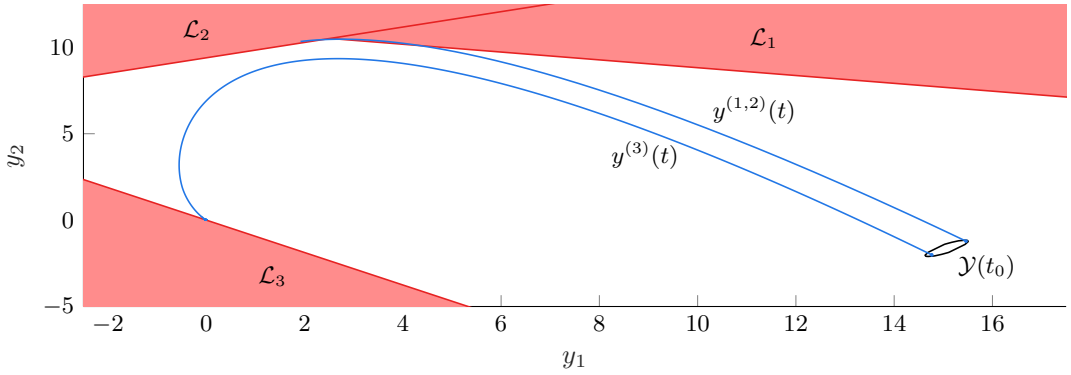

Figure 3: Falsifiable verification benchmark: Initial output set $\mathcal{Y}(t_0) = C\mathcal{X}_0$, unsafe halfspaces $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$, and falsifying trajectories $y^{(1)}(t), y^{(2)}(t), y^{(3)}(t)$.

4.1 Illustrative Examples

For both illustrative examples, we set the state dimension to $n = 5$, the input dimension to $m = 1$, and the output dimension to $r = 2$. For the real and imaginary parts of the eigenvalues, we use the intervals $\mathcal{I}_{\text{real}} = [-5, -1]$ and $\mathcal{I}_{\text{imag}} = [-0.5, 0.5]$. We use a zonotope to represent the initial set $\mathcal{X}_0$ and an interval to represent the input set $\mathcal{U}$. The set $\mathcal{V}$ is only the origin since it does not affect the difficulty of the generated verification benchmark. In both cases, we bound the Hausdorff distance in (6) by $\varepsilon = 0.01$. The generated benchmarks can be obtained in the CORA version v2024.3.0[2] using the file `example_linear_verify_randomGeneration.m`.

Figure 2 shows a verifiable benchmark. The verification algorithm [19] computes an outer approximation of the output set $\widehat{\mathcal{Y}}([0, t_{\text{end}}])$ in $\omega = 233$ time steps, with time step sizes ranging between $\Delta t = 0.0019$ and $\Delta t = 0.1243$. The computed output set $\widehat{\mathcal{Y}}([0, t_{\text{end}}])$ successfully avoids all three unsafe sets, thereby verifying safety for this instance of Problem 1.

[2] Available at https://github.com/TUMcps/CORA.

In contrast, Figure 3 shows a benchmark that cannot be verified. The verification algorithm in [20] proves that there are initial states $x(0) \in \mathcal{X}_0$ with input trajectories $u(t)$, for which the resulting output trajectory $y(t)$ reaches each unsafe set. Since we have at least one violation, safety is falsified for the given instance of Problem 1.

4.2 Scalability

Finally, we validate our randomized generation of verification benchmarks for increasingly complex scenarios. To this end, we vary the state dimension from 5 to 100 and the number of inputs and outputs from 1 to 10. Furthermore, we restrict ourselves to 1 to 10 unsafe sets represented as halfspaces, as they constitute one of the most common types of safety specifications. We alternate between benchmarks with satisfiable and unsatisfiable specifications. The difficulty of the generated benchmarks is determined by the Hausdorff distance ε in (6), which we set relative to the size of the initial set $\mathcal{X}_0$ using the factor $\mu > 0$ in Table 1:

$$\varepsilon = \mu \left\| \sum_{i=1}^{n} d_{(i)} \right\|_2 \tag{12}$$

where $d = b - a \in \mathbb{R}^n$ is the diameter of the interval hull $[a, b] \supseteq \mathcal{S}$.

Overall, the generation time increases with the dimension of the linear system, as it is increasingly time-consuming to compute outer approximations with a small Hausdorff distance ε to the exact output set. This is also why we increased the factor μ for state dimensions 50 and 100. Still, our approach can generate large-scale verification benchmarks within less than a second on average. In general, the computation time grows as the value ε in (6) decreases.

To show that our proposed generation indeed returns verification benchmarks that are verifiable or falsifiable as requested, we feed them to the verification algorithm from [20]. In all cases, the correct result is returned. The falsifiable benchmarks are solved a bit faster than the verifiable benchmarks, as their analysis can be aborted as soon as an unsafe set is entered.

5 Conclusion

We propose an approach to generate random verification benchmarks for linear time-invariant systems that are verifiable or falsifiable by construction. After generating a random linear system from user inputs, such as the state dimension or ranges for the eigenvalues of the state matrix, we compute the reachable set of known tightness with respect to the exact reachable set. This allows us to place unsafe sets that are known to be avoidable or unavoidable. The proposed generation of verification benchmarks can be used by current state-of-the-art tools for formal verification to identify their strengths and weaknesses, as well as to test automated approaches for reachability and verification alike.

6 Acknowledgments

The authors gratefully acknowledge financial support by the project justITSELF funded by the European Research Council (ERC) under grant agreement No 817629.

Table 1: Increasingly complex randomly generated verification benchmarks solved using the approach in [20]. Computation times on average over 5 randomly generated benchmarks.

