Cone-Based Abstract Interpretation for Nonlinear Positive Invariant Synthesis

Guillaume O. Berger
UC Louvain
Belgium
guillaume.berger@uclouvain.be

Masoumeh Ghanbarpour
University of Colorado Boulder
USA
masoumeh.ghanbarpour@colorado.edu

Sriram Sankaranarayanan
University of Colorado Boulder
USA
first.lastname@colorado.edu

ABSTRACT

We present an abstract interpretation approach for synthesizing nonlinear (semi-algebraic) positive invariants for systems of polynomial ordinary differential equations (ODEs) and switched systems. The key behind our approach is to connect the system under study to a positive nonlinear system through a “change of variables”. The positive invariance of the first orthant \((\mathbb{R}^n_+)\) for a positive system guarantees, in turn, that the functions involved in the change of variables define a positive invariant for the original system. The challenge lies in discovering such functions for a given system. To this end, we characterize positive invariants as fixed points under an operator that is defined using the Lie derivative. Next, we use abstract-interpretation approaches to systematically compute this fixed point. Whereas abstract interpretation has been applied to the static analysis of programs, and invariant synthesis for hybrid systems to a limited extent, we show how these approaches can compute fixed points over cones generated by polynomials using sum-of-squares optimization and its relaxations. Our approach is shown to be promising over a set of small but hard-to-analyze nonlinear models, wherein it is able to generate positive invariants to place useful bounds on their reachable sets.

ACM Reference Format:

1 INTRODUCTION

In this paper, we provide solutions to the problem of synthesizing semi-algebraic positive invariants for ordinary differential equations (ODEs) whose right-hand sides are defined by polynomials over the system variables. We show how our approach extends to switched systems that contain multiple modes, each described by polynomial ODEs along with transitions between these modes, governed by semi-algebraic guard conditions. Positive invariants of ODEs and switched systems help us prove bounds on the sets of states that can be reached over an infinite time horizon, thus proving that a certain set of unsafe states will never be reached. The problem of automatically synthesizing such positive invariants has been of great interest for verification. Many approaches have been studied for this problem that include constructing finite abstractions \([4, 6, 50]\), dynamic programming-based approaches \([52]\), approaches based on solving nonlinear constraints in order to construct barrier functions and their generalizations \([7, 8, 36, 37]\).

The key of our approach is to relate an ODE \(\dot{x} = f(x)\) to a positive system of the form \(\dot{\omega} = -\lambda \omega + F(z, \omega)\), wherein \(\omega \in \mathbb{R}^m\) is connected to the state \(x \in \mathbb{R}^n\) through a polynomial map \((\omega_1, \ldots, \omega_m) = (g_1(x), \ldots, g_m(x))\) such that \(\lambda\) is a positive scalar quantity, \(g_1, \ldots, g_m\) are polynomials and \(F\) is a nonlinear function with the property that for all \(z \in \mathbb{R}\) and \(\omega \in \mathbb{R}^m_+\), we have \(F(z, \omega) \in \mathbb{R}^m_+\). Here, \(z\) is taken to be an external input. In other words, we show that our original vector field is “l-related” to that of a positive system. Positive systems have a key property that if \(\omega(t) \in \mathbb{R}^m_+\) then \(\omega(\tau) \in \mathbb{R}^m_+\) for all times \(\tau\) over which the trajectory is defined. We can use this property to infer that \(g_1 \geq 0, \ldots, g_m \geq 0\) is a positive invariant set for the original ODE. There are many difficulties to this approach, including: (a) choosing the dimensions of the unknown state space \(\omega\); (b) discovering the function \(F\); and (c) synthesizing the functions \(g_1, \ldots, g_m\). We propose a method to synthesize the polynomials \(g_1, \ldots, g_m\) that will implicitly define the positive system, and thus, our positive invariant set. We require the user to fix the constant \(\lambda\), an upper limit on the number \(m\), and a degree limit on the polynomials \(g_1, \ldots, g_m\). Our approach either converges with positive invariants (the function \(F\) gets defined implicitly by the generators upon convergence) or fails with a trivial answer.

To discover positive invariants, we iterate over finitely generated polynomial cones that are defined by a basis set of polynomials. We first derive a “refinement operator” of the cone from the given ODE. We derive a closure condition based on this operator: i.e., for a given finitely generated cone of polynomials \(C\), our closure condition requires that every polynomial in \(C\) is mapped by the refinement operator to a related cone \(\bar{C}\). If \(C\) satisfies the closure condition, then we prove that its generators form the required functions \(g_1, \ldots, g_m\) that will map the original system dynamics to a positive nonlinear system, as described above. Therefore, \(g_1 \geq 0, \ldots, g_m \geq 0\) will define a positive invariant. Having defined the notion of positive invariants in terms of closure of an operator, the challenge now lies in computing cones that are closed in this manner. To do so, we employ an approach based on abstract interpretation. Abstract interpretation was originally proposed by Cousot and Cousot in 1977 as an approach for establishing invariants of programs \([14, 15]\). It has been very successful in proving properties of large safety-critical software systems used in avionics \([3, 16]\). However, applying abstract interpretation to our framework is challenging since we
will observe that the closure condition used leads us from finitely generated polynomial cones to possibly infinitely generated cones. We show how a finite set of generators can be selected and maintained using a projection operator defined in this paper. Finally, we show that a “standard” widening operator commonly used in abstract interpretation can force termination in finitely many steps.

