Reachability Analysis via Block Decomposition

Abhinav Chawla
Department of Computer Science
Stony Brook University
Stony Brook, NY, USA

***Abstract*—This paper works on experimental results and analysis regarding Reachability Analysis using decomposition and over approximation of sets. Using the idea of block decomposition, we are able to transform the majority of set operations(e.g. Minkowski Sum) into lower dimensions multiplications while matrix operations like exponentation is taken in high dimension itself. This paper implements the novel idea in python using zonotopes to represent states while trying to find new optimizations.**

I. Introduction

Set based Reachability computation is an essential requirement to understand about the behaviour of dynamical system [2]. We are interested in the set-based reachability problem for affine dynamical systems. Linear time invariant systems can be written in the form of

$$\dot{x}(t) = Ax(t) + Cu(t) \tag{1}$$

where $u(t)$ is non deterministic input or noise. Solving this, we can get a recurrence relation of the form

$$X(k+1) = \phi X(k) \oplus U(k), \text{ for } k = 1, \cdots, N \tag{2}$$

In this paper, we will ignore the non deterministic inputs or U(k) for now. $\oplus$ denotes the Minkowski sum between n-dimensional sets , ϕ is $n \times n$ matrix which is calculated from the exponentiation of e^{At} where t is the step size.

The success of any reachability analysis algorithm depends on the set representation of the set of states. Several approaches have been provided for solving the above equation by representing the reachability set in the form of ellipsoids [14], template polyhedra such as zonotopes [10] or support functions [12], or a combination [1] or linear star sets [9]. The same problem is even extended for hybrid systems with linear continuous dynamics. The issue with most of the approaches is scalability as with more dimensions, the cost of some set operations for the set representations may increase super linearly.

The paper by Bogmolov et Al [5] focused on the issue of scalability by providing a novel decomposition algorithm to solve equation (1) by decomposing the initial set into smaller sets and perform set operations on these smaller sets while keeping the matrix operations at higher dimensions, hence keeping an approximation to a certain level. This paper will focus on r-implementing this approach on Python language using various set representations and figure out what other optimizations can be done over this approach.

II. Related Work

Kaynama and Oishi [13] used Schur-Based Decomposition to compute the reachability sets for Linear time invariant systems. The authors proposed a Schur-based decomposition technique for computing reachable sets and synthesizing safety-preserving controllers. Subsystems are analyzed separately, and reachable sets of subsystems are back-projected and intersected to construct an overapproximation of the reachable set, so that safety can still be guaranteed. Our paper uses the cartesian decomposition technique while getting similar optimizations in the overapproximation.

Seladji and Bouissou define a sub-polyhedra abstract domain based on support functions [16]. Our work doesn't depend on a typical set representation.

If the system is singularly perturbed with different time scales ("slow and fast variables"), time-scale decomposition can be applied [8], [11]. This setting is not considered in this paper.

Asarin and Dang [3] worked on creating decomposition approach to a non linear ODE, however this paper considers a unique decomposition approach to a linear affine system.

Bak and Duggirala [4] consider a "simulation-equivalent" reachability analysis to compute reachable sets by using the simulation approach for each distinct point in the reachable set, by decomposing the linear program and optimizing the minkowski sum. This set, however, does not include all trajectories of equation (1).

Chen and Sankaranarayanan apply uniform hybridization to analyze the subsystems over time and feed the results to the other subsystems as time-varying interval-shaped inputs [7]. Even though this is for a more general non-linear ODE, due to decoupling in our reachability algorithm, it need not be performed iteratively. Additionally, Chen and Sankaranarayanan use the concept of Taylor Models set representation, while our algorithm is independent of set representation.

Chen et al. show that, using Hamilton-Jacobi methods, the (analytically) exact reachable states can be reconstructed from an analysis of the subsystems for general ODE systems [6]. It is however, based on an assumption of system being composed of *self-contained subspaces* which is not true always. The technique is based on [15] which has no such limitation except a projection error. For general LTI systems, an approximation error is unavoidable.

