EPiC Series in Computing

Volume 96, 2023, Pages 9–33

Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)

Checking and Establishing Reachset Conformance in CORA 2023

Matthias Althoff

Technical University of Munich,
Department of Computer Engineering,
Munich, Germany
althoff@tum.de

Abstract

Tool presentation: When formally verifying models of cyber-physical systems, it is obviously important that their verification results can be transferred to all previous observations of the modeled systems. Our tool CORA makes it possible to transfer safety properties by checking whether all measurements of the real system lie in the set of reachable outputs of the corresponding model – we call this reachset conformance checking. In addition, we provide strategies to establish reachset conformance by injecting nondeterminism in models. This can be seen as some form of system identification, where instead of finding the most likely parameters, we compute a set of parameter values – not only for the model dynamics but also for the set of disturbances and measurement errors – to establish reachset conformance. By replacing real measurements with simulation results from a high-fidelity model, one can also check whether a high-fidelity model conforms to a simple model. We demonstrate the usefulness of reachset conformance by several use cases.

1 Introduction

The ultimate goal of formal verification methods is to ensure that a real system adheres to a formal specification (see Fig. 1). While formal verification methods for software can use models that exactly match the real implementation (e.g., C code), this is not possible for cyber-physical systems. Just alone due to Heisenberg's uncertainty principle, there is a fundamental limit to the accuracy with which we can model a physical system. A way out of this dilemma is to use nondeterministic models that subsume model uncertainties. In case the model encloses all possible behaviors of the real system, we can transfer safety verification results to the real system – otherwise, we would just verify models and not real systems. Interestingly, the verification of a model can also be seen as verifying conformance with a specification as illustrated in Fig. 1.

Even by ignoring Heisenberg's uncertainty principle, only for special system classes (e.g., linear systems without disturbances and measurement noise), a finite number of perfect measurements without quantization effects would suffice to uniquely identify a model [71]. For this reason, verifying reachset conformance for real systems is not possible, and our best option is to check

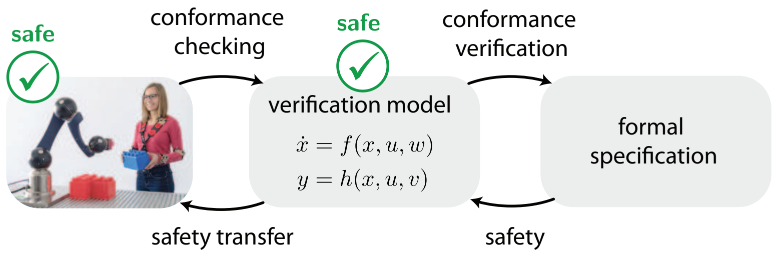


Figure 1: Reachset conformance can transfer safety properties from a verification model to a real system.

reachset conformance using samples, which we refer to as conformance checking. Nevertheless, conformance checking is one of the most systematic ways to obtain trustworthy models of cyber-physical systems subject to disturbances and measurement noise. In contrast, reachset conformance between two models can be verified by checking whether one reachable set encloses the other.

It is relatively trivial to establish reachset conformance by injecting a large amount of nondeterminism into the model. The apparent downside is that the obtained model is very conservative so that many verification problems cannot be solved although the real system would meet the corresponding specifications. Obviously, one has to find a compromise between the simplicity of the model and the required nondeterminism to simultaneously verify the model against the specification and reduce the complexity of the verification problem.

By replacing a real system with another model, one can obviously also check conformance between models. A possible use case is that due to intellectual property rights, a model of a supplier is confidential and can only be replicated for one's own development purposes. Conformance checking could provide sufficient confidence that the safety verification result of a system including the replicated model also holds after the supplied module is integrated.

1.1 State of the Art

Model conformance for cyber-physical systems has recently been surveyed [58]. In the following paragraphs, we will content ourselves with a summary of [58] and articles published after the survey. In addition to the survey, we provide an overview of tools for conformance checking.

Let us introduce some notation to define conformance. We are given a specification `spec`, an abstract system S_A, and an implementation S_I. The compliance of S_A with `spec` is written as $S_A \models spec$. In this work, we define that S_I conforms to S_A ($S_I \,\texttt{conf}\, S_A$) as [58, Sec. 2]:

$$S_I \,\texttt{conf}\, S_A \quad \wedge \quad S_A \models \texttt{spec} \quad \Longrightarrow \quad S_I \models \texttt{spec}.$$

In other words, if S_I conforms to S_A and S_A can be verified, then S_I is verified as well. We also say that S_A is a conformant model of S_I. Please note that alternative expressions exist for conformance relations, such as implementation relations [15] and refinement relations [14]. Subsequently, we refer to a white-box model if the dynamics is known and to a black-box model if the model can produce outputs given inputs but the dynamics is unknown.

Trace Conformance In essence, an implementation S_I is trace conformant to S_A if all output trajectories of S_I are possible output trajectories of S_A [58, Sec. 3.1]. Please note that the set of input trajectories to generate all output trajectories of S_I is the same as that for S_A. The term *trace conformance* originates from the computer science community, where behaviors are often referred to as traces.

While some previous work uses the term *trace conformance* [21, 22], other works use the terms *hybrid input-output conformance* [69, 70], *language inclusion* [12], *refinements* [13, 35], *implementation relations* [47], or *behavioral inclusion* [63]. In case a system is not trace conformant, one can use the ModelPlex approach [48] to still transfer safety properties by using a formally correct fail-safe controller [6, 16]. Approximate trace conformance relations are summarized in [58, Sec. 3.2]. An advantage of trace conformance is that specifications beyond safety specifications can be transferred as presented in [58, Sec. 3.1]. However, no method exists to ensure trace conformance when measurements are uncertain because infinitely many possible traces can produce the same measured trace. This would require to analyze trace conformance approximately.

Simulation Relations In contrast to trace conformance, simulation relations additionally relate system states [25, 63]. Thus, simulation relations cannot be used for real systems (of which one only has measurements) but can be helpful to compare two white-box models. An implementation S_I is simulated by system S_A if their states can be related so that $(x_I, x_A) \in \mathcal{S}$ and all outputs are identical given the same inputs. Simulation relation implies trace conformance, i.e., if S_A is a simulation of S_I, it is also a trace conformant model of S_I [15, Thm. 7.70]. The work in [65] provides necessary and sufficient conditions for a simulation relation between two constrained linear systems. A fixed-point computation is used in [25, 28] to compute simulation relations of hybrid automata.

If a simulation relation holds in both directions, the systems are called bisimilar. Bisimulation relations have been computed for linear systems [52, 68], nonlinear systems [50, 64], switching linear systems [54], and hybrid systems [63, 67]. The work in [34] unifies the notion of bisimulation such that it transcends discrete and continuous systems. Approximate simulation relations are summarized in [58, Sec. 3.4].

Reachset Conformance Due to the nature of real cyber-physical systems, trace conformance is challenging to check – just alone due to uncertain measurements, a real system would always produce a different output trace for the same initial state and input trajectory. Also, as discussed above, simulation relations cannot be used for real systems. Thus, to transfer safety properties to real systems, our tool focuses on reachset conformance, which essentially checks whether for all times t and input trajectories $u(\cdot) \in \mathcal{U}$ ($\mathcal{U}$ is the set of input trajectories), the set of reachable outputs $Reach_t(S_A, u(\cdot))$ of the abstract system S_A contains all possible measurements $Reach_t(S_I, u(\cdot))$ of the implementation S_I:

$$S_I \, \mathtt{conf}_R \, S_A \quad \Longleftrightarrow \quad \forall t, \, \forall u(\cdot) \in \mathcal{U} : Reach_t(S_I, u(\cdot)) \subseteq Reach_t(S_A, u(\cdot)).$$

In [59, Thm. 1.], it is shown that reachset conformance is necessary and sufficient to transfer safety properties, which are the predominant properties for certifying cyber-physical systems. In addition, properties that can be formalized using reachset temporal logic [56] can be transferred. Reachset conformance is a weaker conformance notion than trace conformance, i.e., if S_A is a trace conformant model of S_I, it is also a reachset conformant model of S_I [59, Thm. 2].

Reachset conformance can only be proven between models because one obviously cannot compute the reachable set of a real system. For real systems, one resorts to checking whether the reachable output of the abstract model S_A contains all measurements of the implementation S_I. Even when S_I is a model, one often resorts to using simulation results rather than reachability analysis due to the computational complexity of computing reachable sets. As mentioned above, we call this approach reachset conformance checking, because we can only check the containment of samples rather than proving reachset conformance. To the best knowledge of the author, reachset conformance checking was first presented in [5]. However, the term *reachset conformance* is not used in that work and was introduced in [57] together with a formalization of reachset conformance checking. This method has been applied to numerous applications, such as safe human-robot co-existence [45], safe robot manipulators [43], force control [44], and analog circuits [37].

Open-Source Tools for Conformance Checking While there exist many tools for reachability analysis, such as Ariadne [17], C2E2 [24], CORA [4], Flow* [19], Isabelle/HOL [36], JuliaReach [18], and SpaceEx [27] (alphabetical order), none of these tools support reachset conformance checking. There exist tools for (approximate) trace conformance checking, e.g., [1, 51], but they do not seem to be supported anymore. To obtain simulation relations of cyber-physical systems, PHAVer [26] is arguably the most developed tool. For approximate bisimulation relations of linear continuous systems, one typically uses simulation functions using linear matrix inequalities [31, 33], and for nonlinear systems, one can use tools for solving sum-of-squares problems [32]; one of the most used tools for sum-of-squares problems is SOSTOOLS [55]. The approach in [50] additionally uses dReal [30] in case SOSTOOLS does not find a proper simulation function. The previously mentioned ModelPlex approach is implemented in KeYmaera X [29].

1.2 Contributions and Organization

We present the first tool to check and establish reachset conformance of continuous and hybrid systems. As an underlying method, we use reachability analysis to check whether a model encloses all observed behaviors of the real system. Noteworthy features of our tool are:

- We provide methods to check reachset conformance as well as to inject uncertainties to establish reachset conformance.
- Our tool can be used for real systems, black-box models, and white-box models.
- Continuous (linear and nonlinear) as well as hybrid system dynamics are supported.
- Various sources of uncertainties are considered: Initial state uncertainties, measurement uncertainties, disturbances, and uncertain parameters.
- We provide a unified user interface so that reachset conformance can be checked or established using only a few lines of code.

