Automatic Abstraction of Non-Linear Systems Using Change of Bases Transformations.

Sriram Sankaranarayanan *
University of Colorado, Boulder, CO, USA.
firstname.lastname@colorado.edu

ABSTRACT

We present abstraction techniques that transform a given non-linear dynamical system into a linear system, such that, invariant properties of the resulting linear abstraction can be used to infer invariants for the original system. The abstraction techniques rely on a change of bases transformation that associates each state variable of the abstract system with a function involving the state variables of the original system. We present conditions under which a given change of basis transformation for a non-linear system can define an abstraction.

Furthermore, we present a technique to discover, given a non-linear system, if a change of bases transformation involving degree-bounded polynomials yielding a linear system abstraction exists. If so, our technique yields the resulting abstract linear system, as well. This approach is further extended to search for a change of bases transformation that abstracts a given non-linear system into a system of linear differential inclusions. Our techniques enable the use of analysis techniques for linear systems to infer invariants for non-linear systems. We present preliminary evidence of the practical feasibility of our ideas using a prototype implementation.

Categories and Subject Descriptors: F.3.1(Specifying and Verifying and Reasoning about Programs):Invariants,
C.1.m(Miscellaneous): Hybrid Systems.

Terms: Theory, Verification.

Keywords: Ordinary Differential Equations, Hybrid Systems, Algebraic Geometry, Invariants, Verification, Abstraction.

1. INTRODUCTION

> *... the purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise.* – Edsger Dijkstra (ACM Turing Lecture, 1972).

*This material is based upon work supported by the US National Science Foundation (NSF) under Grant No. 0953941. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

HSCC'11, April 12–14, 2011, Chicago, Illinois, USA.

In this paper, we present techniques to search for a "*linear system within*" a given non-linear system. Specifically, we wish to discover affine differential abstractions of continuous systems defined by non-linear differential equations. Given a system of non-linear differential equations, we seek a *change of bases* transformation, mapping the trajectories of the non-linear system into those of a linear abstraction. We present conditions under which a change of bases transformation defines an *abstraction*. Therefore, we can use the invariants for the abstraction to infer invariants for the original system. In this regard, an affine system abstraction is quite useful. Numerous techniques have been proposed to verify safety properties of affine systems efficiently, including *zonotopes* [10], *template polyhedra* [25] and *support functions* [31, 11]. These techniques have been implemented in tools such as HyTech [12], Phaver [9] and TimePass [25, 27]. However, these techniques are mostly restricted to systems with affine dynamics. Relying on these techniques to infer properties of non-linear systems is, therefore, a natural step forward. In this paper, we make the following contributions:

1. We first present techniques for discovering a *linearizing change of bases transformation* that results in a linear abstraction whose dynamics are described by affine ordinary differential equations (ODEs). We prove that a linearizing transformation for a non-linear system corresponds one-to-one to a finite dimensional vector space of functions that also contains the (Lie) derivatives of its elements. The basis functions of a vector space satisfying this closure property yields the desired change of bases transformation. This, in turn, yields the desired linear abstraction.

2. We extend our technique to discover transformations of non-linear ODEs into *differential inequalities*. We show that these transformations are closely related to finitely generated cones of functions that satisfy the property of *closure with a positive semi-definite residue*. We consider two approaches for discovering such cones: one based on the Sum-of-Squares (SOS) relaxation for finding positive semi-definite polynomials [18], and the other based on a simpler, heuristic approach using polyhedral cones over a finite set of posynomials. The result of this abstraction is a non-autonomous linear system with non-negative (disturbance) inputs.

We have implemented our approaches and present interesting preliminary results on finding abstractions for non-linear ODEs. Our implementation, the benchmarks used in the evaluation and the outputs are available on-line or upon request. We motivate our approach on a simple non-linear system.

Example 1.1 (Motivating Example). *Consider a continuous system over* $\{x, y\}$*:* $\dot{x} = xy + 2x, \;\; \dot{y} = -\frac{1}{2}y^2 + 7y + 1$*, with*

initial conditions given by the set $x \in [0,1],\ y \in [0,1]$. *Consider the function* $\alpha : (x,y) \mapsto (w_1, w_2, w_3)$ *wherein* $\alpha_1(x,y) = x$, $\alpha_2(x,y) = xy$ *and* $\alpha_3(x,y) = xy^2$. *The Lie derivatives of* $\alpha_1, \alpha_2, \alpha_3$ *w.r.t the system of ODEs are*

$$\begin{array}{rcl} \frac{d\alpha_1}{dt} & = & xy + 2x \\ & = & \alpha_2(x,y) + 2\alpha_1(x,y) \\ \frac{d\alpha_2}{dt} & = & x + 9xy + \frac{1}{2}xy^2 \\ & = & \alpha_1 + 9\alpha_2 + \frac{1}{2}\alpha_3 \\ \frac{d\alpha_3}{dt} & = & 2xy + 16xy^2 \\ & = & 2\alpha_2 + 16\alpha_3 \end{array}$$

We find that the dynamics over $\vec{w}$ *can be written as*

$$\dot{w_1} = 2w_1 + w_2,\ \dot{w_2} = w_1 + 9w_2 + \frac{1}{2}w_3, \dot{w_3} = 2w_2 + 16w_3$$

Its initial conditions are given by $w_1 \in [0,1],\ w_2 \in [0,1],\ w_3 \in [0,1]$. *We analyze the system using the TimePass tool as presented in our previous work [27] to obtain polyhedral invariants:*

$$\begin{array}{l} -w_1 + 2w_2 \geq -1,\ w_3 \geq 0,\ w_2 \geq 0 \\ -16w_1 + 32w_2 - w_2 \geq -17,\ 32w_2 - w_3 \geq -1,\ w_1 \geq 0 \\ 2w_1 - 4w_2 + 17w_3 \geq -4,\ 286w_1 - 32w_2 + w_3 \geq -32 \end{array}$$

Substituting back, we can infer polynomial inequality invariants on the original system including,

$$\begin{array}{l} -x + 2xy \geq -1, xy^2 \geq 0,\ -16x + 32xy - xy^2 \geq -17, \\ x \geq 0,\ 2x - 4xy + 17xy^2 \geq -4, \cdots \end{array}$$

Note that not every transformation yields a linear abstraction. In fact, most transformations will not define an abstraction. The conditions for an abstraction are discussed in Section 3.

1.1 Related Work

Many different types of *discrete abstractions* have been studied for hybrid systems [2] including predicate abstraction [28] and abstractions based on invariants [17]. The use of counter-example guided abstraction-refinement for iterative refinement has also been investigated in the past (Cf. Alur et al. [1] and Clarke et al. [5], for example). In this paper, we consider continuous abstractions for continuous systems specified as ODEs using a change of bases transformation. As noted above, not all transformations can be used for this purpose. Our abstractions bear similarities to the notion of topological semi-conjugacy between flows of dynamical systems [15].

Reasoning about the reachable set of states for flows of non-linear systems is an important primitive that is used repeatedly in the analysis of non-linear hybrid systems. This has been addressed using a wide variety of techniques in the past, including algebraic and semi-algebraic geometric techniques, interval analysis, constraint propagation and Bernstein polynomials [26, 29, 21, 16, 23, 19, 20, 14, 22, 8]. In particular, the hybridization of non-linear systems is an important approach for converting it into affine systems by subdividing the invariant region into numerous sub-regions and approximating the dynamics as a hybrid system by means of a linear differential inclusion in each region [12, 3, 7]. However, such a subdivision can be expensive as the number of dimensions increases and may not be feasible if the invariant region is unbounded. Our techniques can work on unbounded invariant regions. On the other hand, our abstraction search is *incomplete*. As a result, the techniques presented here may result in a trivial abstraction that does not yield useful information about the system. Nevertheless, we have been able to present some preliminary evidence of usefulness of our ideas over some complex non-linear system benchmarks.

Previous work on invariant generation for hybrid system by the author constructs invariants by assuming a desired template form (ansatz) with unknown parameters and applying the "consecution" conditions such as *strong consecution* and *constant scale* consecution [26]. Matringe et al. present generalizations of these conditions using morphisms [14]. Therein, they observe that strong and constant scale consecution conditions correspond to a linear abstraction of the original non-linear system of a restrictive form. Specifically, the original system is abstracted by a system of the form $\frac{dx}{dt} = 0$ for strong consecution, and a system of the form $\frac{dx}{dt} = \lambda x$ for constant-scale consecution. This paper builds upon this observation by Matringe et al. using fixed-point computation techniques to search for a general linear abstraction that is related to the original system by a change of basis transformation. Moving from equality invariants to inequalities, our work is closely related to the technique of differential invariants proposed by Platzer et al. A key primitive used in this technique can be cast as a search for abstractions of the form $\frac{dx}{dt} \geq 0$ [20]. The approach presented here uses fixed-point computation over cones to search for generalized linear differential inequality abstractions.