We provide an empirical evaluation of our approach over a set of nonlinear polynomial ODE benchmarks and switched systems, some of which are taken from the related work. We demonstrate that our approach can yield useful positive invariant sets. Even though we employ floating-point numbers in the calculation of these invariants, they have been successfully verified using the exact-arithmetic-based nonlinear theorem prover Z3. We also compare our approach with well-established approaches for synthesizing barrier functions based on sum-of-squares programming [36].

1.1 Related Work

There have been numerous techniques to directly synthesize positive invariants for ODEs and hybrid systems. This includes barrier function synthesis [36], approaches based on constraint solving by assuming a template form of the nonlinear invariant [20, 26, 35, 46, 48, 51] and abstract-interpretation approaches based on forward propagation and widening [21, 42], or in other cases through forward propagation and extrapolation [22]. Additionally, theorem provers such as Keymaera-X support proving properties of practical hybrid systems using positive invariant synthesis to support human reasoning [19, 32, 33, 35]. Due to space limitations, we do not expand on these approaches, noting that some of the recent textbooks cover these approaches and the theory behind them [5, 29, 34, 39].

The closest related works to our approach include the notion of comparison systems proposed by Sogokon et al. [46], the notion of a change-of-basis transformation proposed by Sankaranarayanan [40, 41], and abstract-interpretation-based iteration over polyhedral cones for linear systems first proposed by Sankaranarayanan et al. [42]. Sogokon et al. [46] propose the notion of vector barrier function which is a vector of functions that relates the flow of a nonlinear ODE to that of a positive linear system of the form \( \dot{x} = Ax + r(t) \), where \( A \) is a constant Metzler (aka. essentially nonnegative) matrix and \( r(t) \geq 0 \). Their approach synthesizes the polynomials \( g_1, \ldots, g_m \) by (a) assuming that the matrix \( A \) is given by the user and (b) the degree limits of the polynomials are specified. Our approach extends this concept to matrices \( \Lambda \) whose off-diagonal entries are positive definite polynomials and significantly does not require the user to provide us these matrices. However, the computational complexity of our approach is significant and we have to rely on heuristics to select generators from an infinitely generated cone. Nevertheless, we show success on small but interesting nonlinear systems. Sankaranarayanan [40, 41] proposes a similar idea of a change-of-basis transformation from a given nonlinear system to an autonomous linear system and uses an iterative abstract-interpretation-based procedure similar to what is being proposed here. However, the connection to an autonomous system essentially requires the original nonlinear system to be integrable, or in other words, have equality invariants (though these may not be necessarily be polynomial). Finally, the idea of abstract interpretation on cones was developed in Sankaranarayanan et al. [42]. But this work was restricted to linear systems where the iterations need to be over polyhedral cones. This paper goes much further and considers nonlinear systems as well as non-polyhedral cones generated by polynomials.

The synthesis of positive invariants has received much attention in the past. Taly and Tiwari provide a proof system that is sound and relatively complete for a single polynomial inequality using higher-order derivatives [49]. Liu et al. extend this to a powerful relatively complete method for synthesizing semi-algebraic positive invariants for polynomial hybrid systems [26]. Their approach fixes the form (aka. template) of the desired invariant and uses a condition based on higher-degree Lie derivative. This is essentially a refinement of the barrier set condition that states that the first non-zero higher Lie derivative must be positive at the boundaries of the invariant set. By cleverly connecting their approach to the descending chain condition for ideals on a polynomial ring, they are able to provide a relative completeness guarantee. The proof system of Liu et al. is relatively complete unlike ours which is weaker than that of Liu et al. because (a) our approach is limited to (closed) basic semi-algebraic sets and (b) we use a weaker positive-invariance condition based on relating to a positive nonlinear system. On the other hand, our approach does not use expensive quantifier elimination over semi-algebraic sets: each iteration of our approach uses sum-of-squares optimization. Ghorbal et al. provide a hierarchy of proof rules for computing semi-algebraic invariants that places the work of Liu et al. in context at the apex of a series of increasingly more complex proof rules. They also provide interesting comparisons on the types of flows that each rule can handle [20]. Our approach has two major differences: (a) we use the connection with positive systems to avoid reasoning about the boundaries of the invariant sets or explicitly compute higher-order derivatives; and (b) we work on basic semi-algebraic sets defined as intersections of polynomial inequalities. Our approach does not extend to general semi-algebraic sets which are unions of these basic sets.

