Reachability analysis aims at identifying states reachable by a system within a given time horizon. This task is known to be computationally hard for hybrid systems. One of the main challenges is the handling of discrete transitions, including computation of intersections with invariants and guards. In this paper, we address this problem by proposing a state-space decomposition approach for linear hybrid systems. This approach allows us to perform most operations in low-dimensional state space, which can lead to significant performance improvements.

1 Introduction

A hybrid system is a formalism for modeling cyber-physical systems. Reachability analysis is a rigorous way to reason about the behavior of hybrid systems.

In this paper, we describe a new reachability algorithm for linear hybrid systems, i.e., hybrid systems with dynamics given by linear differential equations and constraints given by linear inequalities. The key feature of our algorithm is that it works in low dimensions, which enables compositional analysis with high scalability. To this end, we integrate our recent reachability algorithm for (purely continuous) LTI systems, which we call Post\textsuperscript{C} in the following, in a new algorithm for linear hybrid systems.

The Post\textsuperscript{C} algorithm decomposes the calculation of the reachable states into calculations in subspaces (called “blocks”). This decomposition has two benefits. The first benefit is that computations in lower dimensions are generally more

---

* This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award) and the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 754411 and the Air Force Office of Scientific Research under award number FA2386-17-1-4065. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.
efficient (and thus the algorithm is highly scalable). The second benefit is that the analysis for different subspaces is decoupled; hence one can effectively skip the computations for dimensions that are of no interest (e.g., for a safety property).

Extending algorithms from purely continuous systems to hybrid systems is conceptually easy by adding a “hybrid loop” that interleaves a continuous-post algorithm and a discrete-post algorithm. If we consider $Post_C^\square$ as a black box, we can plug it into this hybrid loop, which we refer to as $Post_H^H$ (cf. Section 3). However, there are two shortcomings. First, all operations aside from $Post_C^\square$ are still performed in high dimensions, and so $Post_H^H$ still suffers from scalability issues. Second, $Post_H^H$ does not make use of the decoupling of $Post_C^\square$ at all.

In this work, we present the first truly decomposed algorithm for general linear hybrid systems. We demonstrate that, unlike in $Post_H^H$, it is possible to perform all computations in low dimensions (cf. Section 4). Surprisingly, we show that, in common cases, there is not even an additional approximation error. Furthermore, our algorithm makes proper use of the second benefit of $Post_C^\square$ by computing the reachable states only in specific dimensions whenever possible.

We have implemented the algorithm in JuliaReach, a toolbox for reachability analysis [13,2], and we evaluate the potential of our algorithm on several benchmark problems, including a 1024-dimensional hybrid system (cf. Section 5). Our algorithm outperforms the naive $Post_H^H$ by one order of magnitude.

To summarize, we show how to integrate the decomposition-based approach from [12] into a decomposed reachability algorithm for linear hybrid systems. The key insights are (1) to exploit the decomposed structure of the reachable states to perform all operations in low dimensions and (2) to only compute the reachable states in specific dimensions as often as possible.

**Related work**

**Decomposition.** Hybrid systems given as a network of components can be explored efficiently in a symbolic way, e.g., using bounded model checking [14]. We consider a decomposition in the continuous state space here. Schupp et al. perform such a decomposition by syntactic independence [41], which corresponds to dynamics matrices of block-diagonal form.

For purely continuous systems there exist various decomposition approaches. In this work we build on [12] for LTI systems, which decomposes the system into blocks and exploits the linear dynamics to avoid the wrapping effect. Other approaches for LTI systems are based on Krylov subspace approximations [31], time-scale decomposition [19,26], similarity transformations [34,35], projectahedra [27,43], and sub-polyhedra abstract domains [42]. Approaches for nonlinear systems are based on projections with differential inclusions [8], Hamilton-Jacobi methods [39,10], and hybridization with iterative refinement [17].

**Lazy flowpipe computation.** The support-function representation of convex sets can be used to represent a flowpipe (a sequence of sets that covers the behaviors of a system) symbolically [24,11,29]. Only sets that are of interest, e.g.,
those that intersect with a constraint, need then be approximated \[21\]. Using our decomposition approach, we can even avoid the symbolic computation in dimensions that are irrelevant to the intersection. Our approach is independent of the set representation, so it can also be applied in analyses based on, e.g., zonotopes \[23,25,6,4\]. Given a linear switching system with a hyperplanar state-space partition, Hamadeh and Goncalves compute ellipsoidal over- and underapproximations of the reach set on the partition borders, without computing the full reach set \[30\].

Intersection of convex sets. Performing intersections in low dimensions allows for efficient computations that are not possible in high dimensions. For example, checking for emptiness of a polyhedron in constraint representation is a feasibility linear program, which can be solved in weakly polynomial time, but solutions in strongly polynomial time are only known in two dimensions \[33\]. In the context of hybrid-system reachability, computing intersections is considered a major challenge because it usually requires a conversion from efficient state representations (like zonotopes, support function, or Taylor models) to polytopes and back, which often involves an additional approximation. Below we summarize how other approaches tackle the intersection problem.

A coarse approximation of the intersection with a guard can be obtained by only detecting a nonempty intersection (which is generally easier to do) and then taking the original set as overapproximation \[40\]. In general, the intersection between a polytope and polyhedral constraints (invariants and guards) can be computed exactly, but such an approach is not scalable \[15\]. Girard and Le Guernic consider hyperplanar constraints where reachable states are either zonotopes, in which case they work in a two-dimensional projection \[24\], or general polytopes \[28,36\]. The tool SpaceEx approximates the intersection of polytopes and general polyhedra using template directions \[20\]. Frehse and Ray propose an optimization scheme for the intersection of a compact set \(\mathcal{X}\), represented by its support function, and a polyhedron \(\mathcal{Y}\), and this scheme is exact if \(\mathcal{X}\) is a polytope \[21\]. Althoff et al. approximate zonotopes by parallelotopes before considering the intersection \[6\]. For must semantics, Althoff and Krogh use constant-dynamics approximation and obtain a nonlinear mapping \[5\]. Under certain conditions, Bak et al. apply a model transformation by replacing guard constraints by time-triggered constraints, for which intersection is easy \[9\].