Fixed point techniques for deriving invariants of differential equations have been proposed by the author in previous papers [27, 24] These techniques have addressed the derivation of polyhedral invariants for affine systems [27] and algebraic invariants for systems with polynomial right-hand sides [24]. In this technique, we employ the machinery of fixed-points. Our primary goal is not to derive invariants, per se, but to search for abstractions of non-linear systems into linear systems.

Finally, our approach for polynomials is closely related to *Carlemann embedding* that can be used to linearize a given differential equation with polynomial right-hand sides [13]. The standard Carlemann embedding technique creates an infinite dimensional linear system, wherein, each dimension corresponds to a monomial or a basis polynomial. In practice, it is possible to create a linear approximation with known error bounds by truncating the monomial terms beyond a degree cutoff. Our approach for differential equation abstractions can be seen as a search for a "finite submatrix" inside the infinite matrix created by the Carleman linearization. The rows and columns of this submatrix correspond to monomials such that the derivative of each monomial in the submatrix is a linear combination of monomials that belong the submatrix. Our approach of differential inequalities allows for a *residue* involving monomials outside the submatrix that is required to be positive semi-definite.

2. PRELIMINARIES

In this section, we briefly introduce some basic concepts behind multivariate polynomials and hybrid systems. Let $\mathbb{R}$ denote the field of real numbers. Let $x_1, \ldots, x_n$ denote a set of variables, collectively represented as $\vec{x}$. The $\mathbb{R}[\vec{x}]$ denotes the ring of multivariate polynomials over $\mathbb{R}$.

A *monomial* over $\vec{x}$ is of the form $x_1^{r_1} x_2^{r_2} \cdots x_n^{r_n}$, succinctly written as $\vec{x}^{\vec{r}}$, wherein each $r_i \in \mathbb{N}$. A *term* is of the form $c \cdot m$ where $c \in \mathbb{R},\ c \neq 0$ and m is a monomial. The degree of a monomial $\vec{x}^{\vec{r}}$ is given by $\sum_{i=1}^{n} r_i = \vec{1} \cdot \vec{r}$. The degree of a multivariate polynomial p is the maximum over the degrees of all monomials m that *occur* in p with a non-zero coefficient.

Vector Fields: A *vector field* F over a manifold $X \subseteq \mathbb{R}^n$ is a map $F : X \mapsto \mathbb{R}^n$ from each $\vec{x} \in X$ to a vector $F(\vec{x}) \in \mathbb{R}^n$ ($F(\vec{x}) \in T_x(X)$, the tangent space of X at $\vec{x}$). A vector field F is continuous if the map F is continuous. A polynomial vector field $F : X \mapsto \mathbb{R}[\vec{x}]^n$ is specified by a map $F(\vec{x}) = \langle p_1(\vec{x}), p_2(\vec{x}), \ldots, p_n(\vec{x}) \rangle$,

wherein $p_1, \ldots, p_n \in \mathbb{R}[\vec{x}]$. A system of ordinary differential equations D, $\frac{dx_1}{dt} = p_1(x_1, \ldots, x_n)$, $\cdots$, $\frac{dx_n}{dt} = p_n(x_1, \ldots, x_n)$, specifies the evolution of variables $(x_1, \ldots, x_n) \in X$ over time t. The system defines a vector field $F(\vec{x}) : \langle p_1(\vec{x}), \ldots, p_n(\vec{x}) \rangle$.

Def. 2.1 (Lie Derivative). *For a vector field $F : \langle f_1, \ldots, f_m \rangle$, the* Lie derivative *of a smooth function $f(\vec{x})$ is given by*

$$\mathcal{L}_F(f) = (\nabla f) \cdot F(\vec{x}) = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \cdot f_i \right)$$

Henceforth, wherever the vector field F is clear from the context, we will drop subscripts and use $\mathcal{L}(p)$ to denote the Lie derivative of p w.r.t F.

We assume that all vector fields F considered in this paper are (locally) Lipschitz continuous over the domain X. In general, all polynomial vector fields are locally Lipschitz continuous, but not necessarily *globally* Lipschitz continuous over an unbounded domain X. The Lipschitz continuity of the vector field F, ensures that given $\vec{x} = \vec{x}_0$, there exists a time $T > 0$ and a unique time trajectory $\tau : [0, T) \mapsto \mathbb{R}^n$ such that $\tau(t) = \vec{x}_0$ [15].

Def. 2.2 (Continuous System). *A continuous system over variables $x_1, \ldots, x_n$ consists of a tuple $\mathcal{S} : \langle X_0, \mathcal{F}, X_I \rangle$ wherein $X_0 \subseteq \mathbb{R}^n$ is the set of initial states, $\mathcal{F}$ is a vector field over the domain $X_I \subseteq \mathbb{R}^n$.*

Note that in the context of hybrid systems, the set X_I is often referred to as the *state invariant* or the *domain*.

2.1 Hybrid Systems

Hybrid systems consists of continuous state variables and a finite set of discrete modes. The dynamics of the continuous state variables are a function of the system's current discrete mode. Furthermore, the system performs (instantaneous) mode changes upon encountering a switching condition (or a transition guard).

Def. 2.3 (Hybrid System). *An hybrid system is a tuple $\langle \mathcal{S}, \mathcal{T} \rangle$, wherein $\mathcal{S} = \{S_1, \ldots, S_k\}$ consists of k discrete modes and $\mathcal{T}$ denotes discrete transitions between the modes. Each mode $S_i \in \mathcal{S}$ is a continuous sub-system $\langle X_{0,i}, F_i, X_i \rangle$, defined by the vector field $\mathcal{F}_i$, the initial conditions $X_{0,i}$ and the invariance condition X_i.*

Each transition $\tau : \langle S_i, S_j, P_{ij} \rangle \in \mathcal{T}$ consists of an edge $S_i \rightarrow S_j$ along with an transition relation *$P_{ij}[\vec{x}, \vec{x}']$ specifying the next state $\vec{x}'$ in relation to the previous state $\vec{x}$. Note that the transition is* guarded *by the assertion $\exists \vec{x}' P_{ij}[\vec{x}, \vec{x}']$.*

3. CHANGE OF BASES TRANSFORMATION

In this section, we will present change of bases (CoB) abstractions and some of their properties.

Consider a map $\alpha : \mathbb{R}^k \mapsto \mathbb{R}^l$. Given a set $S \subseteq \mathbb{R}^k$, let $\alpha(S)$ denote the set obtained by applying α to all the elements of S. Similarly, the inverse map over sets is $\alpha^{-1}(T) : \{s \mid \alpha(s) \in T\}$. Let $\mathcal{S} : \langle X_0, \mathcal{F}, X_I \rangle$ be a continuous system over variables $\vec{x} : (x_1, \ldots, x_n)$ and $\mathcal{T} : \langle Y_0, \mathcal{G}, Y_I \rangle$ be a continuous system over variables $\vec{y} : (y_1, \ldots, y_m)$.

Def. 3.1 (Simulation). *We say that $\mathcal{T}$* simulates *$\mathcal{S}$ iff there exists a smooth mapping $\alpha : \mathbb{R}^n \mapsto \mathbb{R}^m$ such that*

1. *$Y_0 \supseteq \alpha(X_0)$ and $Y_I \supseteq \alpha(X_I)$.*
2. *For any trajectory $\tau : [0, T) \mapsto X_I$ of $\mathcal{S}$, $\alpha \circ \tau$ is a trajectory of $\mathcal{T}$.*

A simulation relation implies that any time trajectory of $\mathcal{S}$ can be mapped to a trajectory of $\mathcal{T}$ through α. However, since α need not be invertible, the converse need not hold. I.e, $\mathcal{T}$ may exhibit time trajectories that are not mapped onto by any trajectory in $\mathcal{S}$.

Let $\mathcal{S}$ and $\mathcal{T}$ be defined by Lipschitz continuous vector fields. The following theorem enables us to check given $\mathcal{S}$ and $\mathcal{T}$, whether $\mathcal{T}$ simulates $\mathcal{S}$.

Theorem 3.1. *$\mathcal{T}$ simulates $\mathcal{S}$ if the following conditions hold:*

1. *$Y_0 \supseteq \alpha(X_0)$.*
2. *$Y_I \supseteq \alpha(X_I)$.*
3. *$\mathcal{G}(\alpha(\vec{x})) = J_\alpha . \mathcal{F}(\vec{x})$, wherein, J_α is the Jacobian*

$$J_\alpha(x_1, \ldots, x_n) = \begin{bmatrix} \frac{\partial \alpha_1}{\partial x_1} & \cdots & \frac{\partial \alpha_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \alpha_m}{\partial x_1} & \cdots & \frac{\partial \alpha_m}{\partial x_n} \end{bmatrix},$$

and $\alpha(\vec{x}) = (\alpha_1(\vec{x}), \cdots, \alpha_m(\vec{x}))$, $\alpha_i : \mathbb{R}^n \mapsto \mathbb{R}$.

PROOF. Let τ_x be a trajectory over $\vec{x}$ for system $\mathcal{S}$. Note that at any time instant $t \in [0, t)$, $\frac{d\tau_x}{dt} = \mathcal{F}(\tau(t))$.