2 PROBLEM STATEMENT

Notation. Let \( \mathbb{N} \) be the set of natural numbers, and \( \mathbb{R}_+ \) be the set of nonnegative real numbers. Given \( n \in \mathbb{N} \), let \( [n] = \{1, \ldots, n\} \). We will use bold-face to denote vectors \( x, y, z \in \mathbb{R}^n \) and capital letters to describe matrices \( A, B, C \in \mathbb{R}^{m \times n} \). For a vector \( x \in \mathbb{R}^n \), the \( i \)-th component for \( i \in [n] \) is denoted as \( x_i \). Let \( \mathbb{R}[x] \) be the ring of polynomials over variables \( x = (x_1, \ldots, x_n) \). Given a function \( f : A \rightarrow B \), let \( \text{dom}(f) = A \).

2.1 Polynomial Systems

Consider a continuous-time dynamical system ODE(\( f \)) : \( x = f(x) \), wherein \( f : \mathbb{R}^n \rightarrow \mathbb{R}^n \) is locally Lipschitz continuous. A trajectory of ODE(\( f \)) is defined as a differentiable function \( \phi : [0, T] \rightarrow \mathbb{R}^n \) satisfying that for all \( t \in [0, T) \), \( \dot{\phi}(t) = f(\phi(t)) \).

Remark 2.1. The trajectory may exist for all time, i.e, \( T = \infty \). However, since our focus is on safety, we simply assume that the trajectory exists at least until some time \( T \geq 0 \), and place no bound on \( T \). Also, since we assume local Lipschitz continuity, the trajectory must exist and be unique [28].
Let $I \subseteq \mathbb{R}^n$ be an initial set. A system $\Sigma := \langle f, I \rangle$ consists of a dynamical system ODE($f$) and an initial set $I$. The system $\Sigma$ is said to be safe if no trajectory $\phi$ of ODE($f$) with $\phi(0) \in I$ reaches some given unsafe set $S_{\text{unsafe}}$. We wish to prove the safety of polynomial systems, wherein (a) the dynamics are described by polynomials and (b) the initial set is a basic semi-algebraic set, described by polynomial inequalities, as follows.

**Definition 2.1 (Basic Semi-Algebraic Sets).** A basic semi-algebraic set (BSA set) is a set described by a finite set of polynomial inequalities of the form $g_i(x) \geq 0$. More formally, for a set of polynomials $G = \{g_1, \ldots, g_m\} \subseteq \mathbb{R}[x]$, we denote the BSA generated by $G$ as

$$S(G) = \{x \in \mathbb{R}^n : g_i(x) \geq 0, i = 1, \ldots, m\}.$$ 

**Example 2.2.** Fig. 1 (left) depicts the BSA set $S(G)$ wherein $G = \{1 + \frac{1}{2}x_1^2 + \frac{1}{2}x_1x_2 - \frac{1}{2}x_2^2, 1 - x_1^2 - \frac{1}{2}x_2^2\}$.

**Definition 2.3 (Polynomial System).** The system $(f, I)$ is a polynomial system if (a) the vector field $f$ is defined by polynomials: $f(x) = (f_1(x), \ldots, f_m(x))$ with $f_i \in \mathbb{R}[x]$; (b) $I$ is a BSA set $S(G_{\text{init}})$ for a finite set of polynomials $G_{\text{init}} \subseteq \mathbb{R}[x]$.

**Example 2.4.** Consider the following Vanderpol oscillator:

$$\text{ODE}(f) : \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{1}{2}x_2 - x_1 - \frac{1}{2}x_1^3 \end{bmatrix},$$
with initial set $I = \{x_1^2 + x_2^2 \leq \frac{1}{4}\}$, i.e., $G_{\text{init}} = \{\frac{1}{4} - x_1^2 - x_2^2\}$. The tuple $(f, I)$ is a polynomial system. The vector field and sample trajectories of ODE($f$) are presented in Fig. 1 (right).

### 2.2 Constrained and Switched Polynomial Systems

Next, we define constrained and switched systems.

**Definition 2.5 (Constrained Polynomial System).** A constrained polynomial system is a triple $(f, I, D)$ wherein the dynamics is given by ODE($f$) for $f(x) = (f_1(x), \ldots, f_m(x))$ with $f_i \in \mathbb{R}[x]$, $D \subseteq \mathbb{R}^n$ is the domain of evolution (or constraint) that restricts the state space of the system and $I \subseteq D$ is the initial set. Furthermore, $I$ and $D$ are nonempty BSA sets.

A trajectory of the constrained dynamics ODE($f, D$) is a trajectory $\phi : [0, T) \to \mathbb{R}^n$ of ODE($f$) such that for all time $t \in [0, T)$, the state is in $D$, i.e., $\phi(t) \in D$. Consequently, a constrained system is not allowed to reach a state outside of its domain of evolution. The constrained system $\langle f, I, D \rangle$ is said to be safe if no trajectory $\phi$ of ODE($f, D$) with $\phi(0) \in I$ reaches some given unsafe set $S_{\text{unsafe}}$.

Next, we define a switched system through a combination of a finite number of constrained systems connected by transitions.