2 Preliminaries

We introduce some notation. The real numbers are denoted by \(\mathbb{R}\). Given two vectors \(x, y \in \mathbb{R}^n\), their dot product is \(\langle x, y \rangle := \sum_{i=1}^n x_i \cdot y_i\). For \(p \geq 1\), the \(p\)-norm of a matrix \(A \in \mathbb{R}^{n \times n}\) is denoted \(\|A\|_p\). The diameter of a set \(\mathcal{X} \subseteq \mathbb{R}^n\) is \(\Delta_p(\mathcal{X}) := \sup_{x, y \in \mathcal{X}} \|x - y\|_p\). The \(n\)-dimensional unit ball of the \(p\)-norm is \(B^p_n := \{x \in \mathbb{R}^n | \|x\|_p = 1\}\). An \(n\)-dimensional half-space is the set \(\{\langle a, x \rangle \leq b | x \in \mathbb{R}^n\}\) parameterized by \(a, b \in \mathbb{R}^n\). A polyhedron is an intersection of finitely many half-spaces, and a polytope is a bounded polyhedron.
Given two sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$, a scalar $\lambda \in \mathbb{R}$, a matrix $A \in \mathbb{R}^{n \times n}$, and a vector $b \in \mathbb{R}^n$, we use the following operations on sets: scaling $\lambda \mathcal{X} := \{ \lambda x \mid x \in \mathcal{X} \}$, linear map $A \mathcal{X} := \{ Ax \mid x \in \mathcal{X} \}$, Minkowski sum $\mathcal{X} + \mathcal{Y} := \{ x + y \mid x \in \mathcal{X} \text{ and } y \in \mathcal{Y} \}$ (if $n = m$), affine map $(A, b) \circ \mathcal{X} := A \mathcal{X} + \{ b \}$, Cartesian product $\mathcal{X} \times \mathcal{Y} := \{ (x, y) \mid x \in \mathcal{X}, y \in \mathcal{Y} \}$, intersection $\mathcal{X} \cap \mathcal{Y} := \{ z \mid z \in \mathcal{X}, z \in \mathcal{Y} \}$ (if $n = m$), and convex hull $\text{CH}(\mathcal{X}) := \{ \lambda \cdot x + (1 - \lambda) \cdot y \mid x, y \in \mathcal{X}, 0 \leq \lambda \leq 1 \}$.

Given two sets $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^n$, the Hausdorff distance is defined as
\[
d_H(\mathcal{X}, \mathcal{Y}) := \inf_{\varepsilon \in \mathbb{R}} \{ \mathcal{Y} \subseteq \mathcal{X} \oplus \varepsilon B^n \text{ and } \mathcal{X} \subseteq \mathcal{Y} \oplus \varepsilon B^n \}.
\]

Let $\mathcal{C}_n \subseteq 2^{\mathbb{R}^n}$ be the set of $n$-dimensional compact and convex sets. For a nonempty set $\mathcal{X} \subseteq \mathcal{C}_n$, the support function $\rho_\mathcal{X} : \mathbb{R}^n \to \mathbb{R}$ is defined as
\[
\rho_\mathcal{X}(d) := \max_{x \in \mathcal{X}} (d, x).
\]

The Hausdorff distance of two sets $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{C}_n$ with $\mathcal{X} \subseteq \mathcal{Y}$ can alternatively be expressed in terms of the support function as
\[
d_H^p(\mathcal{X}, \mathcal{Y}) = \max_{\|d\| \leq 1} \rho_\mathcal{Y}(d) - \rho_\mathcal{X}(d).
\]

Let $\{\pi_j\}_{j=1}^b$ be a set of (contiguous) projection matrices that partition a vector $x \in \mathbb{R}^n$ into $x = [\pi_1 x, \ldots, \pi_b x]$. Given a set $\mathcal{X}$, its Cartesian decomposition is $\pi_1 \mathcal{X} \times \cdots \times \pi_b \mathcal{X}$ with the block structure induced by the projection matrices $\pi_j$. We refer to $\pi_j \mathcal{X}$ as a block of $\mathcal{X}$ and typically write $\hat{\mathcal{X}}$ to indicate that a set is decomposed (i.e., a Cartesian product of lower-dimensional sets). Given a nonempty set $\mathcal{X} \subseteq \mathcal{C}_n$, its box approximation is the Cartesian decomposition into intervals. We can bound the approximation error by the radius of $\hat{\mathcal{X}}$.

**Proposition 1.** Let $\mathcal{X} \subseteq \mathcal{C}_n$ be nonempty, $p = \infty$, $r_\mathcal{X}^p$ be the radius of the box approximation of $\mathcal{X}$, and let $\pi_j$ be appropriate projection matrices. Then $d_H^p(\mathcal{X}, \pi_j \mathcal{X}) \leq r_\mathcal{X}^p$.

### 2.1 LTI systems

An $n$-dimensional **LTI system** $(A, B, \mathcal{U})$, with matrices $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and input domain $\mathcal{U} \subseteq \mathcal{C}_m$, is a system of ODEs of the form
\[
\dot{x}(t) = Ax(t) + Bu(t), \quad u(t) \in \mathcal{U}.
\]  

We denote the set of all $n$-dimensional LTI systems by $\mathcal{L}_n$. From now on, a vector $x \in \mathbb{R}^n$ is also called a (continuous) state. Given an initial state $x_0 \in \mathbb{R}^n$ and an input signal $u$ such that $u(t) \in \mathcal{U}$ for all $t$, a trajectory of (1) is the unique solution $\xi_{x_0,u} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ with
\[
\xi_{x_0,u}(t) = e^{At}x_0 + \int_0^t e^{A(t-s)}Bu(s) \, ds.
\]
Given an LTI system \((A,B,U)\) and a set \(X_0 \in \mathbb{C}^n\) of initial states, the \textit{continuous-post operator}, \(\text{Post}_C\), computes the set of reachable states for all input signals \(u\) over \(U\):

\[
\text{Post}_C((A,B,U), X_0) := \{ \xi_{x_0,u}(t) \mid t \geq 0, x_0 \in X_0, u(s) \in U \text{ for all } s \}.
\]

\section{2.2 Linear hybrid systems}

We briefly introduce the syntax of linear hybrid systems used in this work and refer to the literature for the semantics \cite{12}. An \(n\)-dimensional linear hybrid system is a tuple \(H = (\text{Var}, \text{Loc}, \text{Flow}, \text{Inv}, \text{Grd}, \text{Asgn})\) with variables \(\text{Var} = \{x_1, \ldots, x_n\}\), a finite set of locations \(\text{Loc}\), two functions \(\text{Flow} : \text{Loc} \to \mathbb{C}_n\) and \(\text{Inv} : \text{Loc} \to \mathbb{C}_n\), that respectively assign continuous dynamics and an invariant to each location, and two functions \(\text{Grd} : \text{Loc} \times \text{Loc} \to \mathbb{R}^{n \times n} \times \mathbb{R}^n\) and \(\text{Asgn} : \text{Loc} \times \text{Loc} \to \mathbb{R}^n\) that respectively assign a guard and an assignment in the form of a deterministic affine map to each pair of locations. If \(\text{Grd}((\ell, \ell')) \neq \emptyset\), we call \((\ell, \ell')\) a (discrete) transition.