We wish to show that $\tau_y(t) = \alpha(\tau_x(t))$ is a time trajectory for the system $\mathcal{T}$. Since, $\tau_x(0) \in X_0$, we conclude that $\tau_y(0) = \alpha(\tau_x(0)) \in Y_0$. Since $\tau_x(t) \in X_I$ for all $t \in [0, T)$, we have that $\tau_y(t) = \alpha(\tau_x(t)) \in Y_I$. Differentiating τ_y we get,

$$\begin{array}{rclcl} \frac{d\tau_y}{dt} & = & \frac{d\alpha(\tau_x(t))}{dt} & = & J_\alpha \cdot \frac{d\tau_x}{dt} \quad = \quad J_\alpha \cdot \mathcal{F}(\tau_x(t)) \\ & = & \mathcal{G}(\alpha(\tau_x(t))) & = & \mathcal{G}(\tau_y(t)) \,. \end{array}$$

Therefore $\tau_y = \alpha \circ \tau_x$ conforms to the dynamics of $\mathcal{T}$. By Lipschitz continuity of $\mathcal{G}$, we obtain that τ_y is the unique trajectory starting from $\alpha \circ \tau(0)$. □

Note that, in general, a trajectory $\tau_y(t) = \alpha(\tau_x(t))$ may exist for a longer interval of time than the interval $[0, T)$ over which τ_x is assumed to be defined.

Theorem 3.2. *Let $\mathcal{T}$ simulate $\mathcal{S}$ through a map α. If $Y \subseteq Y_I$ is a positive invariant set for $\mathcal{T}$ then $\alpha^{-1}(Y) \cap X_I$ is a positive invariant set for $\mathcal{S}$.*

PROOF. Assuming otherwise, let τ_x be a time trajectory that starts from inside $\alpha^{-1}(Y) \cap X_I$ and has a time instant t such that $\tau_x(t) \notin \alpha^{-1}(Y) \cap X_I$. Since we defined time trajectories so that $\tau_x(t) \in X_I$, it follows that $\tau_x(t) \notin \alpha^{-1}(Y)$. As a result, $\alpha(\tau_x(t)) \notin Y$. Therefore, corresponding to τ_x, we define a new trajectory $\tau_y = \alpha \circ \tau_x$ which violates the positive invariance of Y. This leads to a contradiction. □

An application of the Theorem above is illustrated in Example 1.1.

Example 3.1. *Consider a mechanical system $\mathcal{S}$ expressed in generalized position coordinates (q_1, q_2) and momenta (p_1, p_2) defined using the following vector field:*

$$F(p_1, p_2, q_1, q_2) : \langle \; -2q_1q_2^2, \; -2q_1^2q_2, \; 2p_1, \; 2p_2 \; \rangle$$

with the initial conditions: $(p_1, p_2) \in [-1, 1] \times [-1, 1] \wedge (q_1, q_2) : (2, 2)$. Using the transformation $\alpha(p_1, p_2, q_1, q_2) : p_1^2 + p_2^2 + q_1^2q_2^2$, we see that $\mathcal{S}$ is simulated by a linear system $\mathcal{T}$ over y, with dynamics given by $\frac{dy}{dt} = 0$, $y(0) \in [16, 18]$.

Incidentally, the form of the system $\mathcal{T}$ above indicates that α is an expression for a conserved quantity (in this case, the Hamiltonian) of the system.

3.1 Linearizing CoB Transformations

In this section, we define the notion of a linearizing CoB transformation. An affine system $\mathcal{T}$ is described by an affine vector field $\frac{d\vec{y}}{dt} = A\vec{y} + \vec{b}$ for an $m \times m$ matrix A and an $m \times 1$ vector $\vec{b}$.

Def. 3.2 (Linearizing CoB Transformation). *Let $\mathcal{S}$ be a (nonlinear) system. We say that α is a linearizing CoB transformation if it maps each trajectory of $\mathcal{S}$ to that of an affine system $\mathcal{T}$. In other words, α ensures that $\mathcal{S}$ is simulated by an affine system $\mathcal{T}$.*

The above definition of a linearizing CoB seems useful, in practice, only if α and $\mathcal{T}$ are already known. We may then use known techniques for safely bounding the reachable set of an affine system, given some initial conditions, and transform the result back through α^{-1} to obtain a bound on the reachable set for $\mathcal{S}$.

We now present a technique that searches for a map α to obtain an affine system $\mathcal{T}$ that simulates a given system $\mathcal{S}$ through α. We ignore the initial condition and invariant, for the time being, and simply focus on the dynamics of $\mathcal{T}$. In other words, we will search for a map α that satisfies

$$J_\alpha(\vec{x}) \cdot \mathcal{F}(\vec{x}) = A\alpha(\vec{x}) + \vec{b}$$

for some constant matrices A, b. Having found such a map, we can always find appropriate initial and invariance conditions for the simulating system $\mathcal{T}$, whose dynamics will be given by $\mathcal{G}(\vec{y}) = A\vec{y} + \vec{b}$ so that Definition 3.1 holds.

We proceed by recasting a linearizing CoB transformation in terms of a vector space that is closed under the action of taking Lie-derivatives.

3.2 Vector Spaces Closed Under Lie Derivatives.

Recall the requirement for α serving as a linearizing change of variables transformation for a vector field $\mathcal{F}$:

$$J_\alpha(\vec{x}) \cdot \mathcal{F}(\vec{x}) = A\alpha(\vec{x}) + \vec{b}$$

for some constant matrices A, b.

Let $\alpha(\vec{x}) : (\alpha_1(\vec{x}), \ldots, \alpha_m(\vec{x}))$ be a smooth mapping $\alpha : \mathbb{R}^n \mapsto \mathbb{R}^m$, wherein each $\alpha_i : \mathbb{R}^n \mapsto \mathbb{R}$. Recall that $\mathcal{L}_F(\alpha_i(\vec{x})) = (\nabla\alpha_i) \cdot \mathcal{F}(\vec{x})$ denotes the Lie derivative of the function $\alpha_i(\vec{x})$ w.r.t vector field $\mathcal{F}$.

Lemma 3.1. $J_\alpha \cdot \mathcal{F}(\vec{x}) = \begin{pmatrix} \mathcal{L}_F(\alpha_1(\vec{x})) \\ \mathcal{L}_F(\alpha_2(\vec{x})) \\ \vdots \\ \mathcal{L}_F(\alpha_m(\vec{x})) \end{pmatrix}$.

PROOF. Recall the definition of the Jacobian J_α:

$$J_\alpha(x_1, \ldots, x_n) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla\alpha_1 \\ \vdots \\ \nabla\alpha_m \end{bmatrix}.$$

It follows that,

$$J_\alpha.\mathcal{F} = \begin{pmatrix} (\nabla\alpha_1) \cdot (\mathcal{F}) \\ (\nabla\alpha_2) \cdot (\mathcal{F}) \\ \vdots \\ (\nabla\alpha_m) \cdot (\mathcal{F}) \end{pmatrix} = \begin{pmatrix} \mathcal{L}_F(\alpha_1(\vec{x})) \\ \mathcal{L}_F(\alpha_2(\vec{x})) \\ \vdots \\ \mathcal{L}_F(\alpha_m(\vec{x})) \end{pmatrix}.$$

□

Given functions $\alpha_1, \ldots, \alpha_m : \mathbb{R}^n \mapsto \mathbb{R}$, and the special constant function $\mathbf{1} : \mathbb{R}^n \mapsto \{1\}$, we consider the vector space generated by these functions:

$$Span\,(\mathbf{1}, \alpha_1, \ldots, \alpha_m) = \left\{ c_0 \cdot \mathbf{1} + \sum_{i=1}^{m} c_i\alpha_i \mid c_0, \ldots, c_m \in \mathbb{R} \right\}.$$

Theorem 3.3 (Vector Space Closure Theorem). *A map $\alpha : (\alpha_1, \ldots, \alpha_m)$ represents a linearizing CoB transformation for a system iff the vector space $V : Span(\mathbf{1}, \alpha_1, \ldots, \alpha_m)$ is closed under the operation of Lie-derivatives. I.e, $\forall g \in V$, $\mathcal{L}_F(g) \in V$.*

PROOF. Let α be a linearizing CoB transformation mapping trajectories of $\mathcal{F}$ onto $\mathcal{G} : \frac{d\vec{y}}{dt} = A\vec{y} + \vec{b}$. Therefore, for each α_i,

$$\mathcal{L}_F(\alpha_i) = b_i + \sum_j A_{ij}\alpha_j \tag{1}$$

Any element $\beta \in V$ can be written as $\beta = c_0 + \sum_k c_k\alpha_k$ (a linear combination of the bases of the vector space). Using (1), $\mathcal{L}_F(\beta)$ can be written, once again, as a linear combination of α_is and $\mathbf{1}$. Therefore $\mathcal{L}_F(\beta) \in V$.

Conversely, if V is closed under the action of a Lie-derivative, then its bases $\mathbf{1}, \alpha_1, \ldots, \alpha_m$ satisfy the condition $\mathcal{L}_F(\alpha_i) = b_i\mathbf{1} + \sum_j a_{ij}\alpha_j$.