III. Approximate Reachability of Affine Systems

In this section, we will introduce the concepts we have used in the implementation.

A. Minkowski Sum

Minkowski sum of two sets of position vectors A and B in Euclidean space is formed by adding each vector in A to each vector in B, i.e., the set -

$$A \oplus B = \{a + b | a \in A, b \in B\} \tag{3}$$

Intuitively it means to add each vector from A with each vector from B and this new set is the minkowski sum of A and B. A simple example of the minkowski sum of a rectangle with a circle in the same plane is shown below.

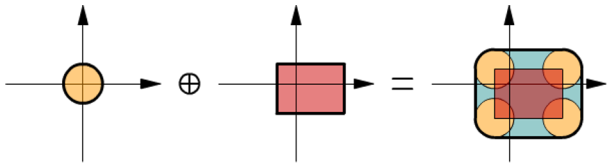

Fig. 1. Minkowski sum of rrectangle and circle

B. Overapproximation of Convex Set

We recall a technique to overapproximate a convex set by creating a polyhedron with help of *support functions* and *support vectors*. The support function $\rho_X : \mathbb{R}^n \to \mathbb{R}$ of a non-empty closed convex set X in $\mathbb{R}^n$ is given by -

$$\rho_X(l) = \sup\{l \cdot x : x \in X\} \text{ where } l \in \mathbb{R}^n \tag{4}$$

The furthest points in A in x direction are the support vectors given by:

$$\sigma_A(x) = \{a \in A : x \cdot a = \rho_A(x)\} \text{ where} \tag{5}$$

Furthermore, the support function, as a function of the set A, is compatible with many natural geometric operations, like scaling, translation, rotation and Minkowski Sum.

The projection of a set into a low dimensional space (a special case of MX) can be conveniently evaluated using support functions, since $\sigma_{MX}(x) = \sigma_X(M^T x)$. Given, $l_1, l_2, \cdots l_m$ directions we can find a tight overapproximation of X which will be the outer polyhedron given by the constraints

$$\bigwedge_i l_i \cdot x \leq \rho_X(l_i) \tag{6}$$

A simple outer polyhedron could be the bounding hyper-rectangle with $2n$ directions where n is the total number of dimensions. Adding more and more directions, will give better approximation but could increase the complexity. We will be using the bounding box approximation for this implementation, however the choice of approximation doesn't matter unless they are not compatible with operations like Minkowski Sum.

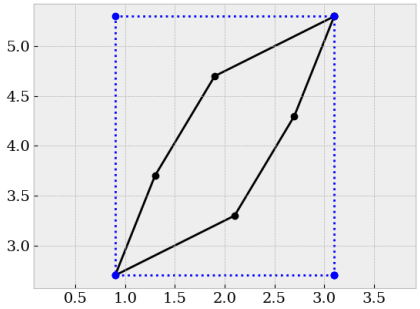

Fig. 2. A simple bounding box for a 2D polygon

C. Zonotopes

First let's define polytopes. Mathematically, we can define a polytope as -

$$P = P(A, b) := \{x \in \mathbb{R}_d | {a_i}^T x \leq b_i \text{ for } 1 \leq i \leq m\} \tag{7}$$

with the extra condition that the set of solutions is bounded, that is, such that there is a constant N such that $\| x \| \leq N$ holds for all $x \in P$. The m conditions are the m halfspaces defining a polytope.

Zonotopes are a special classes of polytopes. Instead of using halfspaces to define a set of polytope, zonotopes utilizes a set of *generators* or line segments. A zonotpe can written as minkowski sum of finite generators. In this paper, we will use the following definition -

$$Z = \left\{ x \in \mathbb{R}_n : x = c + \sum_{i=1}^{p} \alpha_i g_i, -1 \leq \alpha_i \leq 1 \right\} \tag{8}$$

where $c, g_1, \cdots, g_p$ are vectors of $\mathbb{R}_n$. We note $Z = (c, < g_1, \cdots, g_p >)$. c is the center of the zonotope and $g_i \cdots g_p$ are the p generators of zonotope.