Let \(H = (\text{Var}, \text{Loc}, \text{Flow}, \text{Inv}, \text{Grd}, \text{Asgn})\) be a linear hybrid system. A (symbolic) state of \(H\) is a pair \((\ell, X) \in \text{Loc} \times 2^{\mathbb{R}^n}\). The \textit{discrete-post operator}, \(\text{Post}_D\), maps a symbolic state to a set of symbolic states by means of discrete transitions:

\[
\text{Post}_D((\ell, X)) := \bigcup_{\ell' \in \text{Loc}} \{(\ell', \text{Asgn}((\ell, \ell')) \cap (X \cap \text{Inv}(\ell) \cap \text{Grd}((\ell, \ell')))) \cap \text{Inv}(\ell')\}\]

(2)

The \textit{reach set} of \(H\) from a set of initial symbolic states \(\mathcal{R}_0\) of \(H\) is the smallest set \(\mathcal{R}\) of symbolic states such that

\[
\mathcal{R}_0 \cup \bigcup_{(t, X) \in \mathcal{R}} \text{Post}_D((t, \text{Post}_C(\text{Flow}(t), X))) \subseteq \mathcal{R}.
\]

(3)

\section{3 Reachability analysis of linear hybrid systems}

Our reachability algorithm for linear hybrid systems integrates the algorithm from \cite{12}, which implements \(\text{Post}_C\) for LTI systems in a compositional way. In this section, we first recall the algorithm from \cite{12}, which from now on we call \(\text{Post}^{\square}_C\) for convenience. Two important properties of \(\text{Post}^{\square}_C\) are that (1) the output is a sequence of \textit{decomposed} sets and that (2) this sequence is computed in low dimensions.

After explaining the algorithm \(\text{Post}^{\square}_C\), we incorporate it in a standard reachability algorithm for linear hybrid systems. However, this standard reachability algorithm will not make use of the above-mentioned properties. This will motivate our new algorithm, which is a modification of this standard reachability algorithm to make optimal use of these properties (presented in the next section).
Starting from the set of initial states $X_0$ (blue set), we first compute the set $X(0)$ by time discretization (green set), then decompose the set into intervals and obtain $\tilde{X}(0)$ (orange box around $X(0)$), and finally compute the (decomposed) flowpipe $\tilde{X}(1), \ldots, \tilde{X}(4)$ by propagating each of the intervals (other orange sets). (b) The flowpipe from (a) together with a guard (red).

3.1 Decomposed reachability analysis of LTI systems

The decomposition-based approach \cite{12} follows a flowpipe-construction scheme using time discretization, which we shortly recall here. A flowpipe of length $N$ is a sequence of sets. Given an LTI system $(A,B,U)$ and a set of initial (continuous) states $X_0$, by fixing a time step $\delta$ we first compute a set $X(0)$ that overapproximates the reach set up to time $\delta$, a matrix $\Phi = e^{A\delta}$ that captures the dynamics of duration $\delta$, and a set $V$ which overapproximates the effect of the inputs up to time $\delta$. We obtain an overapproximation of the reach set in time interval $[k\delta, (k+1)\delta]$, for step $k > 0$, with

$$X(k) := \Phi X(k-1) \oplus V = \Phi^k X(0) \oplus \bigoplus_{j=0}^{k-1} \Phi^j V.$$ 

Algorithm Post$\Box$ decomposes this scheme. Fixing some block structure, let $\tilde{X}(0) := X_1(0) \times \cdots \times X_b(0)$ be the corresponding Cartesian decomposition of $X(0)$. We compute a sequence $\tilde{X}(k) := X_1(k) \times \cdots \times X_b(k)$ such that $X(k) \subseteq \tilde{X}(k)$ for every $k$. Each low-dimensional set $X_i(k)$ is computed as

$$X_i(k) := \bigoplus_{j=1}^b \Phi_{ij}^k X_j(0) \oplus \bigoplus_{j=0}^{k-1} \left[ \Phi_{i1}^j \cdots \Phi_{ib}^j \right] V$$

where $\Phi^j = \begin{pmatrix} \Phi_{i1}^j & \cdots & \Phi_{ib}^j \\ \vdots & \ddots & \vdots \\ \Phi_{i1}^j & \cdots & \Phi_{ib}^j \end{pmatrix}$.

The above sequences $X(k)$ resp. $\tilde{X}(k)$ are called flowpipes.

Example. We illustrate the algorithms with a running example throughout the paper. For illustration purposes, the example is two-dimensional, and hence we decompose into one-dimensional blocks, but we emphasize that the approach also generalizes to higher-dimensional decomposition. To keep things simple, we consider a hybrid system with only a single location and one transition (a self-loop). Figure 1(a) depicts the flowpipe construction for the example.
3.2 Reachability analysis of linear hybrid systems

We now discuss a standard reachability algorithm for hybrid systems. Essentially, this algorithm interleaves the operators $\text{Post}_C$ and $\text{Post}_D$ following (3) until it finds a fixpoint. Here we use $\text{Post}^\square_C$ as the continuous-post operator.

Using $\text{Post}^\square_C$, we first compute a flowpipe $\hat{X} = \hat{X}(0), \ldots, \hat{X}(N)$ as described above. Then we use $\text{Post}_D$ to take a discrete transition. According to (2), we want to compute $((A, b) \circ (\hat{X} \cap \mathcal{I}_1 \cap \mathcal{G})) \cap \mathcal{I}_2$, where $(A, b)$ is a deterministic affine map. Frehse and Ray showed that for such maps the term can be simplified to

$$ (A, b) \circ (\hat{X} \cap \mathcal{G}^*) $$

where the set $\mathcal{G}^*$ can be statically precomputed [21], which is usually easy because the sets $\mathcal{I}_1$, $\mathcal{G}$, and $\mathcal{I}_2$ are given as polyhedra in constraint representation. Hence we ignore invariants in the rest of the presentation.

Example. We continue in Figure 1(b) with the flowpipe from before. The guard $\mathcal{G}$ is a half-space that is constrained in dimension $x_1$ and unconstrained in dimension $x_2$. Only the last two sets in the flowpipe intersect with the guard. The assignment here is a translation in dimension $x_1$. The resulting intersection, before and after the translation, is depicted in Figure 2(a).

Finally, the algorithm checks for a fixpoint, i.e., for inclusion of the symbolic states we computed with $\text{Post}_D$ in previously-seen symbolic states.

Example. The green set in Figure 2(a) shows that the two sets we obtained from $\text{Post}_D$ are contained in $\hat{X}(0)$ computed before. (Recall that in this example we only consider a single location; hence the inclusion holds for symbolic states.)

The steps outlined above describe one iteration of the standard reachability algorithm. Each symbolic state for which the fixpoint check was negative spawns a new flowpipe. Since this can lead to a combinatorial explosion, one typically applies a technique called clustering (cf. [20]), where symbolic states are merged after the application of $\text{Post}_D$. Here we consider clustering with a convex hull.
Example. Assume that the fixpoint check above was negative for both sets that we tested. In Figure 2(b) we show the convex hull of the sets in purple.

Up to now, we have seen a standard incorporation of an algorithm for the continuous-post operator $Post_C$ (for which we used $Post_C □$) into a reachability algorithm for hybrid systems. Observe that $Post_C$ was used as a black box. Consequently, we could not make use of the properties of the specific algorithm $Post_C □$. In particular, apart from $Post_C □$, we performed all computations in high dimensions. In the next section, we describe a new algorithm that instead performs all computations in low dimensions.