From Lemma 3.1, $J_\alpha\mathcal{F}(\vec{x}) = \begin{pmatrix} \mathcal{L}_F(\alpha_1(\vec{x})) \\ \mathcal{L}_F(\alpha_2(\vec{x})) \\ \vdots \\ \mathcal{L}_F(\alpha_m(\vec{x})) \end{pmatrix} = A\alpha(\vec{x}) + \vec{b}$,

wherein, $A = [a_{ij}]$ and $\vec{b} = (b_i)$. This proves that α is a linearizing CoB transformation. □

Example 3.2. *Consider the ODE from Example 1.1 recalled below:*

$$\begin{array}{rcl} \frac{dx}{dt} & = & xy + 2x \\ \frac{dy}{dt} & = & -\frac{1}{2}y^2 + 7y + 1 \end{array}$$

We claim that the vector space V generated by the set of functions $\{\mathbf{1}, xy, xy^2, x\}$ is closed under the operation of computing Lie derivatives. To verify, we compute the Lie derivative of a function of the form $c_0 + c_1x + c_2xy + c_3xy^2$ to obtain

$$c_1(xy + 2x) + c_2(\frac{1}{2}xy^2 + 9xy + x) + c_3(9xy^2 + 2xy)$$

which is seen to belong to V. As a result the CoB abstraction $\alpha(x, y) : (x, xy, xy^2)$ linearizes the system.

4. SEARCHING FOR ABSTRACTIONS

In this section, we will present search strategies for finding a linearizing change of bases abstraction, if one exists. Following Theorem 3.3, our goal is to find a vector space generated by some functions $\alpha_1, \ldots, \alpha_k$ that are closed under the action of taking Lie derivatives. Given a set of functions $B = \{f_1, \ldots, f_k\}$, we write $Span(B)$ to denote the vector space spanned by the functions in B:

$$Span(B) = \left\{ \sum_{j=1}^{k} a_j f_j(\vec{x}) \mid a_j \in \mathbb{R} \right\}$$

We will proceed using a subspace iteration as follows:

1. Choose an initial vector space $V_0 = Span\,(\{\alpha_0, \ldots, \alpha_N\})$. For instance, V_0 can be generated by all *monomial terms whose degrees are less than a cutoff*. In general, any *ansatz* of the form $\sum_i c_i f_i$, for functions $f_i(\vec{x})$ and parameters c_i, can be written as a vector space $V_0 = Span(\{f_0, \ldots, f_k\})$.

2. At each step, iteratively *refine* V_i for $i \geq 0$ to yield V_{i+1}, a subspace of V_i.

3. Stop when $V_{n+1} = V_n$. If V_n is a non-trivial vector space then, a non-trivial, linearizing change of bases transformation can be extracted along with the resulting system from the generators of V_n.

Initial Basis: For ODEs with polynomial right-hand sides, the initial basis can be generated by all monomial terms up to some degree bound d. However, our technique can be extended to handle other types of functions including trigonometric functions as long as these functions are continuous and differentiable.

Let $V_0 : Span\,(\{\mathbf{1}, \alpha_0, \ldots, \alpha_N\})$ be the initial basis.

Refining the Basis: Let $\{\mathbf{1}, \alpha_1, \ldots, \alpha_k\}$ be the basis of the vector space V_i for the i^{th} iteration. Our goal is to refine this basis to find a subspace $V_{i+1} \subseteq V_i$ such that

$$V_{i+1} = \mathcal{D}(V_i) = \{f \in V_i \mid \mathcal{L}_F(f) \in V_i\}$$

Note that by definition, V_{i+1} is a subset of V_i. It remains to show that V_{i+1} is a vector space.

Lemma 4.1. *If V_i is a vector space, then $V_{i+1} = \mathcal{D}(V_i)$ is a subspace of V_i.*

PROOF. Let V_i be generated by the basis $\{\mathbf{1}, \alpha_1, \ldots, \alpha_k\}$. That $V_{i+1} \subseteq V_i$ follows from its definition. It remains to show that V_{i+1} is a vector space. First $\mathbf{1} \in V_{i+1}$. Let $f_1, \ldots, f_l$ be functions in V_{i+1}. Therefore, by definition, $\mathcal{L}_F(f_1), \ldots, \mathcal{L}_F(f_l) \in V_i$. Consider their affine combination $f : a_0\mathbf{1} + \sum_{j=1}^{l} a_j f_j$ for some $a_0, a_1, \ldots, a_l \in \mathbb{R}$. Its Lie derivative is

$$\mathcal{L}_F(a_0 + \sum_{j=1}^{l} a_j f_j) = a_j \sum_{j=1}^{l} \mathcal{L}_F(f_j)\,.$$

Therefore, $\mathcal{L}_F(f)$ can be written as a linear combination of the Lie derivatives of $f_1, \ldots, f_l$ which are themselves in V_i. Therefore, $\mathcal{L}_F(f) \in V_i$ and therefore $f \in V_{i+1}$. Thus, any linear combination of elements of V_{i+1} also belongs to V_{i+1}. □

Theorem 4.1. *Given an initial vector space V_0 and vector field $\mathcal{F}$, the subspace iteration converges in finitely many steps to a subspace $V^* \subseteq V_0$. Let $\alpha_1, \ldots, \alpha_m$ be the basis functions that generate V^*.*

1. *The transformation $\alpha : (\alpha_1, \ldots, \alpha_m)$ generated by the basis functions of the final vector space is linearizing.*

2. *For every linearizing CoB transformation $\beta : (\beta_1, \ldots, \beta_k)$, wherein each $\beta_i \in V_0$, it follows that $\beta_i \in V^*$.*

PROOF. The convergence of the iteration follows from the observation that if $V_{i+1} \subset V_i$, the dimension of V_{i+1} is at least one less than that of V_i. Since V_0 is finite dimensional, the number of iterations is upper bounded by the number of basis functions in V_0.

The first statement follows directly from Theorem 3.3.

Finally, us assume that a linearizing transformation β exists such that $\beta_i \in V_0$. We note that the space U generated by $\mathbf{1}, \beta_1, \ldots, \beta_k$ is a subset of V_0. We can also prove that if $U \subseteq V_i$, then $U \subseteq V_{i+1}$. As a result, we prove by induction that $U \subseteq V^*$. □

Example 4.1. *Consider the system $\mathcal{F}$ from Example 1.1 recalled below:*

$$\begin{array}{rcl} \frac{dx}{dt} & = & xy + 2x \\ \frac{dy}{dt} & = & -\frac{1}{2}y^2 + 7y + 1 \end{array}$$

The initial basis for V_0 can be chosen to be the set of all monomials whose degree is less than some limit d. For simplicity, let us choose the basis for V_0 to be: $\{\mathbf{1}, x, y, xy, x^2, y^2, x^2y, xy^2\}$. Any element of V_0 can be written as

$$f : c_0\mathbf{1} + c_1x + c_2y + c_3xy + c_4x^2 + c_5y^2 + c_6x^2y + c_7xy^2\,.$$

Its Lie derivative can be written as:

$$\mathcal{L}_F(f) : \left(\begin{array}{l} c_2 + (2c_1 + c_3)x + (7c_2 + 2c_5)y + \\ (c_1 + 9c_3 + 2c_7)xy + (2c_4 + c_6)x^2 \\ -\frac{1}{2}(c_2 + 14c_5)y^2 + (2c_4 + 11c_6)x^2y + \\ (\frac{1}{2}c_3 + 16c_7)xy^2 - c_5y^3 + \frac{3c_6}{2}x^2y^2 \end{array}\right)$$

We note that $\mathcal{L}_F(f) \in V_0$ iff $c_5 = 0,\ c_6 = 0$ (so that terms corresponding to y^3, x^2y^2 vanish. This yields the bases for V_1 :

$$\{\mathbf{1}, x, y, xy, x^2, xy^2\}$$

Once again, computing the Lie derivative yields the constraint: $c_2 = 0, c_4 = 0$, yielding the basis:

$$\{\mathbf{1}, x, xy, xy^2\}$$

The iteration converges in two steps, yielding the linearizing transformation $\alpha : (x, xy, y^2)$, as expected.

Note that it is possible for the converged result V^* to be trivial. I.e, it is generated by the constant function $\mathbf{1}$.

Example 4.2. *Consider the van der Pol oscillator whose dynamics are given by*

$$\dot{x} = y,\ \dot{y} = \mu(y - \frac{1}{3}y^3 - x)\,.$$

Our search for polynomials ($\mu = 1$) of degree up to 20 did not yield a non-trivial transformation.

For a trivial system, the resulting affine system $\mathcal{T}$ is $\frac{dy}{dt} = 0$ under the map $\alpha(\vec{x}) = 0$. Naturally, this situation is not quite interesting but will often result, depending on the system $\mathcal{S}$ and the initial basis chosen V_0. We now discuss common situations where the vector space V^* obtained as the result is guaranteed to be non-trivial. Section 5 presents techniques that can search for a broader class of affine differential inequations instead of just equations.