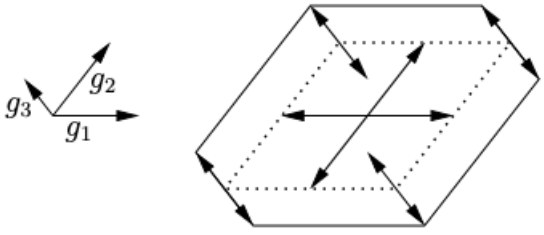

Fig. 3. A simple zonotope with 3 generators

Zonotopes are closed under linear transformation. Let L be a linear map and Z=$(c, < g1, ..., gp >)$ a zonotope, then

$$\begin{aligned} LZ &= \{Lx : x = c + \sum_{i=1}^{p} x_i g_i, -1 \leq x_i \leq 1\} \\ &= (Lc, < Lg_1, \cdots, Lg_p >) \end{aligned}$$

.The image of a zonotope by a linear map can be computed in linear timewith regard to the order of the zonotope.

Zonotopes are closed under Minkowski sum. Let $Z_1 = (c1, < g1, \cdots, g_p >)$ and $Z_2 = (c2, < h_1, \cdots, h_q >)$ be two zonotopes, then

$$Z_1 \oplus Z_2 = (c_1 + c_2, < g_1, \cdots g_p, h_1, \cdots, h_p >)$$

This means for the minkowski sum, one needs to add the center vectors and concatenate the generators of the two Zonotopes.

D. Cartesian Product

The cartesian product of two sets A and B can be written as:

$$A \times B = \{(a, b), \quad a \in A \text{ and } b \in B\} \tag{9}$$

Intuitively, it means to take all possible combinations of the elements from A and B, and a create a new set which would be of $m + n$ dimensions where A had m dimensions and B had n dimensions.

Unlike other operations we use generally, cartesian product is not associative nor commutative, so it needs to be used carefully.

Cartesian products of zonotopes $Z_1 = (c_1, < g1, \cdots, g_p >)$ and $Z_2 = (c_2, < h_1, \cdots, h_q >)$ can be written as -

$$Z_1 \times Z_2 = ((c_1, c_2), bl\_diag(< g1, \cdots, g_p >, < g1, \cdots, g_p >) \tag{10}$$

Here the bl_diag function is a block diagonal matrix which converts two matrices to a block diagonal matrix by padding zeros on the non-diagonal sides. For example,

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} B = \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

$$bl\_diag(A, B) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 4 & 5 \\ 0 & 0 & 6 & 7 & 8 \end{bmatrix}$$

E. Reachability Analysis

A trajectory of the affine ODE with time-varying inputs is the unique solution $x_{x0,u}(t) : [0, T] \to \mathbb{R}^n$, for a given initial condition x_0 at time $t = 0$, and a given input signal u,

$$x_{x_0,u}(t) = e^{At} x_0 + \int_0^t e^{A(t-s)} u(s) ds \tag{11}$$

where we map $Bu(t)$ to $u(t)$ without loss of generality. Here T is the time horizon, which is considered to be finite in this paper. Given a set of initial states X_0 and an input signal u, the reach set at time t is $R(X_0, u, t) := x_{x0,u}(t) : x_0 \in X_0$. This extends to a family of solutions as -

$$R(X_0, Y, t) := \bigcup R(X_0, u, t) : u(s) \in U(s), \forall s \in [0, t] \tag{12}$$

For the implementation in this paper, we would not be considering time-varying inputs for simplicity. However, the same logic can be extended for time-varying inputs.