4 Decomposed reachability analysis

We now present a new, decomposed reachability algorithm for linear hybrid systems. The algorithm uses $Post_C □$ for computing flowpipes and has two major performance improvements over the previous algorithm.

Recall that $Post_C □$ computes flowpipes consisting of decomposed sets. The first improvement is to exploit the decomposed structure in order to perform all other operations (intersection, affine map, inclusion check, and convex hull) in low dimensions.

The second improvement is to compute flowpipes in a sparse way. Roughly speaking, we are only interested in those dimensions of a flowpipe that are relevant to determine intersection with a guard. Hence we only need to compute the other dimensions of the flowpipe after we detected such an intersection.

The algorithm starts as before: Given an initial (symbolic) state, we compute $X(0)$ (discretization) and decompose the set to obtain $\hat{X}(0)$. Next we want to compute a flowpipe, and this is where we deviate from the previous algorithm.

4.1 Computing a sparse flowpipe

We hook into $Post_C □$ in order to control the dimensions of the flowpipe. Recall that the black-box version of $Post_C □$ computed the flowpipe $\hat{X}(k) = X_1(k) \times \cdots \times X_b(k)$ for $k = 1, \ldots, N$, i.e., in all dimensions. This is usually not necessary for detecting an intersection with a guard. We will discuss this formally below, but want to establish some intuition first. Recall the running example from before. The guard was only constrained in dimension $x_1$. This means that the bounds of the sets $\hat{X}(k) = X_1(k) \times X_2(k)$ in dimension $x_2$ are irrelevant. Consequently, we do not need to compute the sets $X_2(k)$ at all (at least for the moment). We only compute those dimensions of a flowpipe that are necessary to determine intersection with the guards. Identifying these dimensions and projecting the guards accordingly has to be performed only once per transition and is often just a syntactic procedure.
Fig. 3. The flowpipe from Figure 1(a) in dimension $x_1$ only consists of intervals (blue). The constraint $G_1$ (red) is the guard $G$ projected to $x_1$. For better visibility, we draw the sets thicker and add a slight offset to some of the intervals.

**Example.** As discussed, we only compute the flowpipe $X_1(1), \ldots, X_1(4)$ for the first block (in dimension $x_1$), i.e., a sequence of intervals, which is depicted in Figure 3. Projecting the guard to $x_1$, we obtain a ray $G_1$. As expected, we observe an intersection with the guard for the same time steps as before (steps 3 and 4).

### 4.2 Decomposing an intersection

Next we discuss how to compute an intersection $\hat{X} \cap \mathcal{Y}$, respectively detect emptiness of this intersection, in low dimensions. The key idea is to exploit that $\hat{X}$ is decomposed. For ease of discussion, we consider the case of two blocks (i.e., $\hat{X} = X_1 \times X_2$). Below we discuss the two cases that $\mathcal{Y}$ is decomposed or not.

**Intersection between two decomposed sets.** We first consider the case that $\mathcal{Y}$ is also decomposed and agrees with $\hat{X}$ on the block structure, i.e., $\mathcal{Y} = \hat{Y} = Y_1 \times Y_2$ and $X_1, Y_1 \subseteq \mathbb{R}^{n_1}$ for some $n_1$. Clearly

$$\hat{X} \cap \hat{Y} = (X_1 \times X_2) \cap (Y_1 \times Y_2) = (X_1 \cap Y_1) \times (X_2 \cap Y_2)$$

because the Cartesian product and intersection distribute, and thus

$$\hat{X} \cap \hat{Y} = \emptyset \iff (X_1 \cap Y_1 = \emptyset) \lor (X_2 \cap Y_2 = \emptyset). \quad (5)$$

Now consider the second disjunct in (5) and assume that $\mathcal{Y}_2$ is universal. We get $X_2 \cap Y_2 = \emptyset \iff X_2 = \emptyset$. In our context, $\hat{X}$ (and hence $X_2$) is nonempty by construction. Hence (5) simplifies to

$$\hat{X} \cap \hat{Y} = \emptyset \iff X_1 \cap Y_1 = \emptyset,$$

so we never need to compute $X_2$ to determine whether the intersection is empty.

In practice, the set $G^*$ from [4] takes the role of $\hat{Y}$ and is often only constrained in some dimensions (and hence decomposed and universal in all other dimensions).

**Intersection between a decomposed and a non-decomposed set.** In general, if $\mathcal{Y}$ is not decomposed in the same block structure as $\hat{X}$, we can still decompose it, at the cost of an approximation error. Let $\pi_i$ be suitable projection matrices. Then

$$\hat{X} \cap \mathcal{Y} \subseteq (X_1 \cap \pi_1 \mathcal{Y}) \times (X_2 \cap \pi_2 \mathcal{Y})$$
and hence
\[ \hat{X} \cap \mathcal{Y} = \emptyset \Longleftrightarrow (X_1 \cap \pi_1 \mathcal{Y} = \emptyset) \lor (X_2 \cap \pi_2 \mathcal{Y} = \emptyset) \Longleftarrow X_1 \cap \pi_1 \mathcal{Y} = \emptyset. \] (6)

From (6) we obtain (1) a sufficient test for emptiness of \( \hat{X} \cap \mathcal{Y} \) in terms of only \( X_1 \) and (2) a more precise sufficient test in terms of \( X_1 \) and \( X_2 \) in low dimensions. If both tests fail, we can either fall back to the (exact) test in high dimensions or conservatively assume that the intersection is nonempty.

The precision of the above scheme highly depends on the structure of \( \hat{X} \) and \( \mathcal{Y} \). If several (but not all) blocks of \( \mathcal{Y} \) are constrained, instead of decomposing \( \mathcal{Y} \) into the low-dimensional block structure, one can alternatively compute the intersection for medium-dimensional sets to avoid an approximation error; we apply this strategy in the evaluation (Section 5). If \( \mathcal{Y} \) is compact, the following proposition shows that the approximation error is bounded by the maximal entry in the diameters of \( \hat{X} \) and \( \mathcal{Y} \), and this bound is tight.

**Proposition 2.** Let \( \hat{X} = \bigtimes_j X_j \in C_n, \ \mathcal{Y} \in C_n, \ \hat{X} \cap \mathcal{Y} \neq \emptyset, \ \hat{\mathcal{Y}} := \bigtimes_j \pi_j \mathcal{Y} \) for appropriate projection matrices \( \pi_j \) corresponding to \( X_j \), and \( p = \infty \). Then
\[
d_H^p(\hat{X} \cap \mathcal{Y}, \hat{X} \cap \hat{\mathcal{Y}}) \leq \max_j \min(\|\Delta_p(X_j)\|_p, \|\Delta_p(\pi_j \mathcal{Y})\|_p).
\]

**Example.** Consider again Figure 3. We have already identified the intersection with the flowpipe for time steps 3 and 4. The resulting sets for \( k = 3 \) and \( k = 4 \) are \( \hat{X}(k) \cap \mathcal{G} = X_1(k) \cap G_1 \times X_2(k) \), where \( G_1 \) was the projection of \( \mathcal{G} \) to \( x_1 \). We emphasize that we compute the intersections in low dimensions, that we need not compute \( X_2(1) \) and \( X_2(2) \) at all, and that in this example all computations are exact (i.e., we obtain the same sets as in Figure 2(a)).