We present selected methods for checking reachset conformance in Sec. 2. Reachset conformance is established in Sec. 3. The usefulness of these methods is demonstrated for several use cases in Sec. 4 and we draw conclusions in Sec. 5.

2 Checking Reachset Conformance

Before checking reachset conformance, we first discuss the influence of measurement errors. One can integrate measurement errors fairly easily in reachset conformance, while this would require the analysis of infinitely many traces when checking trace conformance. Let us denote the set of measurement errors as $\mathcal{V}$. Given the exact reachable set of the abstract model, we can only say for sure that a measurement y falsifies reachset conformance if the set $y \oplus \mathcal{V}$ does not intersect the reachable set, where $\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$ denotes the Minkowski sum (obviously, $\mathcal{A}$ or $\mathcal{B}$ can be singletons). Conversely, we can only be sure that the exact measurement is within the reachable set if $y \oplus \mathcal{V}$ is enclosed by the reachable set. We call this case *measurement conformant*. Unfortunately, one can only compute reachable sets exactly for a few special cases [41], so we have to compute an over-approximation for falsifying reachset conformance and an under-approximation to ensure measurement conformance as illustrated in Fig. 2.

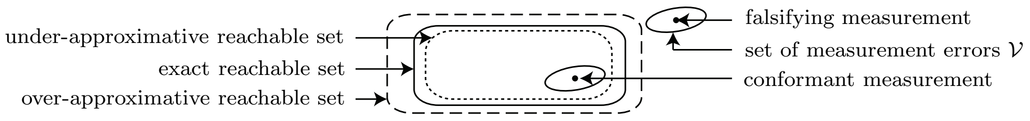

under-approximative reachable set
exact reachable set
over-approximative reachable set
falsifying measurement
set of measurement errors $\mathcal{V}$
conformant measurement

Figure 2: Falsifying and conformant measurement.

To check reachset conformance, we require reachability analysis, test cases, and efficient enclosure checks for different set representations. The over-approximation and under-approximation of reachable sets is well documented in CORA and partly summarized in [7]. Our implementation of test cases is briefly summarized in Sec. 2.1 and the implementation of enclosure checks is briefly summarized in Sec. 2.2.

2.1 Test Cases

Test cases can be recorded from real systems or generated from models. The generation of good test cases is a topic of its own and not discussed in detail here. So far, we have implemented the methods shown in Tab. 1 to generate test cases; obviously, a test case can also be generated manually. The resulting solution can be obtained through

```
simRes = simulateRandom(obj, params, options)
```

where the type of test generation is specified through `options.type` using the options listed in Tab. 1. The sampling strategy of all types can be customized as described in the CORA manual.

Test cases are handled by the class `testCase`. An object of class `testCase` can be constructed as follows:

```
obj = testCase(y,u,x,dt)
obj = testCase(y,u,x0,dt)
```

with input arguments

Table 1: Implemented methods for automatic test case generation (alphabetical order).

Name	Description
`constrained`	Constrains the inputs so that the solutions stay within a specified reachable set. This approach requires white-box models because the internal state is required to enforce the constraints; when transferring the inputs to a real system, dangerous states can be avoided (given that the model is sufficiently accurate).
`gaussian`	Identical to the type `standard`, but the inputs are sampled from Gaussian distributions. The sampled values can be constrained to a specified input set.
`rrt`	Computes rapidly exploring random trees. This approach can only be used for white-box models, because it requires the internal state.
`standard`	Default method that computes random inputs of specified input sets. One can customize the sampling strategy and how frequently the inputs are changed.

- `y` measured outputs sequence
- `u` input sequence
- `dt` sampling time
- `x` state sequence
- `x0` initial state vector

The arguments `y`, `u`, and `x` must all contain a vectors, reflecting the number of samples considered for a test case. Suppose one uses recorded data from a real system. In that case, the state sequence is typically not available so that a test case can also be initialized by its (estimated) initial state `x0`.

The method `sequentialTestCases` generates from one test case a cell array of $a - a^*$ test cases of length $a^* \leq a$, each starting one time increment after the previous one. This results in the relation $y^{(m+1)}(t_k) = y^{(m)}(t_{k+1})$ for the output sequence, where m refers to the m-th generated test case and k refers to the time step; this relationship holds analogously for the input sequence and the state sequence (if it exists). Splitting a lengthy test case into several smaller ones is useful when the reachable set is computed conservatively so that falsification becomes unlikely after a certain time. The method `plot` plots the measured outputs of the test case.

2.2 Efficient Methods for Checking Set Enclosure

The set of measurement errors $\mathcal{V}$ can be considered by computing the Minkowski difference $\mathcal{R} \ominus \mathcal{V}$ (defined as $\mathcal{R} \ominus \mathcal{V} = \{x \in \mathbb{R}^n | x \oplus \mathcal{V} \subseteq \mathcal{R}\}$) or Minkowski sum $\mathcal{R} \oplus \mathcal{V}$ with the reachable set $\mathcal{R}$ to establish measurement conformance or falsify reachset conformance, respectively. Thus, the problem of set enclosure reduces to the problem of checking whether a specific measurement y lies within $\mathcal{R} \ominus \mathcal{V}$ or $\mathcal{R} \oplus \mathcal{V}$. To keep the presentation concise, we only present exact containment checks (up to numerical precision). Approximate solutions are presented in the CORA manual – if these algorithms return containment, the point is certainly contained; however, some contained points may not be confirmed. Only approximate approaches are available for some set representations, such as polynomial zonotopes.

Many set representations can be formulated as several inequalities, which can be directly evaluated as shown in Tab. 2. CORA also supports constrained hyperplanes of the form $\{x | ax = b \wedge Cx \leq d\}$, which can be directly converted to an H-polytope so that the check in Tab. 2 can be used. For the subsequent set representations, we introduce the notation $G_{(\cdot,i)}$ returning the i-th column of the matrix G.

Table 2: Trivial set containment checks (alphabetical order). Matrices, vectors, and functions are of appropriate dimension and the inequalities have to hold for each dimension.

Set	**Definition**	**Containment check**
capsule	$\mathcal{L} \oplus \mathcal{S}$, $\mathcal{L} = \{c + \beta g \vert \beta \in [-1, 1]\}$, $\mathcal{S} = \{x \vert \Vert x\Vert_2 \leq r\}$	$(\Vert(\delta^T g_n) g_n - \delta\Vert_2 < r \wedge \vert\delta^T g_n\vert < \Vert g\Vert_2)$ $\vee \Vert\delta - g\Vert_2 \leq r \vee \Vert\delta + g\Vert_2 \leq r$, $g_n = g/\Vert g\Vert_2$, $\delta = x - c$
ellipsoid	$\{x \vert x^T Q x \leq 1\}$	$x^T Q x \leq 1$
H-polytope	$\{x \vert Cx \leq d\}$	$Cx \leq d$
interval	$[\underline{x}, \overline{x}]$	$\underline{x} \leq x \wedge x \leq \overline{x}$
level set	$\{x \vert \mu(x) \leq 0\}$	$\mu(x) \leq 0$

Zonotope and Zonotope Bundle Given a center $c \in \mathbb{R}^n$ and a generator matrix $G \in \mathbb{R}^{n \times p}$, a zonotope can be defined as [40, Def. 1]

$$\mathcal{Z} := \left\{ c + \sum_{i=1}^{p} \beta_i G_{(\cdot,i)} \,\middle|\, \beta_i \in [-1, 1] \right\}.$$

We introduce the short notation $\mathcal{Z} = \langle c, G\rangle$ for later use. Zonotopes with $n \leq \overline{n}$ (currently $\overline{n} = 4$) are converted to an H-polytope according to [11, Thm. 7] and then the containment check for H-polytopes is applied. For larger dimensions, we use the approach in [39, eq. (8)] due to its favorable computational complexity.

A zonotope bundle $\mathcal{Z}_\cap = \bigcap_{i=1}^{s} \mathcal{Z}_i$ is defined as the intersection of a finite set of zonotopes [10, Def. 4]; the intersection is not computed, but the zonotopes $\mathcal{Z}_i$ are stored in a list to realize so-called lazy computations. A point is within a zonotope bundle if it is in all zonotopes.

Constrained Zonotope Given a start vector $c \in \mathbb{R}^n$, a generator matrix $G \in \mathbb{R}^{n \times p}$, a constraint matrix $A \in \mathbb{R}^{m \times p}$, and a constraint vector $b \in \mathbb{R}^m$, a constrained zonotope $\mathcal{CZ} \subset \mathbb{R}^n$ can be defined as [62, Def. 3]

$$\mathcal{CZ} := \left\{ c + \sum_{i=1}^{p} \beta_i G_{(\cdot,i)} \,\middle|\, \sum_{i=1}^{p} \beta_i A_{(\cdot,i)} = b, \quad \beta_i \in [-1, 1] \right\}.$$

We check whether a point is inside a constrained zonotope by the linear program in [62, eq. (20)], which essentially tries to find $\Vert\beta\Vert_\infty \leq 1$ so that

$$\begin{bmatrix} G \\ A \end{bmatrix} \beta = \begin{bmatrix} x - c \\ b \end{bmatrix}$$

is fulfilled.