**Definition 2.6 (Switched Polynomial System).** A switched polynomial system is defined by a set of modes $Q$ wherein each mode $q \in Q$ is associated with a constrained polynomial system $(f_q, I_q, D_q)$ along with a finite set of transitions $T$. Each transition $\tau \in T$ is a triple $(a_\tau, b_\tau, G_\tau)$ with pre- and post-modes $a_\tau \in Q$ and $b_\tau \in Q$ respectively and a guard set $G_\tau \subseteq D_{a_\tau} \cap D_{b_\tau}$. We assume that $G_\tau$ is a basic semi-algebraic set given by $S(G_{\text{guard}, \tau})$ for a finite set of polynomials $G_{\text{guard}, \tau}$.

A trajectory of the switched dynamics ODE($Q, f_q \in Q, D_q \in Q, T$) is specified by (a) a finite, increasing sequence of times $0 = t_0 < t_1 < \ldots < t_k < t_{k+1}$, (b) a sequence of modes $q_0, q_1, \ldots, q_k \in Q$, (c) a sequence of transitions $\tau_1, \tau_2, \ldots, \tau_k \in T$, and (d) a function $\phi : [0, t_{k+1}) \to \mathbb{R}^n$ such that the following conditions hold:

1. For each $i \in [k]$, $\tau_i$ is a transition from $q_{i-1} \to q_i$ and the guard condition $\phi(t_i) \in G_{\tau_i}$ holds.
2. For each $i \in [k] \cup \{0\}$, the function $\phi_i : [0, t_{i+1} - t_i) \to \mathbb{R}^n$ defined by $\phi_i(t) = \phi(t + t_i)$ is a trajectory of ODE($f_{q_i}, D_{q_i}$).

The switched system $(f_q \in Q, I \in Q, D_q \in Q, T)$ is said to be safe if no trajectory $\langle \phi, q_0, k-1, t_0, k+1 \rangle$ of ODE($Q, f_q \in Q, D_q \in Q, T$) with $\phi(0) \in I_{q_0}$ reaches some unsafe set $S_{\text{unsafe}}$, i.e., satisfies $\phi(t) \in S_{\text{unsafe}, q_i}$, for some $t \in [t_i, t_{i+1})$.

### 2.3 Forward Invariant Sets and Safety

An invariant of a dynamical system is a property that is preserved along the trajectories of the system. They can be used to certify safety, for instance, if the property holds for all initial conditions and for no unsafe states.

**Definition 2.7.** A set $P \subseteq \mathbb{R}^n$ is forward invariant for ODE($f$) if for all trajectories $\phi$ of ODE($f$) with $\phi(0) \in P$, and for all $t \in \text{dom}(\phi)$, the state at time $t$ belongs to $P$: $\phi(t) \in P$.

**Proposition 2.8.** If $P$ is a forward invariant for ODE($f$), $I \subseteq P$ and $P \cap S_{\text{unsafe}} = \emptyset$, then the system $(f, I)$ is safe.

Using Prop. 2.8 we prove safety of a given (constrained and switched) system as follows: Search for a forward invariant set $P$ that includes the initial set $I$ that excludes $S$; i.e., $P \cap S_{\text{unsafe}} = \emptyset$. If such a forward invariant can be found, we conclude that the safety property holds for the system.

In this paper, we seek to compute safe invariants for polynomial systems in the form of basic semi-algebraic sets, that is, sets described by polynomial inequalities. We remind below some background of polynomial inequalities. Extension of forward invariance to constrained and switched systems will be discussed in Sec. 4.

### 2.4 Polynomial Inequalities

Note that the polynomial set $G = \{g_1, \ldots, g_m\}$ defining $S(G)$ is not unique. In fact, for any $a_1, a_2 \in \mathbb{R}_+$ and $i_1, i_2 \in [m]$, it holds...
that $\alpha_1 g_1 + \alpha_2 g_2$ and $g_1, g_2$ are also nonnegative on $S(G)$. This motivates the following definitions.

**Definition 2.9 (Cone).** A set $K \subseteq \mathbb{R}[x]$ is a cone if for all $\alpha_1, \alpha_2 \in \mathbb{R}_+$ and $g_1, g_2 \in K$, it holds that $\alpha_1 g_1 + \alpha_2 g_2 \in K$. Given a set $B \subseteq \mathbb{R}[x]$, we define the conic hull of $B$ as

$$\text{ch}(B) = \left\{ \sum_{i=1}^{m} \alpha_i g_i : m \in \mathbb{N}, g_i \in B, \alpha_i \in \mathbb{R}_+ \right\}.$$ 

A cone $K \subseteq \mathbb{R}[x]$ is called finitely generated if there is a finite set $B = \{g_1, \ldots, g_m\} \subseteq \mathbb{R}[x]$ such that $K = \text{ch}(B)$.