4.1 Strong and Constant Scale Consecution

The notion of "strong" consecution, "constant scale" consecution and "polynomial scale" consecution were defined for equality invariants of differential equations in our previous work [26] and subsequently expanded upon by Matringe et al. [14] using the notion of morphisms. We now show that the techniques presented in this section can capture these notions, ensuring that all the systems handled by the techniques presented in our previous work [26] can be handled by the techniques here (but not vice-versa).

Def. 4.1 (Strong and Constant Scale Invariants). *A function f satisfies the* strong scale *consecution requirement for a vector field $\mathcal{F}$ iff $\mathcal{L}_F(f) = 0$. In other words, f is a conserved quantity.*

Similarly, f satisfies the constant scale *consecution iff $\exists\lambda \in \mathbb{R},\ \mathcal{L}_F(f) = \lambda f$.*

The following theorem is a corollary of Theorem 4.1 and shows that the ideas presented in this section can capture the notion of strong and constant scale consecution without requiring quantifier elimination, solving an eigenvalue problem [26] or finding roots of a univariate polynomial [14].

Theorem 4.2. *The result of the iteration V^* starting from an initial space V_0 contains all the strong and constant scale invariant functions in V_0.*

PROOF. This is a direct consequence of Theorem 4.1 by noting that for a constant scale consecuting function f, the subspace $U \subseteq V_0$ spanned by f is closed under Lie derivatives. □

Furthermore, if such functions exist in V_0 the result after convergence V^* is guaranteed to be a non-trivial vector space (of positive dimension). Finally, constant scale and strong scale functions can be extracted by computing the affine equality invariants of the linear system $\mathcal{T}$ that can be extracted from V^*.

4.1.1 Stability

We briefly address the issue of deducing stability (or instability) of a system $\mathcal{S}$ using an abstraction to a system $\mathcal{T}$. Note that every equilibrium of $\mathcal{S}$ maps onto an equilibrium of $\mathcal{T}$, but not vice-versa. Furthermore, the map $\alpha(\vec{x}) = (\mathbf{0}, \ldots, \mathbf{0})$ is an abstraction from any non-linear system to one with an equilibrium at origin. Therefore, unless restrictions are placed on α, we are unable to draw conclusions on liveness properties for $\mathcal{S}$ based on $\mathcal{T}$. If α has a continuous inverse, then $\mathcal{T}$ is topologically diffeomorphic to $\mathcal{S}$ [15]. This allows us to correlate equilibria of $\mathcal{T}$ with those of $\mathcal{S}$. The preservation of stability under mappings of state variables has been studied by Vassilyev and Ul'yanov [32]. We are currently investigating restrictions that will allow us to draw conclusions about liveness properties of $\mathcal{S}$ from those of $\mathcal{T}$.

5. DIFFERENTIAL INEQUALITIES

In this section, we extend our results to search for transformations that result in an affine differential inequality rather than an equality. Affine differential inequalities represent a broader class of systems that include equalities. Therefore, we expect to find non-trivial affine differential inequality abstractions for a larger class of non-linear systems.

A function $f(\vec{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ is *positive semi-definite* (psd) iff $f(\vec{x}) \geq 0$ for all $\vec{x} \in \mathsf{dom}(f)$. A function is positive definite iff it is positive semi-definite and non-zero everywhere.

Def. 5.1 (Affine Differential Inequality). *An* affine differential inequality *is a non-autonomous system of the form:*

$$\frac{d\vec{y}}{dt} = A\vec{y} + \vec{b} + \vec{u}(t)\,,$$

for $m \times m$ matrix A, $m \times 1$ vector $\vec{b}$ and disturbance inputs $\vec{u}(t) : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}^m$. The disturbance inputs are assumed to be integrable *and positive semi-definite. Since the input $\vec{u}$ is psd, we write the differential inequality informally as: $\frac{d\vec{y}}{dt} \geq A\vec{y} + \vec{b}$.*

In order to define a transformation that results in an abstract system of differential inequalities, we first define the notion of a set of functions that are closed under a Lie derivative with a *positive semi-definite residue*.

Def. 5.2 (Closure with PSD Residue). *A finite set of functions $S : \{\alpha_1, \ldots, \alpha_k\}$ is closed under Lie derivatives with a* positive semi-definite *residue iff for all $\alpha_i \in S$, the Lie derivative of α_i is of the form:*

$$\mathcal{L}_F(\alpha_i) = b_i\mathbf{1} + \sum_j a_{ij}\alpha_j + \rho_i(\vec{x})\,,$$

wherein, a_{ij}, b_i are real-valued constants, and ρ_i is a continuous, positive semi-definite function over $\vec{x}$.

Example 5.1 (Projectile with Energy Dissipation). *Consider the dynamics of a projectile moving "upwards" ($v_y \leq 0$) with dissipation of its kinetic energy due to friction. The variables (x, y) represent the position and (v_x, v_y) its velocity vector. We assume that the drag due to friction is a polynomial function of the velocity. The dynamics are given by $\dot{x} = v_x$, $\dot{y} = v_y$, $\dot{v_x} = -\frac{1}{10}v_x - \frac{1}{100}v_x^2$, $\dot{v_y} = -10 - \frac{1}{10}v_y - \frac{1}{100}v_y^2$. The system operates in the region $v_x \geq 0$ and $v_y \leq 0$.*

Consider the functions $\alpha_1 : -10v_y - y$ and $\alpha_2 : -10v_x - x$. We have $\mathcal{L}(\alpha_1) = 100 + .1v_y^2 = 100 + \rho_1$, wherein $\rho_1 = .1v_y^2$ is psd. Similarly, $\mathcal{L}(\alpha_2) = \rho_2$, for psd $\rho_2 = .1v_x^2$. Therefore, the set $S = \{\alpha_1, \alpha_2\}$ satisfies the conditions of Def. 5.2.

We now establish the connection between sets of functions that are closed under Lie derivatives with a psd residue and affine differential inequalities. Consider a Lipschitz continuous vector field $\mathcal{F}$ over X_I. Let $S : \{\alpha_1, \ldots, \alpha_k\}$ be a finite set of continuous and differentiable functions that are closed under Lie derivatives with a psd residue (Def. 5.2).

For a differential inequality $\frac{d\vec{y}}{dt} \geq A\vec{y}+\vec{b}$, we let $s_{y_0,u} : [0, T') \mapsto \mathbb{R}^m$ denote the unique time trajectory for the ODE $\frac{d\vec{y}}{dt} = A\vec{y} + \vec{b} + \vec{u}(t)$, with initial conditions $s(0) = \vec{y}_0$ and with a continuous input $\vec{u}(t)$ (such that $\vec{u}(t) \geq 0$ for all $t \geq 0$).

Theorem 5.1. *If a finite set $S : \{\alpha_1, \ldots, \alpha_m\}$ exists that is closed under Lie derivatives with a psd residue, then there exists an affine differential inequality $\frac{d\vec{y}}{dt} \geq A\vec{y} + \vec{b}$ over m variables $y_1, \ldots, y_m$, such that for each time trajectory $\tau_x : [0, T) \mapsto \mathbb{R}^n$ for $\mathcal{F}$, there exists a continuous, positive semi-definite input function $\vec{u}(t) : [0, T) \to \mathbb{R}^m$ such that*

$$\alpha(\tau_x(t)) = s_{\alpha(\vec{x}_0),u}(t),\ \forall\, t \in [0, T)$$

PROOF. Consider the Lie derivative for each α_i. It can be written as $\mathcal{L}_F(\alpha_i) = b_i + \sum_{j=1}^{k} a_{jk}\alpha_j + \rho_i$. Fixing some trajectory $\tau : [0, T) \mapsto \mathbb{R}^n$ for $\mathcal{F}$, let $\vec{u}(t) : (\rho_1(\tau(t)), \ldots, \rho_k(\tau(t)))$ be obtained by evaluating the residues over the trajectory τ. The continuity of τ and ρ_i imply the continuity and thus, the integrability of $\vec{u}$. As a result, the trajectory τ is mapped by $\alpha : (\alpha_1(\vec{x}), \ldots, \alpha_k(\vec{x}))$ to a trajectory of the ODE $\frac{d\vec{y}}{dt} = A\vec{y}+\vec{b}+\vec{u}(t)$, which is a trajectory of the inequality $\frac{d\vec{y}}{dt} \geq A\vec{y} + \vec{b}$. □

A set Y is a positive invariant set for a differential inequality $\vec{y}' \geq A\vec{y} + \vec{b}$ iff starting from any point $\vec{y}_0 \in Y$ and for any input function $\vec{u}(t)$ that is integrable and positive semi-definite, the resulting trajectory $s_{y_0,u}$ lies entirely in Y.