3 Synthesizing Reachset Conformance

We present a deliberately simple approach to demonstrate the synthesis of reachset conformant models in the sense that no falsifying measurement exists. The presented approach slightly generalizes [46, Sec. III.B] to systems with initial state uncertainties and also covers the reachset synthesis approach in [43, Sec. IV.C], which only returned required uncertainties as axis-aligned boxes. As a result, one obtains the sets bounding the initial state $x_0 \in \mathcal{X}_0 \subset \mathbb{R}^n$, the disturbance $w(t) \in \mathcal{W}_c \subset \mathbb{R}^n$, and the measurement uncertainty $v(t) \in \mathcal{V} \subset \mathbb{R}^m$ so that the linear system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + w(t) \\ y(t) &= Cx(t) + v(t) \end{aligned} \tag{1}$$

is reachset conformant with respect to all measurements y taken at discrete times with a fixed time increment $h = t_{k+1} - t_k$. All matrices A, B, and C are of appropriate dimension. The extension to nonlinear systems is briefly outlined later. As it also becomes obvious later, the approach can be easily modified so that one or two of the sets $\mathcal{X}_0$, $\mathcal{W}_c$, and $\mathcal{V}$ are given. For later derivations, we split the initial set into a fixed value denoted by x_0 and an initial set $\tilde{\mathcal{X}}_0$ centered at the origin so that $\mathcal{X}_0 = x_0 \oplus \tilde{\mathcal{X}}_0$. The exact reachable set of the state of (1) and its under-approximation [43, Prop. 1] can be computed as

$$\begin{aligned} &e^{At}(x_0 \oplus \tilde{\mathcal{X}}_0) \oplus \int_0^t e^{A(t-\tau)}(Bu(\tau) \oplus \mathcal{W}_c)d\tau \\ \supseteq &e^{At}(x_0 \oplus \tilde{\mathcal{X}}_0) \oplus \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau \oplus \int_0^t e^{A(t-\tau)}d\tau\, \mathcal{W}_c = \mathcal{R}_x(t, x_0, u(\cdot)). \end{aligned} \tag{2}$$

The under-approximation is obtained by only considering constant disturbances within $\mathcal{W}_c$; this makes it possible to factor out $\mathcal{W}_c$ for subsequent derivations. Inserting the reachable set of the state into the output function in (1) results in

$$\mathcal{R}(t, x_0, u(\cdot)) = C\,\mathcal{R}_x(t, x_0, u(\cdot)) \oplus \mathcal{V}. \tag{3}$$

Given the set of measured outputs $\mathcal{Y}(h)$ at time h, we can formulate the following condition for reachset conformance at time h as

$$\forall y(h) \in \mathcal{Y}(h) : y(h) \in \mathcal{R}(h, x_0, u(\cdot)). \tag{4}$$

Let us introduce the state $\tilde{x}(t, x_0, u(\cdot))$ and output $\tilde{y}(t, x_0, u(\cdot))$ obtained by starting in x_0 with input trajectory $u(\cdot)$ and without disturbance and measurement uncertainty. Next, we simplify the conformance synthesis problem by subtracting $\tilde{y}(h, x_0, u(\cdot))$ from both sides of (4):

$$\begin{aligned} &\forall y(h) \in \mathcal{Y}(h) : y(h) - \tilde{y}(h, x_0, u(\cdot)) \in \mathcal{R}(h, x_0, u(\cdot)) - \tilde{y}(h, x_0, u(\cdot)) \\ \overset{(2),(3)}{\iff} &\forall y(h) \in \mathcal{Y}(h) : y(h) - \tilde{y}(h, x_0, u(\cdot)) \in C\left(e^{Ah}\tilde{\mathcal{X}}_0 \oplus \int_0^h e^{A(h-\tau)}\, d\tau\, \mathcal{W}_c\right) \oplus \mathcal{V}. \end{aligned} \tag{5}$$

Thus, subtracting the nominal solution makes the conformance-checking problem independent of x_0 and $u(\cdot)$. For further simplification, we introduce

$$\begin{aligned} A_d &= e^{Ah}, \\ \Upsilon &= \int_0^h e^{A(h-\tau)}\, d\tau, \\ B_d &= \Upsilon\, B, \\ \mathcal{W} &= \Upsilon\, \mathcal{W}_c, \\ \Lambda_k &= C\, A_d^k, \end{aligned} \tag{6}$$

so that we can state using (2) that

$$\mathcal{R}_x(t_{k+1}, x_0, u(\cdot)) - \tilde{x}(t_{k+1}, x_0, u(\cdot)) = A_d\big(\mathcal{R}_x(t_k, x_0, u(\cdot)) - \tilde{x}(t_k, x_0, u(\cdot))\big) \oplus \mathcal{W}. \tag{7}$$

By iteratively applying (7), one obtains for $k \geq 1$ that

$$\mathcal{R}_x(t_k, x_0, u(\cdot)) - \tilde{x}(t_k, x_0, u(\cdot)) = A_d^k \tilde{\mathcal{X}}_0 \oplus \bigoplus_{i=0}^{k-1} A_d^i \mathcal{W}.$$

After inserting the above result into the output function of (1), we can generalize (5) to any time t_k:

$$\forall y(t_k) \in \mathcal{Y}(t_k) : y(t_k) - \tilde{y}(t_k, x_0, u(\cdot)) \in \Lambda_k \tilde{\mathcal{X}}_0 \oplus \bigoplus_{i=0}^{k-1} \Lambda_i \mathcal{W} \oplus \mathcal{V}. \tag{8}$$

Obviously, for $k = 0$, we have that $y(t_k) - \tilde{y}(t_k, x_0, u(\cdot)) \in C\tilde{\mathcal{X}}_0 \oplus \mathcal{V}$. Because this constraint is the same for the discrete-time system

$$\begin{aligned} x(t_{k+1}) &= A_d x(t_k) + B_d u(t_k) + w(t_k) \\ y(t_k) &= C x(t_k) + v(t_k) \end{aligned}$$

with $w(t_k) \in \mathcal{W}$ ($\mathcal{X}_0$ and $\mathcal{V}$ are identical to the continuous-time system) and $\forall t \in [t_k, t_{k+1}[$: $u(t) = u(t_k)$ (zero-order hold), the results transfer to this system class directly. The equivalence can easily be shown by performing the same previous steps for the discrete-time dynamics.

To find optimal values of $\tilde{\mathcal{X}}_0$, $\mathcal{W}$, and $\mathcal{V}$ through a linear program, we a) restrict these sets to special zonotopes which originate from shifting and scaling template zonotopes and b) choose a specific cost function. The corresponding continuous disturbance set $\mathcal{W}_c$ can be obtained from (6) as follows:

$$\mathcal{W}_c = \left(\int_0^h e^{A(h-\tau)}\, d\tau\right)^{-1} \mathcal{W} \overset{[3,\text{ eq. } (3.6)]}{=} \big(A^{-1}(e^{Ah} - I_n)\big)^{-1} \mathcal{W}.$$

In case A is not invertible, one can compute the integral $\int_0^h e^{A(h-\tau)}\, d\tau$ using a Taylor series up to floating point precision or bound that Taylor series by an interval matrix as shown in [3, eq. (3.2)], followed inverting the obtained interval matrix [60]. Because the continuous model would already be reachset conformant when the disturbance is constant between time steps, considering all disturbances will only add conservatism and not impede the soundness of the result. After introducing the centers of the respective zonotopes as c_X, c_W, c_V, the corresponding stretching factors as α_X, α_W, α_V, and the operator $\operatorname{diag}(\alpha_X)$ returning a diagonal matrix whose entries are those of α_X, the shifted and scaled template zonotopes can be formalized as

$$\begin{aligned} \tilde{\mathcal{X}}_0 =& \langle c_X, G_X \operatorname{diag}(\alpha_X)\rangle, \quad G_X \in \mathbb{R}^{n\times p}, \alpha_X \in \mathbb{R}^p, \\ \mathcal{W} =& \langle c_W, G_W \operatorname{diag}(\alpha_W)\rangle, \quad G_W \in \mathbb{R}^{n\times q}, \alpha_W \in \mathbb{R}^q, \\ \mathcal{V} =& \langle c_V, G_V \operatorname{diag}(\alpha_V)\rangle, \quad G_V \in \mathbb{R}^{m\times r}, \alpha_V \in \mathbb{R}^r. \end{aligned} \tag{9}$$

Obviously, some of the values of α_X, α_W, and α_V can be fixed if they should not be optimized. We introduce

$$\begin{aligned} c &= \begin{bmatrix} c_X^T, & c_W^T, & c_V^T \end{bmatrix}^T, \\ \alpha &= \begin{bmatrix} \alpha_X^T, & \alpha_W^T, & \alpha_V^T \end{bmatrix}^T \end{aligned} \tag{10}$$

for later use. Let us also introduce the n-dimensional vector of ones $\mathbf{1}_n \in \mathbb{R}^n$, the n-dimensional identity matrix I_n, and the matrix of zeros $\mathbf{0}$ of proper dimension. From now on, we also make the assumption that absolute values of matrices are computed elementwise, i.e., $|M|_{ij} = |M_{ij}|$ for all entries of a matrix M.

3.1 Interval Norm

For specifying the cost function of the reachset synthesis problem, we define the interval norm of a zonotope $\mathcal{Z} = \langle c, G\rangle$ with $G \in \mathbb{R}^{n\times p}$ and a user-specified weighting vector $\sigma \in \mathbb{R}^n_{>0}$ as

$$\|\mathcal{Z}\|_{\tilde{I},\sigma} := \sigma^T \big|[c,\, G]\big| \mathbf{1}_{p+1}.$$

This norm essentially returns the weighted sum of the edge lengths of the interval hull of a zonotope, where one of the generators is the center c [3, Prop. 2.2]. As a norm it fulfills the following properties: a) $\|\mathcal{Z}\|_{\tilde{I},\sigma} \geq 0$, b) $\|\langle \mathbf{0}\rangle\|_{\tilde{I},\sigma} = 0$, c) $\|\gamma\mathcal{Z}\|_{\tilde{I},\sigma} = |\gamma|\|\mathcal{Z}\|_{\tilde{I},\sigma}$ ($\gamma \in \mathbb{R}$), d) $\|\langle [c_1,\, G_1] + [c_2,\, G_2]\rangle\|_{\tilde{I},\sigma} \leq \|\langle [c, G_1]\rangle\|_{\tilde{I},\sigma} + \|\langle [c_2, G_2]\rangle\|_{\tilde{I},\sigma}$. From now on, we exclude the center from the norm by first translating the center of the zonotope to the origin. The combined operation of moving the center to the origin followed by applying the interval norm is denoted by

$$\|\mathcal{Z}\|_{I,\sigma} := \sigma^T \big|G\big| \mathbf{1}_p. \tag{11}$$

We can now state the theorem for solving the reachset conformance synthesis problem as a linear program.