**Definition 2.10 (Product).** Given two sets $G_1, G_2 \subseteq \mathbb{R}[x]$, their product is defined by

$$G_1 \cdot G_2 = \{g_1 g_2 : g_1 \in G_1, g_2 \in G_2\}.$$ 

Given $G \subseteq \mathbb{R}[x]$, we define $G^0 = \{1\}$, and for $\ell \in \mathbb{N}_{>0}$,

$$G^\ell = G \cdot G \cdot \cdots \cdot G,$$ 

and $G^{\infty} = \bigcup_{\ell \geq 0} G^\ell$.

We are now able to define a set of polynomials that are nonnegative on $S(G)$:

**Proposition 2.11.** Let $K \subseteq \mathbb{R}[x]$ be a cone containing only nonnegative polynomials (i.e., $h(x) \geq 0$ for all $h \in K, x \in \mathbb{R}^n$), and let $\ell \in \mathbb{N}$. Then, the set $\text{ch}(K \cdot G^{\infty})$ contains only polynomials nonnegative on $S(G)$ (i.e., $g(x) \geq 0$ for all $g \in \text{ch}(K \cdot G^{\infty}), x \in S(G)$).

**Example 2.12.** Consider a set $G = \{g_1, g_2, g_3\}$ of polynomials in $\mathbb{R}[x]$ and $K = \mathbb{R}_+$ the set of all nonnegative real numbers. The set $\text{ch}(K \cdot G^{\infty})$ contains all polynomials of the form $\lambda_0 + \lambda_1 g_1 + \lambda_2 g_2 + \lambda_3 g_2 g_3 + \lambda_4 g_1 g_2 + \lambda_5 g_1 g_3 + \lambda_6 g_3^2 + \lambda_7 g_2^2 + \lambda_8 g_2^3$ for $\lambda_0, \ldots, \lambda_8 \in \mathbb{R}_+$.

An example of cone $K$ of nonnegative polynomials is the set of sum-of-squares (SOS) polynomials.

**Definition 2.13 (Sum-of-Squares).** A polynomial $h \in \mathbb{R}[x]$ is a sum-of-squares if $h = \sum_{m=1}^{M} p_i^2$ for $m \in \mathbb{N}$ and $p_i \in \mathbb{R}[x]$. The set of SOS polynomials is denoted by $\text{SOS}[x]$.

**Remark 2.2.** Not all nonnegative polynomials are SOS (e.g., the so-called “Motzkin Polynomial”). However, whereas verifying that a polynomial is nonnegative is NP-hard, SOS offers a practically efficient way to certify positivity of polynomials [31, 45].

**Remark 2.3.** Note that Prop. 2.11 provides a sufficient but not necessary condition for characterizing a set of positive polynomials over a BSA set $S(G)$. In general, for $K = \text{SOS}[x]$ and a finite set $G$, the set of polynomials $\text{ch}(K \cdot G^{\leq 1})$ is identical to Putinar’s positivstellensatz, while $\text{ch}(K \cdot G^{\leq |G|})$ recalls a positivstellensatz by Schmüdgen [44].

Although we assume that $K$ is a cone of nonnegative polynomials, we do not assume that $K$ is finitely generated (e.g., $\text{SOS}[x]$ is not finitely generated). We also assume that $1 \in K$, which implies that $\mathbb{R}_+ \subseteq K$. We denote by $K$ the set of cones $K$ of nonnegative polynomials with $1 \in K$. For instance, $\text{SOS}[x] \subseteq K$.

### 3 Forward Invariance for Polynomial Systems

In this section, we present a sufficient condition on a BSA set to be forward invariant for a polynomial dynamical system. Consider a BSA set $\mathcal{P} \subseteq S(G)$ with $G = \{g_1, \ldots, g_m\} \subseteq \mathbb{R}[x]$. Given $g \in G$, the Lie derivative of $g$ along the field $f$ is defined by $L_f(g)(x) = \langle \nabla g(x), f(x) \rangle$. Concretely, $L_f(g)(x)$ gives the rate of change of the value of $g$ at $x$ along a trajectory of $\text{ODE}(f)$.

The idea of the sufficient condition for $\mathcal{P} \subseteq S(G)$ to be forward invariant is that when a trajectory inside $\mathcal{P}$ reaches a point $x$ on the boundary, i.e., some $g_i(x) = 0$, then its Lie derivative $L_f(g_i)(x)$ at $x$ should be nonpositive so that $g_i$ stays nonnegative. We say that Boundary $f(G)$ holds iff

$$\forall g \in G. \forall x \in \mathcal{P}. \quad g(x) = 0 \implies L_f(g)(x) \geq 0.$$ 

However, the Boundary $f$ condition (inspired by the theory of Lyapunov functions and barrier certificates [36]) does not imply forward invariance of $S(G)$ under $f$, in general. We need additional condition (e.g., those in Rem. 3.1 or 2 below). First, we note the following counterexample by Platzer [32].

**Example 3.1.** Consider the set $G = \{x^{-2}\}$ defining $\mathcal{P} = S(G) = \{x : x^2 \leq 0\} = \{0\}$ and $\text{ODE}(f) : x = -1$. Clearly, $\mathcal{P}$ is not forward invariant for $\text{ODE}(f)$. However, whenever $x^2 = 0$ (that is, $x = 0$), we have $L_f(-x^2) = -2x x = 2x \geq 0$, i.e., Boundary $f(\mathcal{P})$ holds. The reason is because $L_f(-x^2)$ cannot be expressed as a Lipschitz function of the polynomials in $G$ (compare with (2) below).