While computing the intersection of two \( n \)-dimensional sets \( \mathcal{X} \) and \( \mathcal{G} \) in low dimensions is generally beneficial for performance, it is particularly interesting if one of the sets is a polytope that is not represented by its constraints. Common cases are the vertex representation or zonotopes represented by their generators, which are used in several approaches [23,25,6]. To compute the (exact) intersection of such a polytope \( \mathcal{X} \) with a polyhedron \( \mathcal{G} \) in constraint representation, \( \mathcal{X} \) needs to be converted to constraint representation first. A polytope with \( m \) vertices can have \( O\left(\frac{m-n+1}{n-1}\right)\) constraints [38]. (We note that, for two polytopes in vertex representation in general position, there is a polynomial-time intersection algorithm [22], but this assumption is not practical.) A zonotope with \( m \) generators can have \( O\left(m\left(\frac{m}{m-1}\right)\right)\) constraints [6]. If \( \mathcal{G} \) is a polytope in constraint representation, checking whether \( \mathcal{X} \) and \( \mathcal{G} \) are disjoint can also be solved more efficiently in low dimensions, e.g., for \( m \) constraints in \( O(m) \) for \( n \leq 3 \) [10].

### 4.3 Decomposing an affine map

The next step after computing the intersection with the guard is the application of the assignment. We consider an affine map \( A \hat{X} \oplus \{\bar{b}\} \) with \( A \in \mathbb{R}^{n \times n} \) and
\( \tilde{b} \in \mathbb{R}^n \). Affine-map decomposition has already been presented as part of the operator \( \text{Post}^\square_C \) [12]:

\[
A \tilde{X} \oplus \{ \tilde{b} \} \subseteq X \bigoplus_j A_{ij}X_j \oplus \{ \tilde{b}_i \}
\]

where \( A_{ij} \) is the block in the \( i \)-th block row and the \( j \)-th block column. We recall an error estimation.

**Proposition 3.** [12, Prop. 3] Let \( X = \bigtimes_{j=1}^b X_j \in \mathcal{C}_n \) be nonempty, \( A \in \mathbb{R}^{n \times n} \), \( q_j := \arg \max_i \| A_{ij} \|_p \) (the index of the block with the largest matrix norm in the \( j \)-th block column) so that \( \alpha_j := \max_{i \neq q_j} \| A_{ij} \|_p \) is the second largest matrix norm in the \( j \)-th block column. Let \( \alpha_{\text{max}} := \max_j \alpha_j \) and \( \Delta_{\text{sum}} := \sum_j \Delta_{\infty}(X_j) \).

Then

\[
d_H^p(AX \bigoplus_j A_{ij}X_j) = \max_{\|d\|_p \leq 1} \sum_{i,j} \rho_{X_i}(A_{ij}^T d_i) - \rho_{X_j} \left( \sum_k A_{kj}^T d_k \right)
\]

\[
\leq (b - 1) \sum_j \alpha_j \Delta_{\infty}(X_j) \leq \frac{n}{2} \alpha_{\text{max}} \Delta_{\text{sum}}.
\]

In particular, if only one block per block column of matrix \( A \) is nonzero, the approximation is exact [12]. For example, consider a two-block scenario and a block-diagonal matrix \( A \), i.e., \( A_{12} = A_{21} = 0 \). Then

\[
\begin{pmatrix}
A_{11} & 0 \\
0 & A_{22}
\end{pmatrix}
X_1 \times X_2 \oplus \{ \tilde{b}_1 \} \times \{ \tilde{b}_2 \} = (A_{11}X_1 \oplus \{ \tilde{b}_1 \}) \times (A_{22}X_2 \oplus \{ \tilde{b}_2 \}).
\]

In practice, affine maps with such a structure are unrealistic for the \( \text{Post}^\square_C \) operator but typical for assignments. Prominent cases include resets, translations, and scalings, for which \( A \) is even diagonal and hence block diagonal for any block structure.

**Example.** Recall that, after computing the intersections, we ended up with the same sets as in Figure 2(a). In our example, the assignment is a translation in dimension \( x_1 \). Hence, as mentioned above, the application of the decomposed assignment is also exact. In particular, the translation only affects \( X_1(k) \). Thus we again obtain the same result as in Figure 2(a) before.

### 4.4 Inclusion check for decomposed sets

Our algorithm is now fully able to take transitions. Observe that all sets ever occurring in scheme (3) using the algorithm are decomposed. The following proposition gives an exact low-dimensional fixpoint check under this condition.

**Proposition 4.** Let \( \tilde{X} = \bigtimes_j X_j \in \mathcal{C}_n, \tilde{Y} = \bigtimes_j Y_j \in \mathcal{C}_n \) be nonempty sets with identical block structure. Then \( \tilde{X} \subseteq \tilde{Y} \iff \bigwedge_j X_j \subseteq Y_j \).
4.5 Decomposing a convex hull

As the last part of the algorithm, we decompose the computation of the convex hull. We exploit that all sets in the same flowpipe share the same block structure.

**Proposition 5.** Let \( \hat{X} = \bigtimes_j X_j \in C_n, \hat{Y} = \bigtimes_j Y_j \in C_n \) be nonempty sets with identical block structure. Then \( \text{CH}(\hat{X} \cup \hat{Y}) \subseteq \bigtimes_j \text{CH}(X_j \cup Y_j) \).

For the decomposition operations proposed before (intersection, affine map, and inclusion), there are common cases where the approximations were exact. In these cases it is always beneficial to perform the decomposed operations instead of the high-dimensional counterparts. The decomposition of the convex hull, however, always incurs an approximation error, which we can bound by the radius of the box approximation and by the block-wise difference in bounds.

**Proposition 6.** Let \( \hat{X} = \bigtimes_j X_j \in C_n, \hat{Y} = \bigtimes_j Y_j \in C_n \) be nonempty sets with identical block structure and let \( r^{\infty} \) be the radius of the box approximation of \( \text{CH}(\hat{X} \cup \hat{Y}) \). Then

\[
d_H^{\infty}(\text{CH}(\hat{X} \cup \hat{Y}), \bigtimes_j \text{CH}(X_j \cup Y_j)) \leq \min \left( \|r^{\infty}\|_\infty, \max_{\|d\|_p \leq 1} \sum_j |\rho_{X_j}(d_j) - \rho_{Y_j}(d_j)| \right).
\]

**Example.** Figure 2(b) shows the decomposed convex hull of the sets \( \hat{X}(3) \cap \mathcal{G} \) and \( \hat{X}(4) \cap \mathcal{G} \) after applying the translation. Since each block is one-dimensional in our example, we obtain the box approximation.

5 Evaluation

We implemented the ideas presented in Section 4 in JuliaReach [2,13]. The code is publicly available [2]. We performed the experiments presented in this section on a Mac notebook with an Intel i5 CPU@3.1 GHz and 16 GB RAM.