Dynamics			Benchmark generation				Approach in [20]
n	m	r	w	μ	Satisfiable?	Generation time	Verification time
5	1	1	1	0.1	Yes	0.0856 s	0.0116 s
5	1	1	1	0.1	No	0.1899 s	0.0092 s
5	1	1	1	0.01	Yes	0.4387 s	0.0279 s
5	1	1	1	0.01	No	0.3748 s	0.0085 s
10	2	2	3	0.1	Yes	0.0809 s	0.0133 s
10	2	2	3	0.1	No	0.5151 s	0.0108 s
10	2	2	3	0.01	Yes	0.3461 s	0.0210 s
10	2	2	3	0.01	No	0.4105 s	0.0106 s
50	5	5	5	0.25	Yes	0.1205 s	0.0182 s
50	5	5	5	0.25	No	0.3968 s	0.0114 s
50	5	5	5	0.05	Yes	0.8761 s	0.0778 s
50	5	5	5	0.05	No	0.2467 s	0.0104 s
100	10	10	10	0.25	Yes	0.7931 s	0.0707 s
100	10	10	10	0.25	No	0.3862 s	0.0139 s
100	10	10	10	0.05	Yes	0.4708 s	0.0523 s
100	10	10	10	0.05	No	0.9049 s	0.0134 s

References

[1] A. Donzé and O. Maler. Systematic simulation using sensitivity analysis. In *Proc. of the 10th International Workshop on Hybrid Systems: Computation and Control*, pages 174–189. Springer, 2007. DOI: 10.1007/978-3-540-71493-4_16.

[2] T. Dang, A. Donzé, O. Maler, and N. Shalev. Sensitive state-space exploration. In *Proc. of the 47th Conference on Decision and Control*, pages 4049–4054. IEEE, 2008. DOI: 10.1109/CDC.2008.4739371.

[3] P. S. Duggirala and M. Viswanathan. Parsimonious, simulation based verification of linear systems. In *Proc. of the 28th International Conference on Computer Aided Verification*, pages 477–494. Springer, 2016. DOI: 10.1007/978-3-319-41528-4_26.

[4] E. Goubault and S. Putot. Inner and outer reachability for the verification of control systems. In *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control*, pages 11–22. ACM, 2019. DOI: 10.1145/3302504.3311794.

[5] M. Althoff, G. Frehse, and A. Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):369–395, 2021. DOI: 10.1146/annurev-control-071420-081941.

[6] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Proc. of the 7th International Workshop on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004. DOI: 10.1007/978-3-540-24743-2_32.

[7] H. Yin, A. Packard, M. Arcak, and P. Seiler. Reachability analysis using dissipation inequalities for uncertain nonlinear systems. *Systems and Control Letters*, 142:104736, 2020. DOI: 10.1016/j.sysconle.2020.104736.

[8] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi reachability: A brief overview and recent advances. In *Proc. of the 56th Conference on Decision and Control*, pages 2242–2253. IEEE, 2017. DOI: 10.1109/CDC.2017.8263977.

[9] A. Platzer. *Logical analysis of hybrid systems: Proving theorems for complex dynamics*. Springer, 2010. DOI: 10.1007/978-3-642-14509-4.

[10] 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. DOI: 10.1007/978-3-642-22110-1_30.

[11] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015. DOI: 10.29007/zbkv.

[12] S. Bogomolov, M. Forets, G. Frehse, K. Potomkin, and C. Schilling. JuliaReach: A toolbox for set-based reachability. In *Proc. of the 22nd International Conference on Hybrid Systems: Computation and Control*, pages 39–44. ACM, 2019. DOI: 10.1145/3302504.3311804.

[13] S. Schupp and E. Ábrahám. The HyDRA tool–a playground for the development of hybrid systems reachability analysis methods. In *Proc. of the PhD Symposium at iFM18*, pages 22–23, 2018.

[14] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2E2: A verification tool for stateflow models. In *Proc. of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 68–82. Springer, 2015. DOI: 10.1007/978-3-662-46681-0_5.

[15] S. Bak and P. S. Duggirala. HyLAA: A tool for computing simulation-equivalent reachability for linear systems. In *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*, pages 173–178. ACM, 2017. DOI: 10.1145/3049797.3049808.

[16] R. Ray, A. Gurung, B. Das, E. Bartocci, S. Bogomolov, and R. Grosu. XSpeed: Accelerating reachability analysis on multi-core processors. In *Haifa Verification Conference*, pages 3–18. Springer, 2015. DOI: 10.1007/978-3-319-26287-1_1.

[17] M. Althoff, M. Forets, Y. Li, C. Schilling, M. Wetzlinger, and D. Zhuang. ARCH-COMP23 category report: Continuous and hybrid systems with linear continuous dynamics. In *Proc. of the 10th International Workshop on Applied Verification of Continuous and Hybrid Systems*, pages 34–60. EasyChair, 2023. DOI: 10.29007/nl86.

[18] T. Gan, M. Chen, Y. Li, B. Xia, and N. Zhan. Reachability analysis for solvable dynamical systems. *IEEE Transactions on Automatic Control*, 63(7):2003–2018, 2018. DOI: 10.1109/TAC.2017.2763785.

[19] M. Wetzlinger, N. Kochdumper, S. Bak, and M. Althoff. Fully automated verification of linear systems using inner and outer approximations of reachable sets. *IEEE Transactions on Automatic Control*, 68(12):7771–7786, 2023. DOI: 10.1109/TAC.2023.3292008.

[20] M. Wetzlinger, N. Kochdumper, S. Bak, and M. Althoff. Fully-automated verification of linear systems using reachability analysis with support functions. In *Proc. of the 26th International Conference on Hybrid Systems: Computation and Control*. ACM, 2023. DOI: 10.1145/3575870.3587121.