Theorem 1 (Linear Program for Synthesizing Reachset Conformance)**.** *The linear program*

$$\begin{aligned} &\min_z n^T z \\ &\textit{such that } \tilde{A}z \leq \tilde{b}, \end{aligned} \tag{12}$$

minimizes the cost function

$$\sum_{k=0}^{N} \omega_k \|\mathcal{R}(t_k, x_0, u(\cdot)) - \tilde{y}(t_k, x_0, u(\cdot))\|_{I,\sigma} \tag{13}$$

using the user-specified weighting vector ω and ensures reachset conformance, i.e. $\forall k \in \{0, 1, \ldots, N\}$: (8) holds. The cost function and the constraint are composed as follows:

$$z = \begin{bmatrix} c^T, & \alpha^T \end{bmatrix}^T, \tag{14}$$

$$n = \Big[\mathbf{0}, \quad \textstyle\sum_{k=0}^{N} \omega_k \sigma^T \tilde{G}(I_m, t_k)\Big]^T, \tag{15}$$

$$\tilde{A} = \big[\tilde{A}(t_0)^T, \quad \tilde{A}(t_1)^T, \quad \ldots, \quad \tilde{A}(t_N)^T, \quad [\mathbf{0}, -I_{p+q+r}]^T\big]^T, \tag{16}$$

$$\tilde{b} = \big[\tilde{b}(t_0)^T, \quad \tilde{b}(t_1)^T, \quad \ldots, \quad \tilde{b}(t_N)^T, \quad \mathbf{0}\big]^T, \tag{17}$$

$$\tilde{A}(t_k) = \big[-N(t_k)\Gamma(t_k), \quad -\tilde{G}(N(t_k), t_k)\big], \tag{18}$$

$$\tilde{b}(t_k) = -\max_{y(t_k)\in\mathcal{Y}(t_k)} N(t_k)\big(y(t_k) - \tilde{y}(t_k, x_0, u(\cdot))\big), \tag{19}$$

$$\tilde{G}(M, t_k) = \begin{cases} \Big[|M\,\Lambda_k\,G_X|, \quad \sum_{i=0}^{k-1} |M\,\Lambda_i\,G_W|, \quad |M\,G_V|\Big], \textit{ for } k \geq 1 \\ \Big[|M\,\Lambda_k\,G_X|, \quad \mathbf{0}, \quad |M\,G_V|\Big], \textit{ for } k = 0 \end{cases} \tag{20}$$

and $N(t_k)$ is the matrix of normal vectors of the zonotope with generator matrix

$$G(t_k) = \begin{cases} \Big[\Lambda_k\, G_X \operatorname{diag}(\alpha_X), \quad [\Lambda_0\, G_W \operatorname{diag}(\alpha_W), \ldots, \Lambda_{k-1}\, G_W \operatorname{diag}(\alpha_W)], \quad G_V \operatorname{diag}(\alpha_V)\Big] \text{, for } k \geq 1 \\ \Big[\Lambda_k\, G_X \operatorname{diag}(\alpha_X), \quad G_V \operatorname{diag}(\alpha_V)\Big] \text{, for } k = 0 \end{cases} \tag{21}$$

as computed by [11, Thm. 7].

Proof. See Appendix A. □

3.2 Frobenius Norm

As an alternative to the interval norm, one can also reduce the Frobenius norm of a zonotope $\mathcal{Z} = \langle c, G\rangle$, which we define using the positive definite matrix $P \in \mathbb{R}^{n\times n}$ as (inspired by [20, Def. 2]):

$$\|\mathcal{Z}\|_{\tilde{F},P} := \sqrt{\operatorname{trace}([c,\, G]^T P\, [c, G])}.$$

In contrast to [20, Def. 2], the center is added to the definition so that the neutral element is defined and thus the norm is defined on a vector space. Analogously to the interval norm, we exclude the center from the norm from now on by first translating the center of the zonotope to the origin. The combined operation of moving the center to the origin followed by applying the Frobenius norm is denoted by

$$\|\mathcal{Z}\|_{F,P} := \sqrt{\operatorname{trace}(G^T P\, G)}. \tag{22}$$

The cost function using the Frobenius norm results in a quadratic optimization problem. As a prerequisite, we define an operator that changes all entries to 0, except for the diagonal elements:

$$[\widetilde{\operatorname{diag}}(M)]_{ij} = \begin{cases} M_{ij} \text{ , for } i = j, \\ 0 \text{ , for } i \neq j. \end{cases}$$

We also require the operator $\operatorname{blkdiag}(O, P, \ldots, Q)$ returning a block diagonal matrix:

$$\operatorname{blkdiag}(O, P, \ldots, Q) = \begin{bmatrix} O & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & P & \cdots & \mathbf{0} \\ \mathbf{0} & \cdots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & Q \end{bmatrix}.$$

Theorem 2 (Quadratic Program for Synthesizing Reachset Conformance)**.** *The quadratic program*

$$\min_z \frac{1}{2} z^T H z$$
$$\text{such that } \tilde{A} z \leq \tilde{b},$$

minimizes the cost function

$$\sum_{k=0}^{N} \omega_k \|\mathcal{R}(t_k, x_0, u(\cdot)) - \tilde{y}(t_k, x_0, u(\cdot))\|_{F,P}^2 \tag{23}$$

using the user-specified weighting vector ω *and ensures reachset conformance, i.e.* $\forall k \in \{0, 1, \dots, N\}$: (8) *holds. The constraints are identical to Thm. 1 and the cost function is composed as follows:*

$$H = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sum_{k=0}^{N} \omega_k \tilde{H}(t_k) \end{bmatrix}, \tag{24}$$

$$\tilde{H}(t_k) = \text{blkdiag}\left(\widetilde{\text{diag}}\left(G_X^T \Lambda_k^T P \Lambda_k G_X\right), \tilde{H}_W(t_k), \widetilde{\text{diag}}\left(G_V^T P G_V\right)\right), \tag{25}$$

$$\tilde{H}_W(t_k) = \begin{cases} \widetilde{\text{diag}}\left(G_W^T (\sum_{i=0}^{k-1} \Lambda_i^T P \Lambda_i) G_W\right), \textit{ for } k \geq 1 \\ \mathbf{0}, \textit{ for } k = 0 \end{cases}. \tag{26}$$

Proof. See Appendix B. □

3.3 Sketch of Extension to Nonlinear Systems

For nonlinear systems, we restrict ourselves to axis-aligned sets $\tilde{\mathcal{X}}_0$, $\mathcal{W}$, and $\mathcal{V}$ and linearize the dynamics anew for each time step. Binary search for each dimension is utilized to find the sets $\tilde{\mathcal{X}}_0$, $\mathcal{W}$, and $\mathcal{V}$ as done in [43]. Synthesizing reachset conformant models is an active area of research and more advanced methods will be provided in future CORA releases.

4 Numerical Experiments

We demonstrate our novel extension of CORA for reachset conformance with several previously published use cases, which we organize in two dimensions: use cases where the abstract system S_A is linear or nonlinear (first dimension) and use cases where the implementation S_I is a white-box model or either a real system or a black-box model (second dimension). This results in four categories, of which we demonstrate one use case each.

Besides the standard parameters for dynamical systems in CORA stored in `params`, we also require the test suite stored in `params.testSuite` for the subsequent calls of the `conform` method. Whether a conformance check or a conformance synthesis is conducted, is steered by the selected algorithm through `options.alg` – each implemented algorithm is either for checking conformance or synthesizing uncertain sets to ensure conformance. Depending on which algorithm is used, a Boolean value is returned indicating whether the model is reachset conformant or a struct containing various sets ensuring reachset conformance is returned.

4.1 Linear Reachset Conformant Model of a White-Box Model

The first use case taken from [37] is to establish reachset conformance for the analog circuit depicted in Fig. 3. The analog circuit is a second-order low-pass filter modeled by SPICE[1], a general-purpose, open-source analog electronic circuit simulator. In addition, real measurements are later used to additionally try to falsify reachset conformance for the real circuit.

[1]SPICE is an abbreviation of Simulation Program with Integrated Circuit Emphasis.

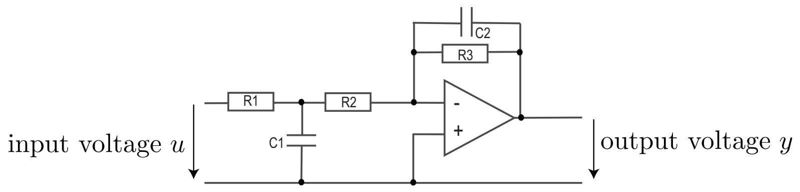


Figure 3: Analog circuit realizing a second-order low-pass filter.

4.1.1 Modeling

In the standard design flow for analog mixed-signal circuits, the complete circuit is decomposed into its principal elements, which are first analyzed and designed using idealized low-order behavioral models. Once the low-order behavioral model meets the specifications, detailed transistor-level designs are analyzed (using, e.g., SPICE) to avoid costly redesign cycles. In this use case, we synthesize a reachset conformant behavioral model so that we can verify this model and transfer the result to the detailed SPICE model and later to the real circuit.

Because obtaining behavior models from SPICE models is quite challenging, we refer the reader to two methods that are used for this study: The first behavioral model is obtained using eigenvalue clustering (see [66]) and the second one using local linearizations (see [42]). Both techniques have in common that the behavior models are linear and only locally valid. Thus, a piecewise affine (PWA) model is created for both approaches. The reachset conformance synthesis then essentially requires a reachset conformant model for each state space region of the piecewise linear model as presented in [37].

4.1.2 Numerical Results

Our test suite consists of simulation results from an accurate system model as well as measurements from the real circuit for the input signals $u(t) = 3V$, $u(t) = 4V$, $u(t) = 3V\sin(2\pi t)$, $u(t) = 3V\sin(200\pi t)$, $u(t) = 4V\sin(2\pi t)$, and $u(t) = 4V\sin(200\pi t)$. Using this test suite and the first behavioral model `lowPassFilter_eig`, we call (less relevant commands are not shown for brevity)

```
options.alg = conformanceSynthesis;
params.testSuite = lowPassFilterTest;
paramsConform = conform(lowPassFilter_eig,params,options);
```

to generate a new set of parameters `paramsConform` that establishes reachset conformance. This is repeated for the other model `lowPassFilter_lin`. As shown in Fig. 4, the reachable sets for the reachset conformant models enclose all measurements for the input signals $u(t) = 4.5V$ and $u(t) = 4V sin(20\pi t)$, even though these input signals are not included in the test suite `lowPassFilterTest`.

4.2 Nonlinear Reachset Conformant Model of a White-Box Model

The second use case inspired by [5] checks reachset conformance for a model of an automated BMW 320i road vehicle depicted in Fig. 5 – if successful, the model can be used for formal verification, see, e.g. [6]. Because we check conformance against a high-fidelity white-box model, we can use rapidly-exploring random trees to falsify the abstract model S_A as quickly as possible.