5.1 Benchmarks descriptions

We run our algorithm on a number of benchmarks taken from the HyPro model library [1] and also from ARCH-COMP 2018 [3]. We briefly describe them below.

**Linear Switching System.** This model is a piecewise linear system with different controlled continuous dynamics, generated randomly and stabilized using an LQR controller. The transitions are determined heuristically from simulations.

**Spacecraft Rendezvous.** This model represents a spacecraft steering toward a passive target in space [15]. We use a linearized version of this model with three locations. In one setting, a mission abort occurs at \( t = 120 \) min.

**Platooning model.** This model represents a three-vehicle platoon with loss of communication at deterministic times [37]. We consider both a time-bounded and a time-unbounded setting (the latter setting requires to find a fixpoint).

**Filtered oscillator.** This model consists of a two-dimensional switched oscillator (dimensions \( x_1 \) and \( x_2 \)) and a parametric number of filters (here: 256–1024) which smooth \( x_1 \) [20]. We fixed the maximum number of transitions to five.
Table 1. For each benchmark we report the dimension $n$ (Dim.), the number of transitions taken ($Jump$), the (uniform) block size ($Block$), the step size of the time discretization ($Step$), and the runtime of the high-dimensional and of the low-dimensional algorithm in seconds. “TO” indicates a timeout of $5 \times 10^3$ seconds.

| Benchmark       | Dim. | Jump | Block | Step  | High-dim. | Low-dim. |
|-----------------|------|------|-------|-------|-----------|----------|
| spacecraft_noabort | 5    | 1    | 1     | 0.04  | $1.4 \times 10^4$ | $7.6 \times 10^0$ |
| spacecraft_120  | 5    | 2    | 1     | 0.04  | $4.3 \times 10^0$  | $5.0 \times 10^0$ |
| linear_switching  | 5    | 4    | 1     | 0.001 | $1.2 \times 10^0$  | $9.0 \times 10^{-1}$ |
| platoon_t20     | 10   | 4    | 1     | 0.001 | $5.3 \times 10^3$  | $1.1 \times 10^1$ |
| platoon_tInf    | 10   | 52   | 1     | 0.001 | $8.5 \times 10^2$  | $1.4 \times 10^2$ |
| filtered_osc256 | 256  | 5    | 1     | 0.01  | $1.6 \times 10^2$  | $3.1 \times 10^1$ |
| filtered_osc256 | 256  | 5    | 2     | 0.001 | $1.2 \times 10^2$  | $1.8 \times 10^1$ |
| filtered_osc256 | 256  | 5    | 2     | 0.0005 | $1.9 \times 10^3$ | $2.0 \times 10^2$ |
| filtered_osc512 | 512  | 5    | 2     | 0.01  | $5.5 \times 10^2$  | $7.6 \times 10^1$ |
| filtered_osc512 | 512  | 5    | 2     | 0.0005 | TO    | $8.3 \times 10^2$ |
| filtered_osc1024 | 1024 | 5   | 2     | 0.01  | TO    | $5.0 \times 10^2$ |
| filtered_osc1024 | 1024 | 5   | 2     | 0.0005 | TO    | $3.4 \times 10^3$ |
two-dimensional block structures. For one-dimensional blocks, we performed the intersection in two dimensions and then projected back to one dimension. In Figure 4, we compare the flowpipes for these two cases. The one-dimensional analysis is moderately less precise. We note that a one-dimensional block structure also reduces the precision of Post\textsuperscript{C} in general, so the additional approximation error does not only stem from our algorithm. Interestingly, the two-dimensional analysis was even slightly faster. Apparently, the benefit of interval operations over two-dimensional operations is outweighed by the increased number of sets here.

6 Conclusion

We have presented an algorithm that integrates a decomposition-based reachability algorithm for LTI systems in the analysis loop for linear hybrid systems. The key insight is that intersections with polyhedral constraints can be efficiently detected and computed (approximately or often even exactly) in low dimensions. This enables the systematic focus on appropriate subspaces and the potential for bypassing large amounts of flowpipe computations.

Another application of our approach, which we have not yet explored, is the fast computation of a low-dimensional flowpipe only for the detection of intersections. Then one can recompute the flowpipe for the relevant time frame(s) with higher precision (either using different algorithmic parameters or even a different algorithm, e.g., one that features arbitrary precision [24]). This is particularly efficient for LTI systems because one can avoid recomputing the homogeneous (state-based) part of the flowpipe [21].

We have not discussed the possibility to change the block structure when switching locations. Different locations may constrain different dimensions, so tracking the “right” dimensions may be necessary to maintain precision. While it is easy to merge different blocks, subsequent computations will become more
expensive. Hence one may also want to split blocks for optimal performance, but this comes with a loss in precision. Devising heuristics for the splitting of blocks, possibly in a refinement loop, is an interesting direction for future work.

References

1. HyPro Benchmark Repository. https://ths.rwth-aachen.de/research/projects/hypro/benchmarks-of-continuous-and-hybrid-systems/ (2019)
2. JuliaReach. https://github.com/JuliaReach (2019)
3. Althoff, M., Bak, S., Chen, X., Fan, C., Fores, M., Frehse, G., Kochdumper, N., Li, Y., Mitra, S., Ray, R., Schilling, C., Schupp, S.: ARCH-COMP18 category report: Continuous and hybrid systems with linear continuous dynamics. In: ARCH. pp. 23–52 (2018). http://www.easychair.org/publications/paper/4cGr
4. Althoff, M., Frehse, G.: Combining zonotopes and support functions for efficient reachability analysis of linear systems. In: CDC. pp. 7439–7446 (2016). https://doi.org/10.1109/CDC.2016.7799418
5. Althoff, M., Krogh, B.H.: Avoiding geometric intersection operations in reachability analysis of hybrid systems. In: HSCC. pp. 45–54 (2012). https://doi.org/10.1145/2185632.2185643
6. Althoff, M., Stursberg, O., Buss, M.: Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. Nonlinear Analysis: Hybrid Systems (2), 233–249 (2010). https://doi.org/10.1016/j.nahs.2009.03.009
7. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid Systems. pp. 209–229 (1992). https://doi.org/10.1007/3-540-540-57318-6_30
8. Asarin, E., Dang, T.: Abstraction by projection and application to multi-affine systems. In: HSCC (2004). https://doi.org/10.1007/978-3-540-24743-2_3
9. Bak, S., Bogomolov, S., Althoff, M.: Time-triggered conversion of guards for reachability analysis of hybrid automata. In: FORMATS. pp. 133–150 (2017). https://doi.org/10.1007/978-3-319-65765-3_8
10. Barba, L., Langerman, S.: Optimal detection of intersections between convex polyhedra. In: SODA (2015). https://doi.org/10.1137/1.9781611973730.109
11. Bogomolov, S., Donzé, A., Frehse, G., Grosu, R., Johnson, T.T., Ladan, H., Podelski, A., Wehrle, M.: Abstraction-based guided search for hybrid systems. In: SPIN. pp. 117–134 (2013). https://doi.org/10.1007/978-3-642-39176-7_8
12. Bogomolov, S., Fores, M., Frehse, G., Podelski, A., Schilling, C., Viry, F.: Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices. In: HSCC. pp. 41–50 (2018). https://doi.org/10.1145/3178126.3178128
13. Bogomolov, S., Fores, M., Frehse, G., Potomkin, K., Schilling, C.: JuliaReach: a toolbox for set-based reachability. In: HSCC (2019). https://doi.org/10.1145/3302504.3311804
14. Bu, L., Li, X.: Path-oriented bounded reachability analysis of composed linear hybrid systems. STTT (4), 307–317 (2011). https://doi.org/10.1007/s10009-010-0163-9
15. Chan, N., Mitra, S.: Verifying safety of an autonomous spacecraft rendezvous mission. In: ARCH (2017). http://www.easychair.org/publications/paper/342723
16. Chen, M., Herbert, S.L., Tomlin, C.J.: Exact and efficient Hamilton-Jacobi guaranteed safety analysis via system decomposition. In: ICRA. pp. 87–92 (2017). https://doi.org/10.1109/ICRA.2017.7989015
17. Chen, X., Sankaranarayanan, S.: Decomposed reachability analysis for nonlinear systems. In: RTSS. pp. 13–24 (2016), https://doi.org/10.1109/RTSS.2016.011