Theorem 5.2. *Let Y be an invariant set for the differential inequality $\vec{y}' \geq A\vec{y} + \vec{b}$ that abstracts a system $\mathcal{S} : \langle X_0, \mathcal{F}, X_I\rangle$ through a map α. It follows that $\alpha^{-1}(Y) \cap X_I$ is a positive invariant set for $\mathcal{S}$.*

Example 5.2. *Continuing with the system in Ex. 5.1, we find that the differential inequality: $\frac{dy_1}{dt} \geq 100$, $\frac{dy_2}{dt} \geq 0$ abstracts the dynamics of the system.*

Example 5.3. *Consider the system:*

$$\frac{dx}{dt} = x + 2xy,\ \frac{dy}{dt} = 1 + 3y - y^2$$

A simple examination suggests that $\alpha_0(x, y) = -y$ has a positive semi-definite residue. However, consider the functions $\alpha_1(x, y) : x^2$ and $\alpha_2(x, y) : 2x^2y - x^2$. We find that $\mathcal{L}(\alpha_1) : 2x^2 + 4x^2y =$

$2\alpha_2 + 4\alpha_1,\ \mathcal{L}(\alpha_2):\ 6x^2y + 6x^2y^2 = 3\alpha_2 + 3\alpha_1 + \rho$, *wherein* $\rho = 6x^2y^2$. *Therefore, the system*

$$\frac{dy_1}{dt} \geq -1 - 3y_1,\ \frac{dy_2}{dt} = 4y_2 + 2y_3,\ \frac{dy_3}{dt} \geq 3y_2 + 3y_3\,,$$

is obtained from the map α *as an abstraction.*

Having described CoB transformations for differential inequalities, we now focus on a best-effort algorithm for finding a CoB transformation. The difficulty in discovering an abstraction arises from the fact that (a) finding if a given polynomial is psd is NP-hard [1] and (b) the components of the mapping α do not form a structure such as vector spaces over which a terminating iteration can be readily defined.

5.1 Finding Polynomial Abstractions

For the remainder of this section, we focus on discovering inequality abstractions for ODEs with polynomial right-hand sides through maps α that involve polynomials. Note that the term *posynomial* refers to a positive semi-definite polynomial. Before proceeding, we recall the definition *finitely generated cones*.

Def. 5.3 (Finitely Generated Cone). *A* finitely generated cone $C = \mathsf{cone}(\alpha_0, \ldots, \alpha_k)$ *is the set of all functions obtained as conic combinations of its generators:* $C = \{f : \sum_{j=0}^{k} \lambda_j \alpha_j \mid \lambda_j \geq 0,\ \lambda_j \in \mathbb{R}\}$.

A finitely generated cone $C:\ \mathsf{cone}(\alpha_1, \ldots, \alpha_k)$ is closed under Lie derivatives w.r.t a vector field $\mathcal{F}$ *with a positive semidefinite (psd) residue* iff

$$\forall\, f \in C,\ \mathcal{L}_F(f) = a_0 + \sum_i a_i\alpha_i + \rho,\ \text{for positive semi-definite } \rho\,.$$

Lemma 5.1. *A set* $S = \{\alpha_1, \ldots, \alpha_k\}$ *is closed under Lie derivatives w.r.t* $\mathcal{F}$ *with a psd residue iff the* $\mathsf{cone}(S)$ *is also closed with a psd residue.*

5.1.1 Approach

Along the lines of our approach in Section 4, we adopt the following strategy:

1. Choose a *finitely generated cone* $C_0 : \mathsf{cone}(\mathbf{1}, \alpha_1, \ldots, \alpha_k)$ of functions. In practice, we form the initial cone by choosing all monomials m of degree at most d and adding the polynomials $+m$ and $-m$ to the generators of C_0.
2. We refine the cone C_i at step i, starting from $i = 0$, to obtain a cone $C_{i+1} \subseteq C_i$. If $C_{i+1} = C_i$, we stop and use the generators of the cone to extract a mapping.
3. Unlike vector spaces, the iteration over cones need not necessarily converge even for finitely generated (polyhedral) cones. Therefore, we will use a heuristic "widening" operator that will force convergence in finitely many steps [6].

Let us assume that a cone $C : \mathsf{cone}(\mathbf{1}, \alpha_1, \ldots, \alpha_k)$ fails to be closed (with psd residues). Our goal is to derive a sub-cone $D \subseteq C$ that is finitely generated and satisfies the closure conditions:

$$D \subseteq \{f \in C \mid \mathcal{L}_F(f) = c_0 + \sum_i c_i\alpha_i + \rho,\ c_i \in \mathbb{R},\ \rho \text{ is psd}\}\,.$$

A *naive* strategy for finding D is to drop those generators (α_is) in the basis that fail the closure condition. However, this strategy fails to find interesting cones in practice.

[1]The problem is harder if trigonometric functions are involved. Therefore, we restrict our focus to polynomials.

5.1.2 Using Sum-Of-Squares Programming

Sum-Of-Squares relaxation is a well known technique that allows us to relax nonlinear programs involving polynomial inequalities into semi-definite programs. Originally discovered by Shor, the SOS relaxation has been applied widely in control theory and verification [18, 21].

Let $C : \mathsf{cone}(\mathbf{1}, \alpha_1, \ldots, \alpha_k)$ be a finitely generated cone. Any element of the cone can be written as $f(\vec{\lambda}, \vec{x}):\ \lambda_0 + \sum_{i=1}^{k} \lambda_i\alpha_i$, for multipliers $\vec{\lambda}:\ \lambda_0, \ldots, \lambda_k \geq 0$.

1. Consider the ansatz $f(\vec{\lambda}, \vec{x})$, with parameters $\vec{\lambda}$ representing the non-negative multipliers.
2. Compute its Lie derivative w.r.t $\mathcal{F}$, to obtain the polynomial $\mathcal{L}_F(f(\vec{\lambda}, \vec{x}))$.
3. Equate $\mathcal{L}_F(f)$ with the form $g = c_0 + \sum_{i=1}^{k} c_i\alpha_i + \rho$, wherein $c_0, \ldots, c_k$ are real-valued parameters (not necessarily non-negative) and ρ is an unknown generic polynomial template over parameters $d_1, \ldots, d_N$ that we require to be a posynomial.
4. We derive constraints by comparing monomial terms in $\mathcal{L}_F(f)$ and g. The positive semi-definiteness of ρ is encoded using the SOS relaxation. The resulting system of constraints has the following form:

   $$A\vec{\lambda} + \vec{b} = P\vec{c} + Q\vec{d},\ \vec{\lambda} \geq 0,\ Z(\vec{d}) \succeq 0\,.$$

   for matrices $A, P, Q, \vec{b}$ over reals, unknowns $\vec{\lambda}, \vec{c}, \vec{d}$ and matrix $Z(\vec{d})$ whose entries are linear expressions over $\vec{d}$. Z is constrained to be a positive semi-definite matrix.

Unfortunately, the set of values of $\vec{\lambda}$ for which the semi-definite program above is feasible need not form a finitely generated cone (the cone may have infinitely many generators). Therefore, we need to underapproximate the cone by extracting finitely many generators. This can be performed in many ways, including finding optimal solutions to the SDP for various randomly chosen values for the objective function.

5.1.3 Using Convex Polyhedral Cones

A weaker alternative to using SOS relaxation to characterize a finitely generated cone of positive semi-definite polynomials by starting from a finite set of known posynomials of bounded degree. Let $\text{POS} = \{p_1, \ldots, p_m\}$ be a finite set of posynomials. Then any conic combination of polynomials in POS is also a posynomial, yielding us a finitely generated cone of posynomials.

A polynomial all of whose monomials have variables with even powers of the form $\Pi_{i=1}^{n} x_i^{2r_i},\ r_i \in \mathbb{N}$, and all of whose coefficients are non-negative is a posynomial. Collecting all these monomials up to some degree and forming the cone generated by these monomials yields a polyhedral cone of posynomials.

Alternatively, we may derive a finite set by extracting finitely many feasible solutions to a SOS program that encodes that an unknown template polynomial is positive semi-definite.

Given a cone $C : \mathsf{cone}(\mathbf{1}, \alpha_1, \ldots, \alpha_k)$, we wish to refine the cone to obtain a new cone D that satisfies

$$D \subseteq \{f \in C \mid \mathcal{L}_F(f) = c_0 + \sum_i c_i\alpha_i + \rho,\ \rho \text{ is psd}\}\,.$$

Assuming a cone of posynomials generated by POS, we proceed as follows:

1. Create an ansatz $f(\vec{\lambda}, \vec{x}):\ \lambda_0 + \sum_i \lambda_i\alpha_i$.

2. Compute the Lie derivative: $\mathcal{L}_F(f)$. This will be a polynomial over $\vec{\lambda}, \vec{x}$.

3. We equate $\mathcal{L}_F(f)$ with a template polynomial g,

$$g = c_0 + \sum_i c_i \alpha_i + \sum_j \gamma_j p_j, \text{ where } \gamma_j \geq 0\,,$$

and each $p_j \in P$ is a known posynomial.

4. Finally, we obtain linear constraints of the form:

$$A\vec{\lambda} = P\vec{c} + Q\vec{\gamma},\ \vec{\gamma} \geq 0,\ \vec{\lambda} \geq 0\,.$$

5. Eliminating $\vec{c}, \vec{\gamma}$ yields linear inequality constraints over $\vec{\lambda}$ whose generators yield the new cone D.