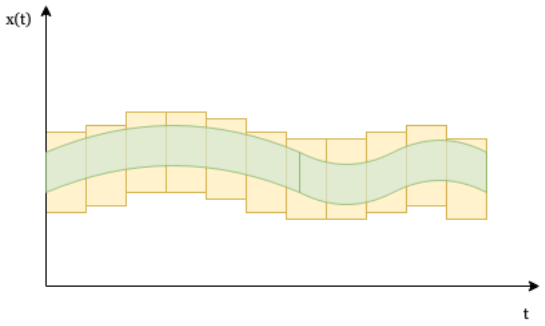

Fig. 4. I
llustration of a reach tube with a set of initial state with rectangles shown as the overapproximation.

F. Cartesian Decomposition

This section will present a novel way to do block decomposition using cartesian product. Let us consider a set $X \in \mathbb{R}^n$ be a convex set. We define the decomposition of X into $b = n/p$ sets of dimension p as follows. Let π_i be the projection matric that maps a vector $x \in \mathbb{R}^n$ to its coordinates in the $i-th$ block, $x_i = \pi_i x$. The *Cartesian Decomposition* can be written as -

$$dcp(X) = \pi_1 X \times \cdots \times \pi_b X \tag{13}$$

We call a set *decomposed* if it is identical to its Cartesian Decomposition. However, these projections are computationally expensive to compute if X is a polyhedron in constraint form or zonotope form. We could easily compute an over approximation as mentioned in Section III.B. We will write the b components as -

$$\hat{X}_1 \times \cdots \times \hat{X}_b = \hat{dcp}(X) \tag{14}$$

This operator overapproximates the Cartesian decomposition with a decomposed set $\hat{X}_1 \times \cdots \hat{X}_b$ such that $dcp(X) \subseteq \hat{dcp}(X)$

IV. Algorithm

A. Decomposing an Affine Map

First, we will see how will the reachability analysis will be calculated for the decomposed set. Let's say ϕ is the $n \times n$ affine transformation matrix -

$$X' = \phi X = \begin{bmatrix} \phi_{11} & \cdots & \phi_{1b} \\ \vdots & \ddots & \\ \phi_{b1} & & \phi_{bb} \end{bmatrix} X \tag{15}$$

ϕ_{ij} denotes the $p \times p$ submatrix of ϕ in row i and column j counting from top to bottom and from left to right respectively. We will call each such matrix as a *block* and the whole row $[\phi_{i1}, \phi_{i2}, \cdots, \phi_{ib}]$ is the i^{th} *row-block*. Each submatrix is a square matrix of dimension p and we ensure without loss of generality, that n is divisible by p so that b is a natural number. For simplicity, current implementation has taken $p = 1$.

The decomposed image is calculated in two major steps and this is the novelty of the algorithm. We start off by decomposing the initial set into Cartesian products of $p-$dimensional

sets $\hat{X}_1, \cdots \hat{X}_b$ using the $\hat{dcp}$ operator described in Cartesian Decomposition section.

After we get $\hat{X}_1, \cdots \hat{X}_b$, every further decomposed set of X' can be calculated by -

$$\hat{X}'_i = \bigoplus_{j=1}^{b} \phi_{ij} \hat{X}_j, \forall j = 1, \cdots b \tag{16}$$

This means the i^{th} decomposed set in reachable set is the cumulative minkowski sum of the product of each ij^{th} block multiplied by the respective j^th decomposed set from initial set. For each i, $\hat{X}'_i$ only depends on the i^{th} row block of ϕ.

The cartesian product $\hat{X}' = X'_1 \times \hat{\cdots} X'_b$, will give us the decomposed image of the equation (14).