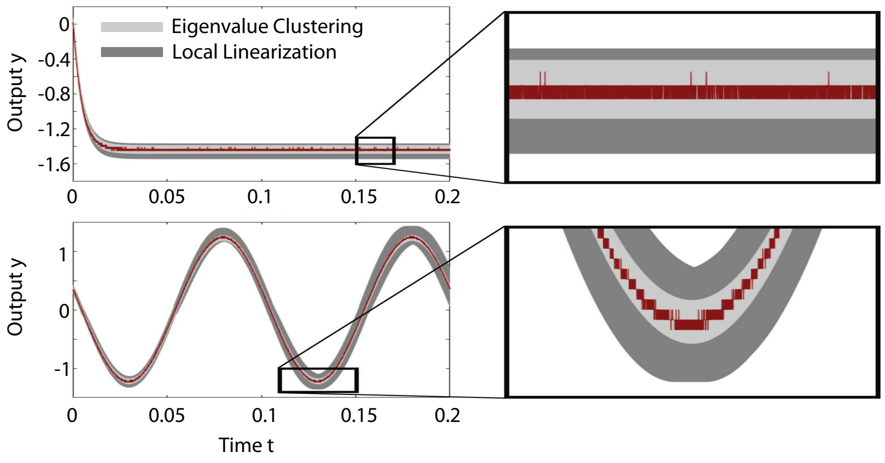


Figure 4: Reachable sets of the reachset conformant models for the input signals $u(t) = 4.5V$ (top) and $u(t) = 4V sin(20\pi t)$ (bottom). The corresponding measurements of the real circuit are depicted in red.

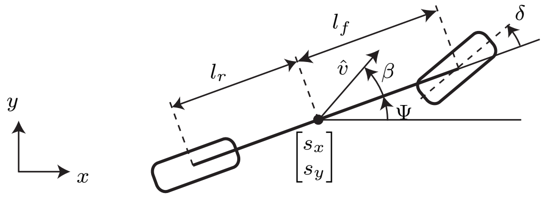


Figure 5: Single-track model.

4.2.1 Modeling

The abstract model S_A is a so-called single-track model (aka bicycle model), which describes the primary effects of the lateral dynamics of a road vehicle. The name of the model refers to the fact that the front and rear wheel pairs are each lumped into one wheel because the roll dynamics is neglected (see Fig. 5). In contrast to many other single-track models, we additionally consider the load transfer due to longitudinal acceleration a_x. In contrast to [5], we use the steering velocity instead of the steering angle as an input to the model to consider that the maximum steering velocity is limited; see [9, Sec. III.C]. Given the slip angle β, the orientation Ψ, the yaw rate $\dot{\Psi}$, the velocity $\hat{v}$, the steering angle δ, the positions s_x, s_y in Cartesian coordinates, the inputs $\hat{v}_\delta$ (steering velocity) and a_{long} (longitudinal acceleration), and the parameters in Tab. 3, the vehicle dynamics can be formulated as (see reference to

CommonRoad model in [9, Sec. III.C]):

$$
\begin{aligned}
\dot{\delta} =& \hat{v}_\delta, \\
\dot{\beta} =& \frac{\mu}{\hat{v}(l_r + l_f)} \Big(C_{S,f}(g l_r - a_{\mathrm{long}} h)\delta - (C_{S,r}(g l_f + a_{\mathrm{long}} h) + C_{S,f}(g l_r - a_{\mathrm{long}} h))\beta \\
& + (C_{S,r}(g l_f + a_{\mathrm{long}} h) l_r - C_{S,f}(g l_r - a_{\mathrm{long}} h) l_f) \frac{\dot{\Psi}}{\hat{v}} \Big) - \dot{\Psi}, \\
\ddot{\Psi} =& \frac{\mu m}{I_z(l_r + l_f)} \Big(l_f C_{S,f}(g l_r - a_{\mathrm{long}} h)\delta \\
& + (l_r C_{S,r}(g l_f + a_{\mathrm{long}} h) - l_f C_{S,f}(g l_r - a_{\mathrm{long}} h))\,\beta \\
& - \left(l_f^2 C_{S,f}(g l_r - a_{\mathrm{long}} h) + l_r^2 C_{S,r}(g l_f + a_{\mathrm{long}} h)\right) \frac{\dot{\Psi}}{\hat{v}} \Big), \\
\dot{\hat{v}} =& a_{\mathrm{long}}, \\
\dot{s}_x =& \hat{v} \cos(\beta + \Psi), \\
\dot{s}_y =& \hat{v} \sin(\beta + \Psi).
\end{aligned}
$$

Table 3: Single-track model parameters.

description	**symbol**	**value**	**unit**
vehicle mass	m	1093.3	kg
moment of inertia (yaw)	I_z	1791.6	kg m^2
distance from center of gravity to front axle	l_f	1.1562	m
distance from center of gravity to rear axle	l_r	1.4227	m
height of center of gravity above ground	h	0.6137	m
cornering stiffness coefficient	$C_{S,i}$	20.898	1/rad
friction coefficient	μ	$[0.8, 1]$	—

The high-order model S_I is taken out of [2, Appendix A]. Unlike the bicycle model, this model considers the vertical load of all 4 wheels due to roll, pitch, and yaw, their individual spin and slip, and a nonlinear tire model (we use the PAC2002 Magic-Formula tire model [49]). The high-order model has 29 state variables and is a standard model of the CommonRoad (commonroad.in.tum.de) benchmark suite.

To follow a given reference trajectory, the same vehicle controller similar to [5, Sec. III.C] is used for S_A and S_I. Because the input is the steering velocity instead of the steering angle as in [5, Sec. III.C], an additional P-controller for the steering velocity is used. The measurement uncertainty is combined in the vector $v = [v_x,\, v_y,\, v_\Psi,\, v_{\dot{\Psi}},\, v_v]^T \in [-1,1]0.08 \times [-1,1]0.08 \times [-1,1]0.2\pi/180 \times [-1,1]0.2\pi/180 \times [-1,1]0.08$. The reference values for the control are denoted by a subscripted d and are held constant between sensor updates. We can now state the controller outputs as

$$
\begin{aligned}
\delta_d =& k_1 \Big(\cos(\Psi_d)(s_{y,d} - s_y - v_y) - \sin(\Psi_d)(s_{x,d} - s_x - v_x) \Big) \\
& + k_2(\Psi_d - \Psi - v_\Psi) + k_3(\dot{\Psi}_d - \dot{\Psi} - v_{\dot{\Psi}}), \\
\dot{\delta} =& k_6(\delta_d - \delta), \\
a_{\mathrm{long}} =& k_4 \Big(\cos(\Psi_d)(s_{x,d} - s_x - v_x) + \sin(\Psi_d)(s_{y,d} - s_y - v_y) \Big) + k_5(\hat{v}_d - \hat{v} - v_v).
\end{aligned}
$$

The controller parameter vector is chosen as $k = [0.2,\ 2,\ 0.3,\ 1,\ 10,\ 10]$.

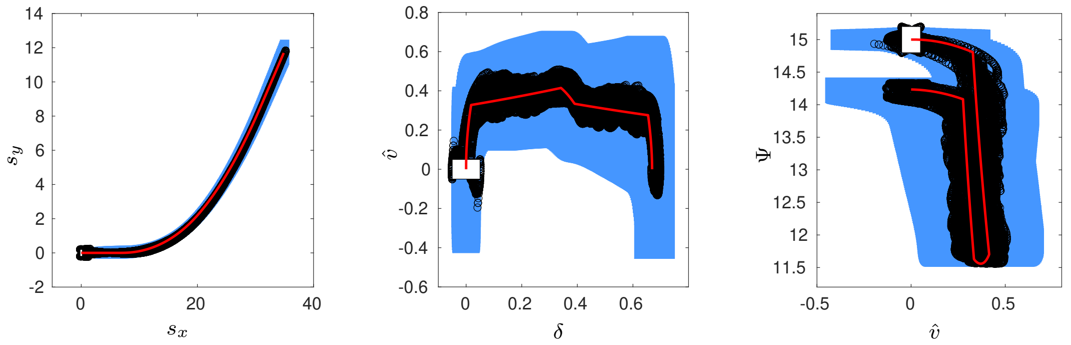
4.2.2 Numerical Results

Our test suite consists of an evasive maneuver, a so-called moose test, and a cornering maneuver. The reference trajectory is provided by the corresponding maneuver and the rapidly-exploring random trees are used to explore the reachable space of S_I by varying the initial state, the disturbances, and the measurement errors.

Using the test suite and the high-fidelity reference model, we call (less relevant commands are not shown for brevity)

```
options.alg = conformanceCheckRRT;
options.refModel = CommonRoadMB2;
params.testSuite = ACC2012Test;
res = conform(vehModelACC2012,params,options);
```

to check whether the model is reachset conformant using the methods presented in Sec. 2 (`res = 1` if the model is reachset conformant). The required additive disturbance to establish reachset conformance is $\mathcal{W} = 0 \times [0, 0.1] \times 0 \times [-0.2, 0.3] \times 0 \times 0 \times [0, 0.1]$. The result of the cornering maneuver is illustrated in Fig. 6 for selected projections.



(a) Projection onto s_x, s_y. (b) Projection onto δ, $\hat{v}$. (c) Projection onto $\hat{v}$, Ψ.

Figure 6: Selected projections of the reachable set for the cornering maneuver. The white box shows the set of initial states, black circles show states of the rapidly-exploring random tree, and the blue region is the reachable set. In subfigure (a), the initial set is small compared to the reachable set and centered at $s_x = 0$ and $s_y = 0$.

4.3 Linear Reachset Conformant Model of a Real System

In the third use case, we synthesize a reachset conformant model of pedestrians with respect to recorded movements. Reachset conformant models of pedestrians are essential for safe human-robot co-existence of mobile robots in public spaces [45] or safe autonomous driving [38]. Related work on reachset conformant models of humans considering their pose is presented in [8] and of impact forces caused by contacts with robots are considered in [44].