The technique, as described above, requires a set POS of posynomials and uses polyhedral projection, which can be expensive in practice. The following example illustrates how this can be avoided in practice.

Example 5.4. *We recall the system in Example 5.3.*

$$\frac{dx}{dt} = x + 2xy,\ \frac{dy}{dt} = 1 + 3y - y^2$$

Consider the cone C generated by

$$\{\mathbf{1}, \alpha_1 :\ x^2, \alpha_2 :\ y^2,\ \alpha_3 :\ x, \alpha_4 :\ -y, \alpha_5 :\ x^2 y\}\,.$$

A generic polynomial inside the cone can be written as

$$f :\ \lambda_0 + \lambda_1 x^2 + \lambda_2 y^2 + \lambda_3 x + \lambda_4 y + \lambda_5 x^2 y\,.$$

Its Lie derivative is given by:

$$-\lambda_4 + \lambda_3 x + (2\lambda_2 - 3\lambda_4)y + (2\lambda_1 + \lambda_5)x^2 + 2\lambda_3 xy + (5\lambda_5 + 4\lambda_1)x^2 y + 3\lambda_5 x^2 y^2 - 2\lambda_2 y^3$$

To avoid an expensive elimination, we heuristically split the lie derivative into two types of terms: (a) terms that need to be linear combinations of α_is and (b) terms that belong to the posynomial. Our heuristic collects all the monomial terms that are part of each polynomial α_i. The corresponding terms in the Lie derivative along with their coefficients are constrained to be a linear combination of α_is. The remainder is constrained to be a posynomial.

In this example, the posynomial part is $2\lambda_3 xy + 3\lambda_5 x^2 y^2 - 2\lambda_2 y^3$. We note that this is a posynomial if

$$\lambda_3 = 0, \lambda_2 = 0, \lambda_5 \geq 0\,.$$

We now add constraints that force the remainder of the Lie derivative to be a linear combination of the α_is:

$$-\lambda_4 + \lambda_3 x + (2\lambda_2 - 3\lambda_4)y + (2\lambda_1 + \lambda_5)x^2 + (5\lambda_5 + 4\lambda_1)x^2 y = c_0 + c_1 x^2 + c_2 y^2 + c_3 x + c_4 y + c_5$$

This yields linear equality constraints with $\vec{\lambda}$ and $\vec{c}$. The $\vec{c}$ variables can be eliminated by Gaussian elimination. In this instance, the resulting constraint after elimination of $\vec{c}$ variables is true. Combining, the overall constraint is $\lambda_3 = 0,\ \lambda_2 = 0,\ \vec{\lambda} \geq 0$. This yields the cone generated by $\{\mathbf{1}, \alpha_1 :\ x^2,\ \alpha_4 :\ -y,\ \alpha_5 :\ x^2 y\}$, as a result of the refinement operation.

5.1.4 Ensuring Termination

The iterative process of refinement starts from some initial cone C_0 that contains all monomial terms and their negations upto some degree cutoffs. At each step, we perform a refinement to compute a refined cone C_{i+1} from the current cone C_i. The refinement can be performed either by formulating a semidefinite program and extracting finitely many feasible solutions or by using polyhedral cones generated by a finite set of posynomials (eg., all monomials which are squares with non-negative coefficients).

In both cases, the refinement iteration is not guaranteed to converge in finitely many steps. To force convergence, we use a *widening* operator [6].

The widening operator ∇ applied to two successive iterates $C : C_i \nabla C_{i+1}$ produces a new finitely generated cone C satisfies $C \subseteq C_i,\ C \subseteq C_{i+1}$ and furthermore, for any sequence $C_0 \supseteq C_1 \supseteq C_2 \supseteq \cdots$, the *widened sequence*

$$C_0,\ C_1,\ D_1 : C_0 \nabla C_1,\ D_2 : D_1 \nabla C_2,\ \ldots,\ D_i : D_{i-1} \nabla C_i,\ \ldots$$

is always guaranteed to converge in finitely many steps.

Def. 5.4 (Widening Over (Dual) Cones). *Given two finitely generated cones C_1, C_2 such that $C_2 \subseteq C_1$, the standard widening $C_1 \nabla C_2$ is defined as* $\mathsf{cone}(\{f \in \mathsf{Generators}(C_1) \mid f \in C_2\})$.

In other words, the standard widening drops those generators in C_1 that do not belong to C_2. It is easy to see why a widened iteration using standard widening terminates in finitely many steps. At each widening step $D_{i+1} : D_i \nabla C_{i+1}$, we drop a generator from the cone D_i that do not belong to C_{i+1}. Since there are finitely many generators to begin with in D_1, the widened iteration terminates in finitely many steps.

5.1.5 Abstractions over Domains

Thus far, our techniques have considered the dynamics of the system $\mathcal{S} : \langle X_0, \mathcal{F}, X_I \rangle$ being abstracted, without using knowledge of its invariant set (or domain) X_I. Often, the abstraction being sought is over some domain $X_I \subseteq \mathbb{R}^n$. Our techniques for finding inequality abstractions can be readily modified to treat invariant domains that need not necessarily be bounded. We now briefly present the generalizations to the definitions and the iterative refinement technique to operate over domains X_I.

Def. 5.5 (Closure over Domain). *A finite set of functions $S : \{\alpha_1, \ldots, \alpha_k\}$ is closed under Lie derivatives with a* positive semidefinite *residue* over a domain X_I *iff for all $\alpha_i \in S$, the Lie derivative of α_i is of the form: $\forall\, \vec{x} \in X_I,\ \mathcal{L}_F(\alpha_i)(\vec{x}) = b_i \mathbf{1} + \sum_j a_{ij}\alpha_j + \rho_i(\vec{x})$, wherein, $a_{ij}, b_i \in \mathbb{R}$, and ρ_i is psd.*

In contrast to Def. 5.2, the Lie derivative of α_i equals a linear combination with positive residue over the domain X_I, as opposed to everywhere. The iterative refinement techniques incorporate the generalized definition. The refinement of the cone C has to guarantee that the refined cone D satisfies

$$D \subseteq \{\, f \in C \mid (\forall\, \vec{x} \in X_I),\ \mathcal{L}_F(f) = \sum_j a_j \alpha_j + \rho,\ \rho(\vec{x}) \ \geq\ 0\}\,.$$

The iteration using SOS relaxation is easily modified under the assumption that X_I is represented as the feasible region of a system of polynomial inequalities. The approach using a finitely generated cone of posynomials can be modified by allowing the cone to include monomials that are known to be positive over X_I (as opposed to everywhere).

If the domain X_I is bounded, our technique defaults to a *truncated* Carleman linearization involving all monomials upto a degree cutoff. Soundness of the truncation is ensured by obtaining

interval bounds on the *residues* for each monomial. This approach can form the basis for a hybridization abstraction wherein accuracy can be improved by repeatedly subdividing the domain X_I. [7].

5.1.6 Application to Hybrid Systems

Thus far, we have presented our techniques for abstracting continuous system. The continuous system abstraction for the dynamics corresponding to a mode can be directly employed to compute reachable sets for a given initial condition. This can be used repeatedly as a primitive inside hybrid systems analysis tools.

It is possible to extend the notion of change of bases transformations to discrete transitions using closure under the weakest precondition operator as opposed to Lie derivatives. As a result, we can integrate the conditions for closure under Lie derivatives for continuous systems and the closure under pre-conditions for discrete transitions to yield an extension of our theories for hybrid systems. The full technique for hybrid systems will be described in an extended version of this paper.

6. EXPERIMENTAL EVALUATION

In this section, we describe a prototype implementation of some of the ideas presented thus far. We also report on the experimental evaluation of some benchmark systems using our implementation. Our implementation, the benchmark systems and the outputs are available on-line for review [2].

6.1 Implementation

We have implemented the search for a change of bases transformation using vector space iteration as well as iteration over polyhedral cones using monomials of the form $\Pi x_i^{2r_i}$ as the generator of the cone of posynomials. Our implementation (in OCaml) reads in the description of a continuous system and a degree bound for the abstraction search. It then performs a vector space iteration followed by a polyhedral iteration. The polyhedral iteration uses the *Parma Polyhedral Library* [4]. However, polyhedral cone operations such as computing the generators is worst-case exponential in the number of variables. We implement an optimized version of the iteration over polyhedral cones that separates the linear equality and inequality constraints. The equality constraints are maintained in a triangular form so that we may minimize the number of variables involved in the inequalities. This enables us to handle systems with upwards of 5000 basis polynomials. To improve the quality of the result, we use a *delayed widening* strategy that starts applying the widening operator only after there are no more linear equality constraints to be added.

6.2 Experimental Evaluation

In this section, we describe the results of our technique on some benchmarks. Table 1 summarizes the results of our analysis and the performance of our implementation over various benchmarks assuming various degree bounds. We discuss some of these benchmarks below, briefly. The details of the invariants discovered are part of our release and will be discussed in an extended version.