So for the k^{th} step, the non-recurrent form for (14) can be written as-

$$X(k) = \phi^k X(0) \tag{17}$$

The decomposed image of the above equation can be written as-

$$\hat{X}_i(k) = \bigoplus_{j=1}^{b} \phi_{ij}^k \hat{X}_j(0) \tag{18}$$

Algorithm 1: Function Reach

Input : A: $n \times n$ Affine Matrix,
τ: time step,
$X(0)$: Initial Set of states,
N: Total steps

Output: $\{\hat{X}(i)\}_N$: N-sized array of overapproximated reach states

$\hat{X}(0) = \hat{dcp}(X(0))$
$\phi = e^{A\tau}$
$Q = \phi$
$B = n/p$
for $i = 1$ **to** N **do**
  $\hat{X}_{tmp} = []$
  for $j = 1$ **to** B **do**
    $\hat{X}_{tmp}[j] = [0_p]$
    for $k = 1$ **to** B **do**
      $\hat{X}_{tmp}[j] = \hat{X}_{tmp}[j] \oplus Q_{jk} \odot \hat{X}(0)[j]$
    end
  end
  $\hat{X}(i) = \hat{X}_{tmp}$
  $Q = Q \times \phi$
end

Remember in the above algorithm ϕ_{ij} means the j^{th} row and k^{th} column block which is a $p \times p$ matrix

B. Cost Analysis

We will compare the costs of (17) and (18). Let us denote the cost of computing the image of an $n \times n$ linear map by $C_\odot(n, m)$ and the cost of computing the Minkowski sum of two n-dimensional sets by $C_\oplus(n, m)$, where m is a parameter that depends on the set representation. For (17), the dimension of $/phi$ matrix is being used for the minkowski sum and the linear map, we we have $n = pb$ and n will be the set dimension and m will be the number of generators in the original problem

$$Cost(17) \in \mathcal{O}(C_\odot(pb, m) + C_\oplus(pb, m)).$$

However for (18), we calculate the minkowski sum for smaller sets and as we mentioned $b = \frac{n}{p}$ where p is the dimension of each set and b are the number of *blocks*, each minkowski sum and linear map will consist sets of p dimension and m' generators,where m' is the generators required to create each of those small sets. We do this b^2 times as we have divided the affine matrix into $b/p \times b/p$ from a $n \times n$matrix

$$Cost(18) \in \mathcal{O}(b^2 C_\odot(p, m') + b^2 C_\oplus(p, m')).$$

Whenever $C_\odot$ and $C_\oplus$ depend at least quadratically on the dimension, and since $m' << m$ due to the overapproximation, the cost of the decomposed image (18) is asymptotically smaller than the cost of the non-decomposed image (17).

Col1	Polyhedra (m vertices)	Zonotope (m generators)	Support Vectors (m directions)
$C_\odot(n, m)$	$\mathcal{O}(mn^2)$	$\mathcal{O}(mn^2)$	$\mathcal{O}(mn^2\mathcal{L})$
$C_\oplus(n, m)$	$\mathcal{O}(m^2 n)$	$\mathcal{O}(n)$	$\mathcal{O}(m\mathcal{L})$

TABLE I

COMPLEXITY OF SET OPERATIONS INVOLVED IN THE AFFINE MAP COMPUTATION BY DECOMPOSITION.($\mathcal{L}$ IS THE COST OF EVALUATING THE SUPPORT FUNCTION OF X. FOR POLYHEDRA IN CONSTRAINT REPRESENTATION WE ASSUME THAT ϕ IS INVERTIBLE; OTHERWISE THE COMPLEXITY IS $\mathcal{O}(mn)$. NOTE THAT m IS NOT COMPARABLE BETWEEN DIFFERENT REPRESENTATIONS

V. IMPLEMENTATION AND PRELIMINARY RESULTS

In this paper, we have implemented the above algorithm with representing the set with a Zonotope implementation of a set of states. Current implementation results below show the a zonotope in 2 dimensions with 3 generators in a Harmonic Oscillator System, finds a bounding box which is the *decomposed* image of the zonotope. The boudning box is stored as a form of 2 1-dimensional zonotopes and then the run the reach algorithm over it. The bounding box is craetd by the cartesian product of the two zonotopes. Linear map, Minkowski sum and Cartesian Product are written in accordance of the two zonotopes. Code will be available on request.