4.3.1 Modeling

Let us first introduce the positions s_x and s_y, the velocities $\hat{v}_x$ and $\hat{v}_y$, and accelerations a_x and a_y (all in x-direction and y-direction). Since the acceleration is unknown, it is modeled as a disturbance. The used acceleration-constrained model is

$$\dot{p}_x = \hat{v}_x, \qquad \dot{p}_y = \hat{v}_y, \qquad \dot{\hat{v}}_x = a_x, \qquad \dot{\hat{v}}_y = a_y. \tag{27}$$

The disturbance set $\mathcal{W}$ containing the accelerations is assumed to be approximately circular so that we use a zonotope template with 20 generators whose directions are uniformly distributed. As measurements, we have the positions p_x and p_y and we assume that the measurement uncertainty is independent in each direction so that the zonotope template for $\mathcal{V}$ has a generator in x-direction and another one in y-direction. The same template is used for the set of initial states $\tilde{\mathcal{X}}_0$, except that two additional generators are added, one for the velocity in x-direction and one for the y-direction.

4.3.2 Numerical Results

We synthesize the reachset-conformant model using ground-truth trajectories from a recorded street scene in Zurich, Switzerland [53]. We consider a time horizon of 2 [s], which is larger than the largest time required in the online verification scheme of [45].

Using the recorded data, we call (less relevant commands are not shown for brevity)

```
options.alg = conformanceSynthesis;
params.testSuite = Pellegrini2009Test;
paramsConform = conform(pedAccModel,params,options);
```

to obtain the sets $\tilde{\mathcal{X}}_0$, $\mathcal{W}$, and $\mathcal{V}$ stored in `paramsConform` so that the acceleration model is reachset conformant using the methods presented in Sec. 3. The resulting reachable sets for the interval norm ($\sigma = \mathbf{1}_2$) and the Frobenius norm ($P = I_2$) as well as the corresponding recorded data are presented in Fig. 7. All time steps are weighted equally so that ω in (13) and (23) is a vector of ones.

4.4 Nonlinear Reachset Conformant Model of a Real System

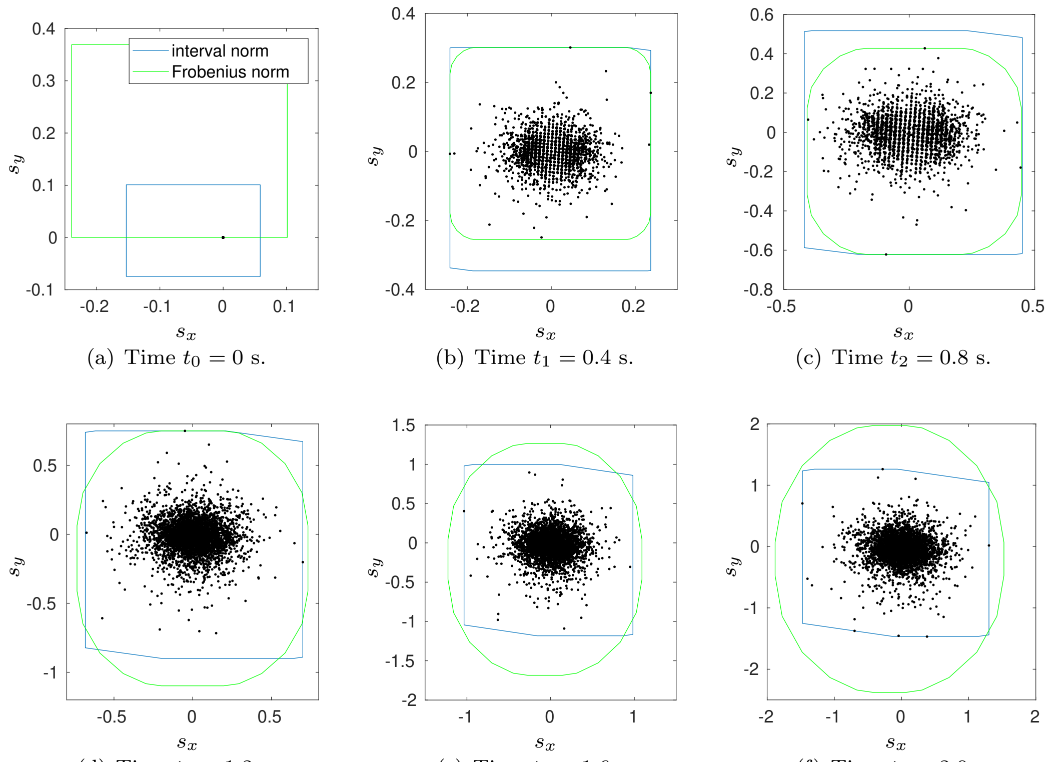
The fourth and last use case from [43] synthesizes a reachset conformant model for a six-degree-of-freedom Schunk LWA 4P robot shown in Fig. 8. Reachset conformant models of robots are essential for the online verification of safe human-robot co-existence; see, e.g., [8]. Besides the presented method using reachset conformance, related works on nonlinear models exist that are conformant with respect to real systems using approximate trace conformance [23, 61].

4.4.1 Modeling

The modeling of robotic manipulators is quite involved, so instead of presenting the model, we refer to [43, Sec. IV]. Three models are derived in that work, and we use Model 1.

4.4.2 Numerical Results

We use 152 recorded test suites, each obtained from a fixed trajectory of motor torques that are applied to the real robot 15 times from the same initial state. Each test suite is up to five seconds long.



(a) Time $t_0 = 0$ s. (b) Time $t_1 = 0.4$ s. (c) Time $t_2 = 0.8$ s.

(d) Time $t_3 = 1.2$ s. (e) Time $t_4 = 1.6$ s. (f) Time $t_5 = 2.0$ s.

Figure 7: Reachable sets of the conformant models obtained by using the interval norm (blue) and Frobenius norm (green). Dots show recorded data points.

Using the recorded data, we call (less relevant commands are not shown for brevity)

```
options.alg = conformanceSynthesis;
params.testSuite = Schunk-LWA-4P-Test;
paramsConform = conform(Schunk-LWA-4P-1,params,options);
```

to generate a new set of parameters `paramsConform` establishing reachset conformance by using the approach sketched in Sec. 3.3. The returned parameters `paramsConform` are shown in Tab. 4.

5 Conclusions

We present the first tool for checking and establishing reachset conformance of systems and models. This makes it possible to transfer safety properties to real systems – otherwise, one would only verify models. Reachset conformance is also relevant in the development process to ensure that a model used later in the development process conforms to all safety regulations. The alternative approach would be to re-verify the refined model against all specifications resulting in potentially huge costs. Our numerical examples demonstrate the usefulness of reachset conformance for various relevant use cases.

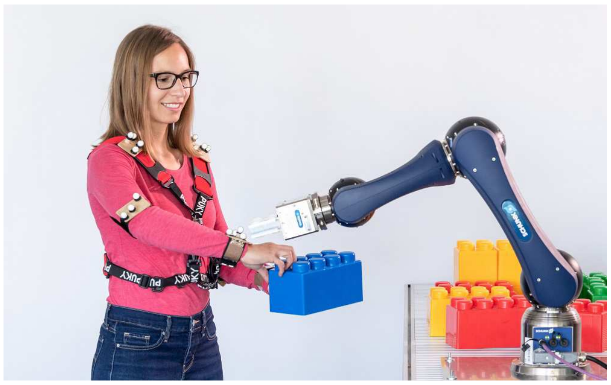


Figure 8: Schunk LWA 4P robot used in the experiments.

Table 4: Required intervals of the initial position, initial velocity, and torque disturbance for each joint to establish reachset conformance.

joint	**initial position [rad]**	**initial velocity [rad/s]**	**torque disturbance [Nm]**
1	$[-0.0030, 0.0030]$	$[-0.0317, 1.2140]$	$[-2.7201, 3.6017]$
2	$[-0.0017, 0.0017]$	$[-0.6586, 0.2550]$	$[-2.3225, 7.1559]$
3	$[-0.0025, 0.0025]$	$[-0.0154, 0.0463]$	$[-6.8833, 2.6287]$
4	$[-0.0027, 0.0027]$	$[-0.0184, 0.0331]$	$[-1.7374, 2.0317]$
5	$[-0.0075, 0.0075]$	$[-0.1179, 0.0551]$	$[-1.4919, 0.5486]$
6	$[-0.0063, 0.0063]$	$[-0.0765, 0.0765]$	$[-1.0060, 1.0060]$

Acknowledgment

The author gratefully acknowledges financial support from the European Commission project justITSELF under grant number 817629 and the Bavarian Research Foundation project STROM under grant number AZ-1473-20.

References

[1] A. Aerts, M. R. Mousavi, and M. Reniers. A tool prototype for model-based testing of cyber-physical systems. In *Proc. of Theoretical Aspects of Computing*, page 563–572. Springer, 2015.

[2] R. W. Allen, H. T. Szostak, D. H. Klyde, T. J. Rosenthal, and K. J. Owens. Vehicle dynamic stability and rollover. Final Report DOT HS 807 956, U.S. Department of Transportation, 1992.

[3] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars.* Dissertation, Technische Universität München, 2010. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20100715-963752-1-4.

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

[5] M. Althoff and J. M. Dolan. Reachability computation of low-order models for the safety verification of high-order road vehicle models. In *Proc. of the American Control Conference*, page 3559–3566, 2012.

[6] M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

[7] 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.

[8] M. Althoff, A. Giusti, S. B. Liu, and A. Pereira. Effortless creation of safe robots from modules through self-programming and self-verification. *Science Robotics*, 4(31):1–14, 2019.

[9] M. Althoff, M. Koschi, and S. Manzinger. CommonRoad: Composable benchmarks for motion planning on roads. In *Proc. of the IEEE Intelligent Vehicles Symposium*, page 719–726, 2017.

[10] M. Althoff and B. H. Krogh. Zonotope bundles for the efficient computation of reachable sets. In *Proc. of the 50th IEEE Conference on Decision and Control*, page 6814–6821, 2011.

[11] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010.

[12] R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional refinement for hierarchical hybrid systems. In *Hybrid Systems: Computation and Control*, page 33–48, 2001.

[13] R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional modeling and refinement for hierarchical hybrid systems. *The Journal of Logic and Algebraic Programming*, 68(1):105–128, 2006.

[14] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In *Proc. of the 9th International Conference on Concurrency Theory*, page 163–178, 1998.

[15] C. Baier and J.-P. Katoen. *Principles of Model Checking.* MIT Press, 2008.

[16] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha. The system-level simplex architecture for improved real-time embedded system safety. In *Proc. of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, page 99–107, 2009.