Two Spring Mass System With Friction We model a mechanical system with two masses that are connected to each other and to a fixed end using springs with constants k_1, k_2 and masses m_1, m_2 adjusted so that $\frac{k_1}{m_1} = \frac{k_2}{m_2} = k$. Furthermore, we assume that $m_1 = 5m_2$. The variables $\vec{x} : (x_1, x_2, v_1, v_2, k)$ representing the displacements, velocities and the spring constant. We assume that the drag due to friction is proportional to the velocity. The dynamics are described by the vector field

$$(v_1, v_2, -kx_1 - \frac{k}{5}(x_1 - x_2) - .1v_1, k(x_1 - x_2) - .1v_2, 0)\,.$$

The search for degree bound 3 yields the transformation below:

$$\alpha_0 :\ k,\ \alpha_1 :\ -10v_1v_2 + 5v_1^2 + x_2^2k - 12x_1x_2k + 11x_1^2k\,.$$

The Lie derivative of α_1 is v_2^2, a posynomial. The application of our technique to the conservative system without friction also discovers non-trivial transformations in the form of conserved quantities and inequality invariants.

Collision Avoidance We consider the algebraic abstraction of the collision avoidance system analyzed recently by Platzer et al. [20] and earlier by Tomlin et al. [30]. The two airplane collision avoidance system consists of the variables (x_1, x_2) denoting the position of the first aircraft, (y_1, y_2) for the second aircraft, (d_1, d_2) representing the velocity vector for aircraft 1 and (e_1, e_2) for aircraft 2. ω, θ abstract the trigonometric terms. In addition, the parameters a, b, r_1, r_2 are also represented as system variables. The dynamics are modeled by the following differential equations:

$$\begin{array}{cccc} x_1' = d_1 & x_2' = d_2 & d_1' = -\omega d_2 & d_2' = \omega d_1 \\ y_1' = e_1 & y_2' = e_2 & e_1' = -\theta e_2 & e_2' = \theta e_1 \\ a' = 0 & b' = 0 & r_1' = 0 & r_2' = 0 \end{array}$$

A search for transformations of degree 2 yields a closed vector space with 27 basis functions within 0.2 seconds. The basis functions include a, b, r_1, r_2 and all degree two terms involving these. Removing these from the basis, gives us 14 basis functions that yield a transformation to a 14 dimensional affine ODE.

Biochemical reaction network: Finally, we analyze a biochemical reaction network benchmark from Dang et al. [7]. The ODE along with the values are parameters in our model coincide with those used by Dang et al. The ODE consists of 12 variables. Our search for degree bound ≤ 2 discovers a transformation generated by three basis functions (in roughly .3 seconds). This leads to two differential equalities and one inequality. Note that our analysis does not assume any information about the invariant region. Searching for an abstraction over the invariant region may help us derive a finer abstraction of this ODE along the lines of Dang et al. [7]. Table 1 reports on the results of two versions of the system: with numerical values for the various parameters involved and by encoding these

Table 1: Summary of experimental results over benchmark systems. Legend: Var: number of variables, deg: degree bound, T: time taken (seconds), iter: number of iterations, eq: number of equalities, ineq: number of inequalities, †: transformations purely involving the system parameters were removed.

Sys	Vars	deg	T	iter	eq	ineq
proj-drag	4	3	.5	15	5	2
spr-mass2	5	3	.1	10	$1^\dagger$	1
spr-mass2	5	4	.6	15	$1^\dagger$	4
spr-mass2	5	7	61.7	28	$1^\dagger$	5
coll-avoid-2	14	2	.2	7	$14^\dagger$	0
coll-avoid-2	14	4	144	23	$160^\dagger$	0
bio-net	12	3	4.9	18	1	2
bio-net	12	4	112	19	1	2
bio-net-par	26	2	1.7	4	$16^\dagger$	1
bio-net-par	26	3	181	5	~ 100	~ 14

[2]Cf. `http://www.cs.colorado.edu/~srirams/code/nlsys-release.tar.gz`.

parameters as extra variables whose derivatives are zero. The ability to treat a 26 dimensional system (reasoning over vector-spaces and cones of dimension ~ 3600) is quite a promising result for our approach.

7. CONCLUSION

Thus far, we have presented some techniques for discovering linear abstractions through a change of bases transformation and an evaluation of our techniques using a prototype implementation. In the future, we wish to implement our techniques to search for abstraction over domains. The extension of our techniques to handle non-linear switched and hybrid systems is ongoing.

8. REFERENCES

[1] Rajeev Alur, Thao Dang, and Franjo Ivančić. Counter-example guided predicate abstraction of hybrid systems. In *TACAS*, volume 2619 of *LNCS*, pages 208–223. Springer, 2003.

[2] Rajeev Alur, Thomas A. Henzinger, G. Lafferriere, and George Pappas. Discrete abstractions of hybrid systems. *Proc. of IEEE*, 88(7):971–984, 2000.

[3] Eugene Asarin, Thao Dang, and Antoine Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43:451—476, 2007.

[4] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Quaderno 457, Dipartimento di Matematica, Università di Parma, Italy, 2006.

[5] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[6] Patrick Cousot and Rhadia Cousot. Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *ACM Principles of Programming Languages*, pages 238–252, 1977.

[7] Thao Dang, Oded Maler, and Romain Testylier. Accurate hybridization of nonlinear systems. In *HSCC '10*, pages 11–20. ACM, 2010.

[8] Thao Dang and David Salinas. Image computation for polynomial dynamical systems using the bernstein expansion. In *Computer Aided Verification*, volume 5643 of *Lecture Notes in Computer Science*, pages 219–232. Springer, 2009.

[9] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *HSCC*, volume 2289 of *LNCS*, pages 258–273. Springer, 2005.

[10] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *HSCC*, volume 3414 of *LNCS*, pages 291–305. Springer, 2005.

[11] Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250 – 262, 2010.

[12] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43:540–554, 1998.

[13] K. Kowalski and W-H. Steeb. *Non-Linear Dynamical Systems and Carleman Linearization*. World Scientific, 1991.

[14] Nadir Matringe, Arnoldo Viera Moura, and Rachid Rebiha. Morphisms for non-trivial non-linear invariant generation for algebraic hybrid systems. In *HSCC*, volume 5469 of *LNCS*, pages 445–449, 2009.

[15] James D. Meiss. *Differential Dynamical Systems*. SIAM publishers, 2007.

[16] Venkatesh Mysore, Carla Piazza, and Bud Mishra. Algorithmic algebraic model checking II: Decidability of semi-algebraic model checking and its applications to systems biology. In *ATVA*, volume 3707 of *LNCS*, pages 217–233. Springer, 2005.

[17] Meeko Oishi, Ian Mitchell, Alexandre M. Bayen, and Claire J. Tomlin. Invariance-preserving abstractions of hybrid systems: Application to user interface design. *IEEE Trans. on Control Systems Technology*, 16(2), Mar 2008.

[18] Pablo A Parillo. Semidefinite programming relaxation for semialgebraic problems. *Mathematical Programming Ser. B*, 96(2):293–320, 2003.

[19] André Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reasoning*, 41(2):143–189, 2008.

[20] André Platzer and Edmund Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in Systems Design*, 35(1):98–120, 2009.

[21] Stephen Prajna and Ali Jadbabaie. Safety verification using barrier certificates. In *HSCC*, volume 2993 of *LNCS*, pages 477–492. Springer, 2004.

[22] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *HSCC*, volume 3414 of *LNCS*, pages 573–589. Springer, 2005.

[23] Enric Rodriguez-Carbonell and Ashish Tiwari. Generating polynomial invariants for hybrid systems. In *HSCC*, volume 3414 of *LNCS*, pages 590–605. Springer, 2005.

[24] Sriram Sankaranarayanan. Automatic invariant generation for hybrid systems using ideal fixed points. In *Hybrid Systems: Computation and Control*, pages 211–230. ACM Press, 2010.

[25] Sriram Sankaranarayanan, Thao Dang, and Franjo Ivančić. Symbolic model checking of hybrid systems using template polyhedra. In *TACAS*, volume 4963 of *LNCS*, pages 188–202. Springer, 2008.

[26] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Constructing invariants for hybrid systems. *Formal Methods in System Design*, 32(1):25–55, 2008.

[27] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Fixed point iteration for computing the time-elapse operator. In *HSCC*, LNCS. Springer, 2006.

[28] Ashish Tiwari. Abstractions for hybrid systems. *Formal Methods in Systems Design*, 32:57–83, 2008.

[29] Ashish Tiwari and Gaurav Khanna. Non-linear systems: Approximating reach sets. In *HSCC*, volume 2993 of *LNCS*, pages 477–492. Springer, 2004.

[30] Claire J. Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Trans. on Aut. Control*, 43(4):509–521, April 1998.

[31] Pravin Varaiya. Reach set computation using optimal control. In *Proc. KIT Workshop on Hybrid Systems*, 1998.

[32] S. Vassilyev and S. Ul'yanov. Preservation of stability of dynamical systems under homomorphisms. *Differential Equations*, 45:1709–1720, 2009.