18. Chutinan, A., Krogh, B.H.: Computational techniques for hybrid system verification. IEEE Trans. Automat. Contr. (1), 64–75 (2003), https://doi.org/10.1109/TAC.2002.806655

19. Dontchev, A.L.: Time-scale decomposition of the reachable set of constrained linear systems. MCSS (3), 327–340 (1992), https://doi.org/10.1007/BF01211565

20. Frehse, G., Guernic, C.L., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable verification of hybrid systems. In: CAV. pp. 379–395 (2011), https://doi.org/10.1007/978-3-642-22110-1_30

21. Frehse, G., Ray, R.: Flowpipe-guard intersection for reachability computations with support functions. In: ADHS. pp. 94–101. No. 9 (2012), https://doi.org/10.3182/20120606-3-NL-3011.00053

22. Fukuda, K., Liebling, T.M., Lütfol, C.: Extended convex hull. Comput. Geom. (1-2), 13–23 (2001), https://doi.org/10.1016/S0925-7721(01)00032-3

23. Girard, A.: Reachability of uncertain linear systems using zonotopes. In: HSCC. pp. 291–305 (2005), https://doi.org/10.1007/107-3-540-46983-6_19

24. Girard, A., Guernic, C.L.: Efficient reachability analysis for linear systems using support functions. IFAC Proceedings Volumes (2), 8966–8971 (2008), https://doi.org/10.3182/20080706-5-KR-1001.01514

25. Girard, A., Guernic, C.L.: Zonotope/hyperplane intersection for hybrid systems reachability analysis. In: HSCC. pp. 215–228 (2008), https://doi.org/10.1007/978-3-540-78929-1_16

26. Goncharova, E.V., Ovseevich, A.I.: Asymptotics for singularly perturbed reachable sets. In: LSSC. pp. 280–285 (2009), https://doi.org/10.3182/20090706-3-NL-3011.00053

27. Greenstreet, M.R., Mitchell, I.: Reachability analysis using polygonal projections. In: HSCC. pp. 103–116 (1999), https://doi.org/10.1007/107-3-540-48983-5_12

28. Guernic, C.L., Girard, A.: Reachability analysis of hybrid systems using support functions. In: CAV. pp. 540–554 (2009), https://doi.org/10.1007/978-3-642-02658-8_40

29. Gurung, A., Ray, R., Bartocci, E., Bogomolov, S., Grosu, R.: Parallel reachability analysis of hybrid systems in XSpeed. STTT pp. 1–23 (2018), https://doi.org/10.1007/s10009-018-0485-6

30. Hamadeh, A.O., Goncalves, J.M.: Reachability analysis of continuous-time piecewise affine systems. Automatica (12), 3189–3194 (2008), https://doi.org/10.1016/j.automatica.2008.05.023

31. Han, Z., Krogh, B.H.: Reachability analysis of large-scale affine systems using low-dimensional polytopes. In: HSCC. pp. 287–301 (2006), https://doi.org/10.1007/11730637_23

32. Heuvinger, T.A.: The theory of hybrid automata. In: LICS. pp. 278–292 (1996), https://doi.org/10.1109/LICS.1996.561342

33. Hochbaum, D.S., Naor, J.: Simple and fast algorithms for linear and integer programs with two variables per inequality. SIAM J. Comput. (6), 1179–1192 (1994), https://doi.org/10.1137/S0097539793251876

34. Kaynama, S., Oishi, M.: Overapproximating the reachable sets of LTI systems through a similarity transformation. In: ACC. pp. 1874–1879 (2010), https://doi.org/10.1109/ACC.2010.5531392

35. Kaynama, S., Oishi, M.: Complexity reduction through a Schur-based decomposition for reachability analysis of linear time-invariant systems. Int. J. Control (1), 165–179 (2011), https://doi.org/10.1080/00207179.2010.543703
36. Le Guernic, C.: Reachability analysis of hybrid systems with linear continuous dynamics. Ph.D. thesis, Université Grenoble 1 - Joseph Fourier (2009), https://tel.archives-ouvertes.fr/tel-00422569
37. Makhlouf, I.B., Kowalewski, S.: Networked cooperative platoon of vehicles for testing methods and verification tools. In: ARCH. pp. 37–42 (2014), http://www.easychair.org/publications/paper/252253
38. McMullen, P.: The maximal number of faces of a convex polytope. Mathematika pp. 179–184 (1970)
39. Mitchell, I.M., Tomlin, C.: Overapproximating reachable sets by Hamilton-Jacobi projections. J. Sci. Comput. (1-3), 323–346 (2003), https://doi.org/10.1023/A:1025364227553
40. Nedialkov, N.S., Mohrenschildt, M.V.: Rigorous simulation of hybrid dynamic systems with symbolic and interval methods. In: ACC. pp. 140–147 (2002), https://doi.org/10.1109/ACC.2002.1024794
41. Schupp, S., Nellen, J., Abrahám, E.: Divide and conquer: Variable set separation in hybrid systems reachability analysis. In: QAPL@ETAPS. pp. 1–14 (2017), https://doi.org/10.4204/EPTCS.250.1
42. Seladji, Y., Bouissou, O.: Numerical abstract domain using support functions. In: NASA Formal Methods. pp. 155–169 (2013), https://doi.org/10.1007/978-3-642-38088-4_11
43. Yan, C., Greenstreet, M.R.: Faster projection based methods for circuit level verification. In: ASP-DAC. pp. 410–415 (2008), https://doi.org/10.1109/ASPDAC.2008.4483985
A Proofs

Proof (Proposition 1).