Results for higher dimensions are given below. The zonotopes

Dimensions	Generators	Running Time with (16)	Running Time with (17)
5	30	1.35	1.07
5	50	1.76	1.07
5	250	6.02	1.07
3	30	1.34	0.89
3	50	1.78	0.89
3	250	6.21	0.89

TABLE II

RESULTS FOR DIFFERENT COMBINATIONS OF DIMENSIONS AND GENERATORS

The results above signify show that time taken reachability analysis using cartesian decomposition is quite less than the

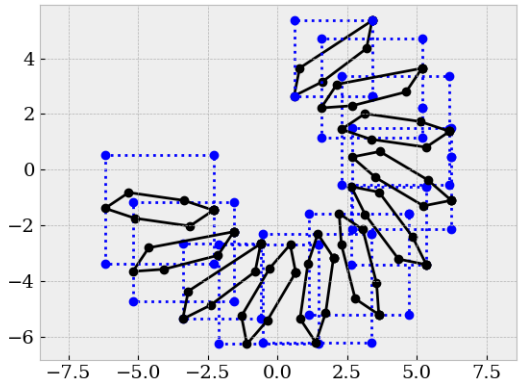

Fig. 5. Reachability Analysis with the Reach algorithm for a Zonotope in a Harmonic oscillator system

actual reachability analysis. With the increase in number of generators, the percentage decrease in time increases even more. This is because of the approximation factor which is currently independent of the m (generators) factor and is only dependent on the dimension n. However, any approximation can be fit in and as it is an approximation, the number of genrators in the approximation (m') will be quite less than m which will ensure in the decrease in time taken to run reachability analysis albeit with tradeoff of precision.

A. Changing value of p

Till now we have mentioned that $p = 1$ i.e. we divide the initial set into multiple smaller sets of dimension 1. However, this value of p could be anything where $n\%p = 0$ in order to divide the set uniformly. Here is an example of a bounding box created after a single step with $p = 1$ and $p = 2$

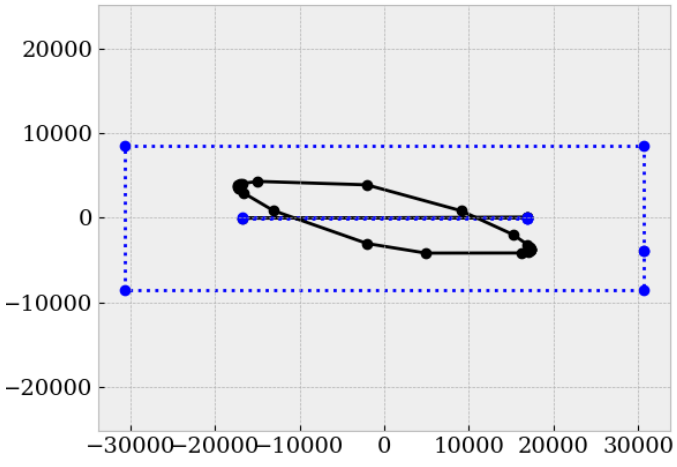

Fig. 6. $p = 1$ Here the output is displayed as the cartesian product of the first two decomposed zonotopes as they represent the first two dimensions