**Remark 3.1.** Let us note that if the condition $L_f(g(x)(\geq 0)$ is changed to $L_f(g)(x)(> 0)$, then we can avoid cases such as those mentioned above and prove soundness [13].

We refine the condition (1) using the notion of cone introduced before. This will have the double advantage of (i) ensuring invariance, and (ii) making the condition easier to verify/enforce numerically. We say that the predicate $\text{Forward}_f(G; \lambda, K, \ell)$ holds iff

$$\forall g \in G. \quad L_f(g) + \lambda g \in \text{ch}(K \cdot G^{\infty}),$$

where $\lambda \in \mathbb{R}$, $K \subseteq \mathbb{K}$ and $\ell \in \mathbb{N}$ are fixed.

**Example 3.2.** Consider the dynamics $\text{ODE}(f)$ with $f(x) = -x^3$, and let $G = \{g_1, g_2\}$ with $g_1 = x + 2$ and $g_2 = 2 - x$. We show that $\text{Forward}_f(G; \lambda, K, \ell)$ holds for $\lambda = 2$, $K = \text{SOS}[x]$ and $\ell = 1$. Indeed, $L_f(g_1) + \lambda g_1 = -x^3 + 3x + 4 = (x + 1)^2 \cdot (2 - x) + 2$, where $(x + 1)^2, 2 \in \text{SOS}[x]$. The proof is similar for $g_2$; thus omitted.

**Remark 3.2.** If $\text{Forward}_f(G; \lambda, K, \ell)$ holds then Boundary $f(G)$ holds, but not vice-versa. In Example 3.1 above, we have Boundary $f(G)$ but we can show that for any choice of $K \subseteq \mathbb{K}, \lambda \in \mathbb{R}$ and $\ell \in \mathbb{N}$, $\text{Forward}_f(G; \lambda, K, \ell)$ does not hold.

We now prove that the condition Forward is sufficient to ensure the forward invariance of $\mathcal{P}$. The proof is by relating the evolution of $g_1(\phi(t)), \ldots, g_m(\phi(t))$ over a trajectory $\phi$ and a nonlinear (internally) positive system. We briefly define such systems.

**Definition 3.3.** A dynamical system of the form $x = f(x, u)$ with state $x \in \mathbb{R}^n$, input $u \in \mathbb{R}^k$ and $f$ Lipschitz continuous in $x$ and $u$ is (internally) positive if every trajectory $\phi : [0,T) \rightarrow \mathbb{R}^n$ of the
A forward invariant for constrained and switched polynomial systems

We extend the condition Forward$_f(G; \lambda, K, t)$ to cover forward invariance for constrained and switched polynomial systems. First, we consider a constrained continuous-time dynamics

\[ \text{ODE}(f, D) : \quad x = f(x), \quad x \in D, \]

wherein $D = S(G_{\text{dom}})$ is a BSA set with finite set $G_{\text{dom}} \subseteq \mathbb{R}[x]$. A set $P \subseteq \mathbb{R}^n$ is forward invariant for ODE$(f, D)$ if for all initial conditions $\phi(0) \in P$, it holds that for all $t \in \text{dom}(\phi(t))$, $\phi(t) \in P$.

We say that Forward$_f,f_d(G; \lambda, K, t)$ holds iff

\[ \forall g \in G, L_f(g) + \lambda g \in \text{ch}(K \cdot (G_{\text{dom}})_{\leq t}) \]

wherein $\lambda \in \mathbb{R}$, $K \subseteq \mathbb{R}$ and $t \in \mathbb{N}$ are fixed. Comparing (4) with (2) for ODEs without constraints, we note that the set of polynomials $G_{\text{dom}}$ is combined with $G$. The soundness of the condition “Forward” extends to constrained systems:

**Theorem 2.4**. If Forward$_f,f_d(G; \lambda, K, t)$ holds, then $P \subseteq S(G)$ is forward invariant for ODE$(f, D)$.

Proof. The proof follows the same structure as Theorem 3.5. Again, we fix a trajectory $\phi$ of ODE$(f, D)$ with $\phi(0) \in P$. Since for all $t \in \text{dom}(\phi(t))$, $\phi(t) \in D$, we conclude that for all $h \in G_{\text{dom}}$, $t \in \text{dom}(\phi(t))$, $h(\phi(t)) \geq 0$. Consider the function $\omega : \text{dom}(\phi(t)) \rightarrow \mathbb{R}^m$ defined by $\omega(t) = (g_1(\phi(t)), \ldots, g_m(\phi(t)))$. It holds that $\omega(0) \in \mathbb{R}^m$. Furthermore, for all $t \in \text{dom}(\phi(t))$, $\omega(t) = (L_f(g_1)(\phi(t)), \ldots, L_f(g_m)(\phi(t)))$. Hence, Forward$_f,f_d(G; \lambda, K, t)$ implies that for each $i \in [m]$, $L_f(g_i) + \lambda g_i = \sum_{j=1}^{n_i} h_{ij} \delta_{ij}$, for some $s_i \in N$, $h_{ij} \in K$ and $g_{ij} \in G_{\leq t}$. It follows that $\omega(t)$ is a trajectory of the dynamical system