[17] L. Benvenuti, D. Bresolin, P. Collins, A. Ferrari, L. Geretti, and T. Villa. Assume-guarantee verification of nonlinear hybrid systems with ARIADNE. *International Journal of Robust and Nonlinear Control*, 24:699–724, 2014.

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

[19] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of Computer-Aided Verification*, LNCS 8044, page 258–263. Springer, 2013.

[20] C. Combastel. Zonotopes and Kalman observers: Gain optimality under distinct uncertainty paradigms and robust convergence. *Automatica*, 55:265–273, 2015.

[21] T. Dang. *Model-Based Testing for Embedded Systems*, chapter Model-based Testing of Hybrid Systems, page 383–423. CRC Press, 2011.

[22] T. Dang and T. Nahhal. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2):183–213, 2009.

[23] A. El-Guindy, D. Han, and M. Althoff. Formal analysis of drum-boiler units to maximize the load-following capabilities of power plants. *IEEE Transactions on Power Systems*, 31(6):4691–4702, 2016.

[24] C. Fan, B. Qi, S. Mitra, M. Viswanathan, and P. S. Duggirala. Automatic reachability analysis for nonlinear hybrid models with C2E2. In *Computer Aided Verification*, page 531–538, 2016.

[25] G. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations.* PhD thesis, Radboud Universiteit Nijmegen, 2005.

[26] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *Hybrid Systems: Computation and Control*, LNCS 3413, page 258–273. Springer, 2005.

[27] 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, page 379–395. Springer, 2011.

[28] G. Frehse, Z. Han, and B. Krogh. Assume-guarantee reasoning for hybrid I/O-automata by over-approximation of continuous interaction. In *Proc. of the 43rd IEEE Conference on Decision and Control*, volume 1, page 479–484, 2004.

[29] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In *Proc. of Automated Deduction*, page 527–538. Springer, 2015.

[30] S. Gao, S. Kong, and E. M. Clarke. dReal: An SMT solver for nonlinear theories of the reals. In *Proc. of the 24th International Conference on Automated Deduction*, page 208–214. Springer, 2013.

[31] A. Girard and G. J. Pappas. Approximate bisimulations for constrained linear systems. In *Proc. of the 44th IEEE Conference on Decision and Control*, page 4700–4705, 2005.

[32] A. Girard and G. J. Pappas. Approximate bisimulations for nonlinear dynamical systems. In *Proc. of the 44th IEEE Conference on Decision and Control*, page 684–689, 2005.

[33] A. Girard and G. J. Pappas. Approximate bisimulation relations for constrained linear systems. *Automatica*, 43:1307–1317, 2007.

[34] E. Haghverdi, P. Tabuada, and G. J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, 342(2):229–261, 2005.

[35] T. A. Henzinger, M. Minea, and V. Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 2034, page 275–290, 2001.

[36] F. Immler. Tool presentation: Isabelle/HOL for reachability analysis of continuous systems. In *Proc. of the 2nd Workshop on Applied Verification for Continuous and Hybrid Systems.*, page 180–187, 2015.

[37] N. Kochdumper, A. Tarraf, M. Rechmal, M. Olbrich, L. Hedrich, and M. Althoff. Establishing reachset conformance for the formal analysis of analog circuits. In *Proc. of the 25th Asia and South Pacific Design Automation Conference*, page 199–204, 2020.

[38] M. Koschi, C. Pek, and M. Althoff. Set-based prediction of pedestrians in urban environments considering formalized traffic rules. In *Proc. of the 21st IEEE International Conference on Intelligent Transportation Systems*, page 2704–2711, 2018.

[39] A. Kulmburg and M. Althoff. On the co-NP-completeness of the zonotope containment problem. *European Journal of Control*, 62:84–91, 2021.

[40] W. Kühn. Rigorously computed orbits of dynamical systems without the wrapping effect. *Computing*, 61:47–67, 1998.

[41] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computation for families of linear vector fields. *Symbolic Computation*, 32:231–253, 2001.

[42] H.-S. L. Lee, M. Althoff, S. Hoelldampf, M. Olbrich, and E. Barke. Automated generation of hybrid system models for reachability analysis of nonlinear analog circuits. In *Proc. of the 20th Asia and South Pacific Design Automation Conference*, page 725–730, 2015.

[43] S. B. Liu and M. Althoff. Reachset conformance of forward dynamic models for the formal analysis of robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 370–376, 2018.

[44] S. B. Liu and M. Althoff. Online verification of impact-force-limiting control for physical human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 777–783, 2021.

[45] S. B. Liu, H. Roehm, C. Heinzemann, I. Lütkebohle, J. Oehlerking, and M. Althoff. Provably safe motion of mobile robots in human environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 1351–1357, 2017.

[46] S. B. Liu, B. Schürmann, and M. Althoff. Guarantees for real robotic systems: Unifying formal controller synthesis and reachset-conformant identification. *IEEE Transactions on Robotics*, 2023. early access.

[47] N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata revisited. In *Proc. of Hybrid Systems: Computation and Control*, page 403–417, 2001.

[48] S. Mitsch and A. Platzer. ModelPlex: Verified runtime validation of verified cyber-physical system models. *Formal Methods in System Design*, 49(1):33–74, 2016.

[49] MSC Software, 2 MacArthur Place, Santa Ana, CA 92707. *Adams/Tire help*, April 2011. Documentation ID: DOC9805.

[50] A. Murthy, Md. A. Islam, S. A. Smolka, and R. Grosu. Computing bisimulation functions using SOS optimization and δ-decidability over the reals. In *Proc. of the 18th International Conference on Hybrid Systems: Computation and Control*, page 78–87, 2015.

[51] T. Nahhal. *Model-Based Testing of Hybrid Systems*. PhD thesis, Universit Joseph Fourier - Grenoble 1, 2007.

[52] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, 2003.

[53] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Proc. of the 12th IEEE International Conference on Computer Vision*, page 261–268, 2009.

[54] G. Pola, A. J. van der Schaft, and M. D. Di Benedetto. Bisimulation theory for switching linear systems. In *Proc. of the 43rd IEEE Conference on Decision and Control*, volume 2, page 1406–1411, 2004.

[55] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. Introducing SOSTOOLS: a general purpose sum of squares programming solver. In *Proc. of the 41st IEEE Conference on Decision and Control*, volume 1, page 741–746 vol.1, 2002.

[56] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff. STL model checking of continuous and hybrid systems. In *Proc. of the 14th International Symposium on Automated Technology for Verification and Analysis*, page 412–427, 2016.

[57] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. Reachset conformance testing of hybrid automata. In *Proc. of Hybrid Systems: Computation and Control*, page 277–286, 2016.

[58] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. Model conformance for cyber-physical systems: A survey. *ACM Transactions on Cyber-Physical Systems*, 3(3):Article 30, 2019.

[59] H. Roehm, A. Rausch, and M. Althoff. Reachset conformance and automatic model adaptation for hybrid systems. *Mathematics*, 10(19):Article 3567, 2022.

[60] J. Rohn. Inverse interval matrix. *SIAM Journal on Numerical Analysis*, 30:864–870, 1993.

[61] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff. Ensuring drivability of planned motions using formal methods. In *Proc. of the 20th IEEE International Conference on Intelligent Transportation Systems*, page 1–8, 2017.

[62] J. K. Scott, D. M. Raimondo, G. R. Marseglia, and R. D. Braatz. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69:126–136, 2016.

[63] P. Tabuada. *Verification and Control of Hybrid Systems*. Springer, 2009.

[64] P. Tabuada and G. J. Pappas. Bisimilar control affine systems. *Systems & Control Letters*, 52(1):49–58, 2004.

[65] H. G. Tanner and G. J. Pappas. Abstractions of constrained linear systems. In *Proc. of the American Control Conference*, volume 4, page 3381–3386, 2003.

[66] A. Tarraf and L. Hedrich. Behavioral modeling of transistor-level circuits using automatic abstraction to hybrid automata. In *Proc. of Design, Automation & Test in Europe Conference & Exhibition*, page 1451–1456, 2019.

[67] A. van der Schaft. Bisimulation of dynamical systems. In *Hybrid Systems: Computation and Control*, page 555–569. Springer, 2004.

[68] A. J. van der Schaft. Equivalence of dynamical systems by bisimulation. *IEEE Transactions on Automatic Control*, 49(12):2160–2172, 2004.

[69] M. van Osch. Hybrid input-output conformance and test generation. In *Formal Approaches to Software Testing and Runtime Verification*, page 70–84. Springer, 2006.

[70] M. P. W. J. van Osch. *Automated Model-based Testing of Hybrid Systems.* PhD thesis, Technische Universiteit Eindhoven, 2009.

[71] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. M. De Moor. A note on persistency of excitation. *Systems & Control Letters*, 54(4):325–329, 2005.

A Proof of Theorem 1

We first introduce the Minkowski sum of two zonotopes $\mathcal{Z}_1 = \langle c, G\rangle$ and $\mathcal{Z}_2 = \langle d, H\rangle$ as well as the multiplication of a matrix $M \in \mathbb{R}^{o\times n}$ with a zonotope $\mathcal{Z}_1$, which are a direct consequence of the zonotope definition (see [40]):

$$\begin{aligned} \mathcal{Z}_1 \oplus \mathcal{Z}_2 &= \langle c + d, [G,\, H]\rangle\,, \\ M\,\mathcal{Z}_1 &= \langle M\,c, M\,G\rangle\,. \end{aligned} \tag{28}$$

Let us rewrite (8) in zonotope notation. After inserting (9) in (8) and using (28) as well as (10), we obtain

$$\forall y(t_k) \in \mathcal{Y}(t_k) : y(t_k) - \tilde{y}(t_k, x_0, u(\cdot)) \in \underbrace{\langle \Gamma(t_k)\, c, G(t_k)\rangle}_{=\mathcal{R}(t_k,x_0,u(\cdot))-\tilde{y}(t_k,x_0,u(\cdot))}\,, \tag{29}$$

where $G(t_k)$ is from (21) and

$$\Gamma(t_k) = \begin{cases} \begin{bmatrix}\Lambda_k, & \sum_{i=0}^{k-1}\Lambda_i, & I_m\end{bmatrix}, \text{ for } k \geq 1 \\ \begin{bmatrix}\Lambda_k, & \mathbf{0}, & I_m\end{bmatrix}, \text{ for } k = 0 \end{cases}.$$

Next, we derive the constraint of the linear program, followed by the cost function.