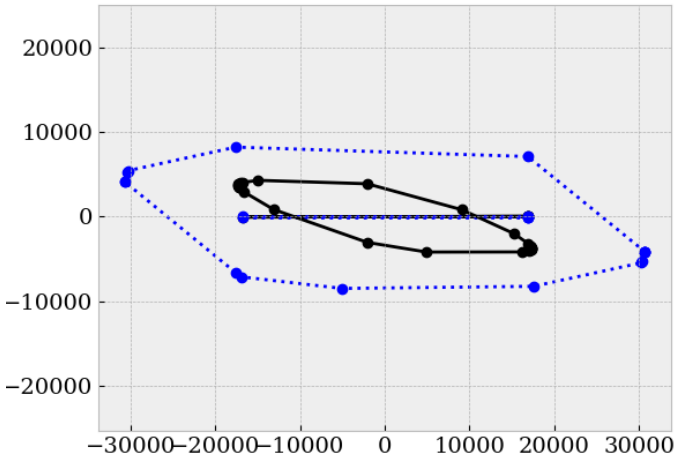

Fig. 7. $p = 2$ Here the output is displayed as first decomposed zonotope as $p = 2$ so the first zonotope will represent the first two dimensions in itself

With an increasing value the bounding box will become much closer to the exact representation as represented above. This is because now the first zonotope has 2 dynamic generators, compared to $p = 1$ case where the first two zonotopes had to be considered and a cartesian product was required to plot, which would always create an axis aligned bounding box in every case. Even in terms of speed for higher values of p, time taken to run the algorithm was being decreased. This could be something expored further as the experiment results in this case show a decrease in running time with increase in p.

p	Running Time
1	1.66
2	1.15
4	1.06
8	1.04

TABLE III
TABLE FOR A SYSTEM WITH $n = 8$ AND $b = \frac{n}{p}$ FOR 600 STEPS AND $m = 250$

REFERENCES

[1] M. Althoff and G. Frehse. Combining zonotopes and support functions for efficient reachability analysis of linear systems. *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995. Hybrid Systems.

[3] E. Asarin and T. Dang. Abstraction by projection and application to multi-affine systems. *Hybrid Systems: Computation and Control Lecture Notes in Computer Science*, page 32–47, 2004.

[4] S. Bak and P. S. Duggirala. Simulation-equivalent reachability of large linear systems with inputs. *Computer Aided Verification Lecture Notes in Computer Science*, page 401–420, 2017.

[5] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling. Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices. *Proceedings of the 21st*

International Conference on Hybrid Systems: Computation and Control (part of CPS Week), 2018.

[6] M. Chen, S. Herbert, and C. J. Tomlin. Exact and efficient hamilton-jacobi guaranteed safety analysis via system decomposition. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[7] X. Chen and S. Sankaranarayanan. Decomposed reachability analysis for nonlinear systems. *2016 IEEE Real-Time Systems Symposium (RTSS)*, 2016.

[8] A. L. Dontchev. Time-scale decomposition of the reachable set of constrained linear systems. *Mathematics of Control, Signals, and Systems*, 5(3):327–340, 1992.

[9] P. S. Duggirala and M. Viswanathan. Parsimonious, simulation based verification of linear systems. *Computer Aided Verification Lecture Notes in Computer Science*, page 477–494, 2016.

[10] A. Girard. Reachability of uncertain linear systems using zonotopes. *Hybrid Systems: Computation and Control Lecture Notes in Computer Science*, page 291–305, 2005.

[11] E. Goncharova and A. Ovseevich. Asymptotics for shapes of singularly perturbed reachable sets *. *IFAC Proceedings Volumes*, 43(14):1391–1393, 2010.

[12] C. L. Guernic and A. Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010.

[13] S. Kaynama and M. Oishi. Complexity reduction through a schur-based decomposition for reachability analysis of linear time-invariant systems. *International Journal of Control*, 84(1):165–179, 2011.

[14] A. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis: internal approximation. *Systems Control Letters*, 41(3):201–211, 2000.

[15] I. Mitchell and C. Tomlin. Overapproximating reachable sets by hamilton-jacobi projections. *Journal of Scientific Computing*, 19, 11 2002.

[16] Y. Seladji and O. Bouissou. Numerical abstract domain using support functions. *Lecture Notes in Computer Science NASA Formal Methods*, page 155–169, 2013.