\[ \omega = -\lambda \omega(t) + F(\omega(0), \omega(t)), \]

wherein $F(\omega, x) = (F_1(\omega, x), \ldots, F_m(\omega, x))$ satisfies that for all $i \in [m]$, $\omega \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$, $F_i(\omega, x) \in \mathbb{R}$. Thus, by Theorem 3.4, (3) is a positive system and $\omega(t) \in \mathbb{R}^m$ for all $t \in \text{dom}(\phi(t))$. \qed

Finally, note that Forward is monotonic with respect to $\lambda$.

**Proposition 3.6.** Let $K \subseteq \mathbb{R}$, and $\lambda_1 \leq \lambda_2 \in \mathbb{R}$ such that $\lambda_1 \leq \lambda_2$. It holds that Forward$_f(G; \lambda_1, K, t) \Rightarrow$ Forward$_f(G; \lambda_2, K, t)$.

Proof. Let us assume Forward$_f(G; \lambda_1, K, t)$ holds. Then, for each $i \in [m]$, $L_f(g_i) + \lambda_1 g_i = \sum_{j=1}^{n_i} h_{ij} \delta_{ij}$, for some $s_i \in \mathbb{N}$, $h_{ij} \in K$ and $g_{ij} \in G_{\leq t}$. We have

\[ L_f(g_i) + \lambda_2 g_i = L_f(g_i) + \lambda_1 g_i + (\lambda_2 - \lambda_1) g_i \]

wherein $h_{ij} = h_{ij} + (\lambda_2 - \lambda_1) h_{ij}$, and $g_{ij} = g_{ij}$ otherwise. Note that $h_{ij} \in K$ since $h_{ij} \in K$, $\lambda_2 - \lambda_1 \in \mathbb{R} \subseteq K$, and $K$ is closed under addition since it is a cone. \qed

**4.1 Forward Invariance for Switched Polynomial Systems**

We recall switched dynamics of the form ODE$_{Q_1,Q_2,D}$ and their semantics from Section 2.2. $Q$ is the finite set of modes and each mode $q \in Q$ is associated with a constrained polynomial system $(f_q, D_q)$. Furthermore, $T$ is a finite set of transitions wherein each $r \in T$ is a triple $(a_r, b_r, G_r)$ for pre/post modes $a_r, b_r \in Q$ and guard set $G_r = S(G_{\text{guard},r})$.

We will consider a collection of sets $\Psi = \{P_q\}_{q \in Q}$ wherein for each $q \in Q$, $P_q = S(G_q)$ for a finite set $G_q \subseteq \mathbb{R}[x]$.

**Definition 4.3.** (Forward Invariance for Switched System). $\Psi$ is a forward invariant for ODE$_{Q_1,Q_2,D}$, $T$ if for all trajectory $(\phi, q_{0:k}, t_{k+1})$ of ODE$_{Q_1,Q_2,D}$, $T$ with $\phi(0) \in P_{q_0}$, it holds that for all $t \in \text{dom}(\phi(t))$, $\phi(t) \in P_{q_t}$ if $t \in [t_{i}, t_{i+1})$.

The condition Forward$_{Q_1,Q_2,D}(G_q; \lambda, K, t)$ is:

1. For each $q \in Q$, Forward$_{q,D}(G_q; \lambda, K, t)$ holds.
For all \( \tau \in T \), we require that \( \mathcal{P}_0 \cap \mathcal{G}_\tau \subseteq \mathcal{P}_0 \). For that, we enforce a sufficient condition over \( \mathcal{G}_x \) and \( \mathcal{G}_b \):
\[
\mathcal{G}_b \subseteq \text{ch}(K \cdot (\mathcal{G}_a \cup \mathcal{G}_{\text{guard},t}))^\mathcal{S}.
\]

**Theorem 4.4.** If \( \text{Forward}_I((\mathcal{G}_q)_{q \in \mathcal{Q}}; \mathcal{L}, K, \ell) \) holds, then \( \mathcal{B} \) is forward invariant for \( \mathcal{S} = \text{ODE}(q, f, \delta, \mathcal{D}_{q^\mathcal{Q}}; T) \).

**Proof.** Let \((\phi, q_0, k, t_{k+1})\) be a trajectory of \( \mathcal{S} \) with \( \phi(t_0) \in \mathcal{P}_{q_0} \), and let \( t_1, \ldots, t_k \) be the associated sequence of transitions. We will establish by induction the following two facts for all \( i \in [k] \cup \{0\} \): (a) \( \phi(t_i) \in \mathcal{P}_{q_i} \); and (b) for all times \( t \in [t_i, t_{i+1}) \), \( \phi(t) \in \mathcal{P}_{q_i} \). These two facts will establish the forward invariance of \( \mathcal{B} \).