Constraint In order to write the constraint as shown in (12), we translate the generator representation in (29) into the halfspace representation using [11, Thm. 7], so that we obtain

$$\forall y(t_k) \in \mathcal{Y}(t_k) : N(t_k)\big(y(t_k) - \tilde{y}(t_k, x_0, u(\cdot))\big) \leq d(t_k),$$

where the row vectors of $N(t_k)$ are the normal vectors of the halfspaces as computed in [11, Thm. 7] and the elements of $d(t_k)$ are the offsets of the halfspaces. To check the above inequality for all measurements $y(t_k) \in \mathcal{Y}(t_k)$, we only have to check the measurements which are closest to each halfspace:

$$\max_{y(t_k)\in\mathcal{Y}(t_k)} N(t_k)\big(y(t_k) - \tilde{y}(t_k, x_0, u(\cdot))\big) \leq d(t_k), \tag{30}$$

where the max operator is applied elementwise. The normal vectors are unaffected by c and α as shifting a zonotope and stretching its generators does not change its normal vectors (see proof of [46, Thm. 1]). However, the offset of the halfspaces encoded in $d(t_k)$ is affected. From [11, Thm. 7] we have that

$$d(t_k) = N(t_k)\Gamma(t_k)c + |N(t_k)G(t_k)|\mathbf{1}_{p+k\cdot q+r} \tag{31}$$

assuming that the absolute value is taken elementwise. It will be convenient to use

$$|Q\,\mathrm{diag}(\alpha_Q)|\mathbf{1}_n = |Q|\alpha_Q, \quad Q \in \mathbb{R}^{m\times n}, \alpha_Q \in \mathbb{R}^n_{\geq 0}. \tag{32}$$

By assuming $\alpha \in \mathbb{R}^{p+q+r}_{\geq 0}$ without loss of generality, we have that

$$\begin{aligned}
&|N(t_k)\, G(t_k)|\mathbf{1}_{p+k\cdot q+r} \\
\overset{(21)}{=}&\big|N(t_k)\,\Lambda_k\, G_X\, \mathrm{diag}(\alpha_X)\big|\mathbf{1}_p+ \\
&\big|N(t_k)[\Lambda_0\, G_W\, \mathrm{diag}(\alpha_W),\, \dots,\, \Lambda_{k-1}\, G_W\, \mathrm{diag}(\alpha_W)]\big|\mathbf{1}_{k\cdot q}+ \\
&\big|N(t_k)\, G_V\, \mathrm{diag}(\alpha_V)\big|\mathbf{1}_r \\
=&\big|N(t_k)\,\Lambda_k\, G_X\, \mathrm{diag}(\alpha_X)\big|\mathbf{1}_p + \sum_{i=0}^{k-1}\big|N(t_k)\,\Lambda_i\, G_W\, \mathrm{diag}(\alpha_W)\big|\mathbf{1}_q + \big|N(t_k)\, G_V\, \mathrm{diag}(\alpha_V)\big|\mathbf{1}_r \\
\overset{(32)}{=}&\big|N(t_k)\,\Lambda_k\, G_X\big|\alpha_X + \sum_{i=0}^{k-1}\big|N(t_k)\,\Lambda_i\, G_W\big|\alpha_W + \big|N(t_k)\, G_V\big|\alpha_V \\
\overset{(10)}{=}&\Big[|N(t_k)\,\Lambda_k\, G_X|, \quad \textstyle\sum_{i=0}^{k-1}|N(t_k)\,\Lambda_i\, G_W|, \quad |N(t_k)\, G_V|\Big]\,\alpha \\
\overset{(20)}{=}&\tilde{G}(N(t_k), t_k)\,\alpha.
\end{aligned} \tag{33}$$

After inserting (33) in (31) and using (14), we obtain

$$d(t_k) = \big[N(t_k)\Gamma(t_k), \quad \tilde{G}(N(t_k), t_k)\big]\, z. \tag{34}$$

Inserting (34) in (30) results in

$$\begin{aligned}
\max_{y(t_k)\in\mathcal{Y}(t_k)} N(t_k)\big(y(t_k) - \tilde{y}(t_k, x_0, u(\cdot))\big) &\leq \big[N(t_k)\Gamma(t_k), \quad \tilde{G}(N(t_k), t_k)\big]\, z \\
\overset{(18),(19)}{\iff}\ \tilde{A}(t_k)\, z &\leq \tilde{b}(t_k).
\end{aligned}$$

To obtain the constraint for all time steps, we add the constraint for each time step, resulting in the constraint in (12) using the shorthands introduced in (16) and (17). This constraint also includes $[\mathbf{0}, -I_{p+q+r}]z \leq \mathbf{0}$ to ensure that $\alpha \in \mathbb{R}^{p+q+r}_{\geq 0}$ as required in this proof. It remains to derive the cost function.

Cost Function Inserting (29) in (13) results in

$$\begin{aligned}
&\sum_{k=0}^{N}\omega_k\|\langle\Gamma(t_k)\, c, G(t_k)\rangle\|_{I,\sigma} \\
\overset{(11)}{\iff}&\sum_{k=0}^{N}\omega_k\sigma^T|G(t_k)|\mathbf{1}_{p+k\cdot q+r} \\
\overset{(33),N=I_m}{\iff}&\sum_{k=0}^{N}\omega_k\sigma^T\tilde{G}(I_m, t_k)\alpha \\
\overset{(14)}{\iff}&\Big[\mathbf{0}, \quad \textstyle\sum_{k=0}^{N}\omega_k\sigma^T\tilde{G}(I_m, t_k)\Big]\, z \\
\iff& n^T z.
\end{aligned}$$

B Proof of Theorem 2

It will be convenient to use

$$\begin{aligned}
&\operatorname{trace}\left(\left[Q\operatorname{diag}(\alpha_Q),\quad R\operatorname{diag}(\alpha_R)\right]^T\, P\, \left[Q\operatorname{diag}(\alpha_Q),\quad R\operatorname{diag}(\alpha_R)\right]\right)\\
=&\operatorname{trace}\left(\operatorname{diag}(\left[\alpha_Q^T,\quad \alpha_R^T\right])\begin{bmatrix}Q^T\\R^T\end{bmatrix}\, P\, \left[Q,\quad R\right]\, \operatorname{diag}\left(\left[\alpha_Q^T,\quad \alpha_R^T\right]\right)\right)\\
=&\left[\alpha_Q^T,\quad \alpha_R^T\right]\,\widetilde{\operatorname{diag}}\left(\begin{bmatrix}Q^T\\R^T\end{bmatrix}\, P\, \left[Q,\quad R\right]\right)\begin{bmatrix}\alpha_Q\\\alpha_R\end{bmatrix}\\
=&\alpha_Q^T\,\widetilde{\operatorname{diag}}(Q^T\,P\,Q)\,\alpha_Q+\alpha_R^T\,\widetilde{\operatorname{diag}}(R^T\,P\,R)\,\alpha_R.
\end{aligned}\tag{35}$$

Furthermore, we will use the equivalence

$$\begin{aligned}
&\alpha_Q^T\,\widetilde{\operatorname{diag}}(Q^T\,P\,Q)\,\alpha_Q+\alpha_R^T\,\widetilde{\operatorname{diag}}(R^T\,P\,R)\,\alpha_R\\
=&\left[\alpha_Q^T\quad \alpha_R^T\right]\,\operatorname{blkdiag}\left(\widetilde{\operatorname{diag}}(Q^T\,P\,Q),\,\widetilde{\operatorname{diag}}(R^T\,P\,R)\right)\begin{bmatrix}\alpha_Q\\\alpha_R\end{bmatrix}.
\end{aligned}\tag{36}$$

We have that

$$\begin{aligned}
&\operatorname{trace}(G(t_k)^T P\, G(t_k))\\
\overset{(21),(35)}{=}&\alpha_X^T\,\widetilde{\operatorname{diag}}\left(G_X^T\Lambda_k^T\,P\,\Lambda_k\,G_X\right)\alpha_X+\\
&\alpha_W^T\,\widetilde{\operatorname{diag}}\left(G_W^T\Lambda_0^T\,P\,\Lambda_0\,G_W\right)\alpha_W+\ldots+\alpha_W^T\,\widetilde{\operatorname{diag}}\left(G_W^T\Lambda_{k-1}^T\,P\,\Lambda_{k-1}\,G_W\right)\alpha_W+\\
&\alpha_V^T\,\widetilde{\operatorname{diag}}(G_V^T\,P\,G_V)\alpha_V\\
\overset{(26)}{=}&\alpha_X^T\,\widetilde{\operatorname{diag}}\left(G_X^T\Lambda_k^T\,P\,\Lambda_k\,G_X\right)\alpha_X+\alpha_W^T\,\tilde{H}_W(t_k)\alpha_W+\alpha_V^T\,\widetilde{\operatorname{diag}}\left(G_V^T\,P\,G_V\right)\alpha_V\\
\overset{(36),(10)}{=}&\alpha^T\,\operatorname{blkdiag}\left(\widetilde{\operatorname{diag}}\left(G_X^T\Lambda_k^T\,P\,\Lambda_k\,G_X\right),\,\tilde{H}_W(t_k),\,\widetilde{\operatorname{diag}}\left(G_V^T\,P\,G_V\right)\right)\alpha\\
\overset{(25)}{=}&\alpha^T\,\tilde{H}(t_k)\,\alpha.
\end{aligned}\tag{37}$$

Inserting (29) in (23) results in

$$\begin{aligned}
&\sum_{k=0}^N \omega_k\|\langle\Gamma(t_k)c,G(t_k)\rangle\|_{F,P}^2\\
\overset{(22)}{\iff}&\sum_{k=0}^N \omega_k\operatorname{trace}(G(t_k)^T P\, G(t_k))\\
\overset{(37)}{\iff}&\sum_{k=0}^N \omega_k\,\alpha^T\,\tilde{H}(t_k)\,\alpha\\
\iff&\alpha^T\left(\sum_{k=0}^N \omega_k\,\tilde{H}(t_k)\right)\alpha\\
\overset{(14)}{\iff}&z^T\begin{bmatrix}\mathbf{0}&\mathbf{0}\\\mathbf{0}&\sum_{k=0}^N\omega_k\,\tilde{H}(t_k)\end{bmatrix}z\\
\overset{(24)}{\iff}&z^T H z.
\end{aligned}$$