\[ d_{\|\cdot\|_p}^H(\mathcal{X}, \bigtimes \pi_j \mathcal{X}) = \max_{\|d\|_p \leq 1} \rho_{\pi_0 \mathcal{X}} - \rho_{\mathcal{X}}(d) = \max_{\|d\|_p \leq 1} \sum_j \rho_{\pi_j \mathcal{X}}(\pi_j d) - \rho_{\mathcal{X}}(d) \]

\[ = \max_{\|d\|_p \leq 1} \sum_j \max_{x \in \mathcal{X}}(\pi_j d, \pi_j x) - \max_{y \in \mathcal{X}}(d, y) = \max_{\|d\|_p \leq 1} \sum_j \min_{x \in \mathcal{X}}(\pi_j d, \pi_j x) - (d, y) \]

\[ = \max_{\|d\|_p \leq 1} \min_{y \in \mathcal{X}} \sum_j \max_{x \in \mathcal{Y}}(\pi_j d, \pi_j x) - (\pi_j d, \pi_j y) = \max_{\|d\|_p \leq 1} \min_{y \in \mathcal{X}} \sum_j \max_{x \in \mathcal{Y}}(\pi_j d, \pi_j x - \pi_j y) \]

For \( y \) we can choose the center of the box approximation of \( \mathcal{X} \), \( c^p_X \in \mathcal{X} \).

\[ \leq \max_{\|d\|_p \leq 1} \sum_j \max_{x \in \mathcal{X}}(\pi_j d, \pi_j x - \pi_j c^p_X) \leq \max_{x \in \mathcal{X}} \pi_j x - \pi_j c^p_X \leq \|r^p_X\|_p \]

\[ \square \]

Proof (Proposition 2).

\[ d_{\|\cdot\|_p}^H(\tilde{\mathcal{X}}, \tilde{\mathcal{Y}} \cap \tilde{\mathcal{Y}}) = \max_{\|d\|_p \leq 1} \rho_{\tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}}} - \rho_{\tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}}}(d) \]

\[ = \max_{\|d\|_p \leq 1} \max_{x, y \in \tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}}} (d, x) - \max_{y \in \tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}}} (d, y) = \max_{\|d\|_p \leq 1} \max_{x, y \in \tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}}} (d, x - y) \]

We can take absolute values in the dot product by choosing \( d \) accordingly.

\[ = \max_{\|d\|_p \leq 1} \max_{x, y \in \tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}}} (d, |x - y|) \leq \max_{\|d\|_p \leq 1} \max_{x, y \in \tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}}} (d, |x - y|) \]

We can conservatively bound the distance \( |x - y| \) by the diameter of \( \tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}} \).

\[ \leq \max_{\|d\|_p \leq 1} \max_{x, y \in \tilde{\mathcal{X}} \cap \tilde{\mathcal{Y}}} \max_{\|\Delta_p(\tilde{\mathcal{X}})\|_p, \|\Delta_p(\pi_j \tilde{\mathcal{Y})\|_p} \]

\[ \square \]

Figure 5 shows that the bound of Proposition 2 is tight.

**Fig. 5.** The rectangle \( \tilde{\mathcal{X}} \) (blue) and the line segment \( \mathcal{Y} \) (green) intersect in a single point (cyan). The Cartesian decomposition \( \tilde{\mathcal{Y}} \) of \( \mathcal{Y} \) (red) intersects with \( \tilde{\mathcal{X}} \) on a whole facet (yellow). Observe that the length of this intersection facet is determined by the width of both \( \tilde{\mathcal{X}} \) and \( \mathcal{Y} \) in dimension \( x_1 \), independent of the height in dimension \( x_2 \).
Proof (Proposition 4).

\[ \widehat{X} \subseteq \widehat{Y} \iff \forall x \in \widehat{X} : x \in \widehat{Y} \iff \forall x_1, \ldots, x_b \in \mathcal{X}_b : \bigwedge_{j=1}^{b} x_j \in \mathcal{Y}_j \]
\[ \iff \bigwedge_{j=1}^{b} \forall x_j \in \mathcal{X}_j : x_j \in \mathcal{Y}_j \iff \bigwedge_{j=1}^{b} \mathcal{X}_j \subseteq \mathcal{Y}_j \quad \square \]

Proof (Proposition 5). Let \( d \in \mathbb{R}^n \). We show \( \rho_{\mathrm{CH}(\widehat{X} \cup \widehat{Y})}(d) \leq \rho_{\chi_j, \mathrm{CH}(\mathcal{X}_j \cup \mathcal{Y}_j)}(d) \).

\[ \rho_{\mathrm{CH}(\widehat{X} \cup \widehat{Y})}(d) = \max(\rho_{\widehat{X}}(d), \rho_{\widehat{Y}}(d)) = \max(\sum_j \rho_{\chi_j}(d_j), \sum_j \rho_{\chi_j}(d_j)) \]
\[ \leq \sum_j \max(\rho_{\chi_j}(d_j), \rho_{\chi_j}(d_j)) = \sum_j \rho_{\mathrm{CH}(\mathcal{X}_j \cup \mathcal{Y}_j)}(d_j) = \rho_{\chi_j, \mathrm{CH}(\mathcal{X}_j \cup \mathcal{Y}_j)}(d) \quad \square \]

Proof (Proposition 6). The first bound is due to Proposition 4.

\[ d_H^p(\mathrm{CH}(\widehat{X} \cup \widehat{Y}), \bigotimes_j \mathrm{CH}(\mathcal{X}_j \cup \mathcal{Y}_j)) = \max_{\|d\|_p \leq 1} \rho_{\chi_j, \mathrm{CH}(\mathcal{X}_j \cup \mathcal{Y}_j)}(d) - \rho_{\mathrm{CH}(\widehat{X} \cup \widehat{Y})}(d) \]
\[ = \max_{\|d\|_p \leq 1} \left( \sum_j \rho_{\mathrm{CH}(\mathcal{X}_j \cup \mathcal{Y}_j)}(d_j) \right) - \max(\rho_{\widehat{X}}(d), \rho_{\widehat{Y}}(d)) \]
\[ = \max_{\|d\|_p \leq 1} \left( \sum_j \max(\rho_{\chi_j}(d_j), \rho_{\chi_j}(d_j)) \right) - \max \left( \sum_j \rho_{\chi_j}(d_j), \sum_j \rho_{\chi_j}(d_j) \right) \]
\[ = \max_{\|d\|_p \leq 1} \min \left( \sum_j \varphi(d_j) - \rho_{\chi_j}(d_j), \sum_j \varphi(d_j) - \rho_{\chi_j}(d_j) \right) \]
\[ = \max_{\|d\|_p \leq 1} \min \left( \sum_j \max(0, \rho_{\chi_j}(d_j) - \rho_{\chi_j}(d_j)), \sum_j \max(0, \rho_{\chi_j}(d_j) - \rho_{\chi_j}(d_j)) \right) \]
\[ \leq \max_{\|d\|_p \leq 1} \sum_j |\rho_{\chi_j}(d_j) - \rho_{\chi_j}(d_j)| \quad \square \]