For the base case \( i = 0 \), we note that \( \phi(t_0) \in \mathcal{P}_{q_0} \) by assumption. Since \( \text{Forward}_I((\mathcal{G}_q)_{q \in \mathcal{Q}}; \mathcal{L}, K, \ell) \) implies \( \text{Forward}_I((\mathcal{G}_q)_{q \in \mathcal{Q}}; \mathcal{L}, K, \ell) \), it follows by Theorem 4.2 that \( \mathcal{P}_{q_i} \) is forward invariant for the mode \( q_0 \). Therefore, \( \phi(t) \in \mathcal{P}_{q_i} \) for all \( t \in [t_0, t_1) \). The base case is thus established.

Now, let us look at the case \( i = 1 \). By continuity of \( \phi \) and because \( \mathcal{P}_{q_i} \) is a closed set, we have that \( \phi(t_1) \in \mathcal{P}_{q_1} \). Since \( \phi(t) \in \mathcal{G}_{t_1} \), we have \( \phi(t) \in \mathcal{P}_{q_1} \cap \mathcal{G}_{t_1} \). Note that \( \text{Forward}_I((\mathcal{G}_q)_{q \in \mathcal{Q}}; \mathcal{L}, K, \ell) \) implies \( \mathcal{P}_{q_1} \cap \mathcal{G}_{t_1} \subseteq \mathcal{G}_{q_1} \). Thus, \( \phi(t) \in \mathcal{P}_{q_1} \). We can then conclude in the same as the base case that for all \( t \in [t_1, t_2) \), \( \phi(t) \in \mathcal{P}_{q_1} \). The case \( i = 1 \) is thus established.

The proof for the cases \( i > 1 \) is identical. \( \square \)

We will now turn our attention to computing such forward invariants for a given system with initial conditions.

## 5 REFINEMENT OPERATORS AND FIXED POINT FORMULATION

Given a polynomial system \( (f, I) \), we wish to compute a BSA set \( \mathcal{P} = S(G) \) for a finite set of polynomials \( G \subseteq \mathbb{R}[x] \) such that \( \mathcal{P} \supseteq I \) (the initial set is contained) and \( \mathcal{P} \) is forward invariant for ODE(f). Note that we do not explicitly enforce in this paper that \( \mathcal{P} \cap \mathcal{S}_{\text{ unsafe}} = \emptyset \), as is common in many approaches to invariant synthesis based on abstract interpretation. Explicit use of \( \mathcal{S}_{\text{unsafe}} \) (example for early termination) will be considered in future work.

Our approach is based on using the framework of abstract interpretation, first introduced by Cousot and Cousot to compute invariants for programs [14, 15]. We will present a similar approach to compute polynomial invariants for polynomial systems. The first step is to define the invariant we seek as a pre-fixed point \( G \) of a monotone operator on the space of cones in \( \mathbb{R}[x] \). The set \( S(G) \) will give us the invariant set \( \mathcal{P} \).

Let \( G \subseteq \mathbb{R}[x] \) be a finite set of polynomials. Recall that \( \text{ch}(G) \) contains all the conic combinations of elements in \( G \). Let us fix a cone \( K \subseteq \mathcal{K} \) of nonnegative polynomials. We define a refinement operator that takes us from \( \text{ch}(G) \) to a new cone \( \text{ch}(G') \).

**Definition 5.1 (Refinement Operator).** Given \( G = \{ g_1, \ldots, g_m \} \subseteq \mathbb{R}[x] \), we define the refinement of \( G \) as follows:
\[
\delta_f(G; \mathcal{L}, K, \ell) = \{ g \in \text{ch}(G) : L_f(g) + \lambda g \in \text{ch}(K \cdot G^\mathcal{S}) \}.
\]

First, note that \( \delta_f(G; \mathcal{L}, K, \ell) \) is a cone (one can easily show that it satisfies the axioms of Def. 2.9). Furthermore, as a corollary of Theorem 3.5, it holds that any finitely generated pre-fixed point of the refinement operator is a forward invariant set.
be written as $\sum_{i=1}^s a_i h_i$ for multipliers $a_i \geq 0$. Also, note that $G^{\leq t}$ is finite, i.e., $G^{\leq t} = \{g_1, \ldots, g_m\}$. Any element of $K \cdot G^{\leq t}$ can thus be written as $\sum_{i=1}^s (\sum_{j=1}^{\ell} \beta_i j) \hat{g}_j$ for multipliers $\beta_i \geq 0$. It follows that any element of $\text{ch}(K \cdot G^{\leq t})$ can be written as $\sum_{i=1}^s (\sum_{j=1}^{\ell} \beta_i j) \hat{g}_j$ for multipliers $\beta_i \geq 0$. Hence, the condition that $p \in \text{ch}(G)$ and $L_f(p) + \lambda p \in \text{ch}(K \cdot G^{\leq t})$ can be written as two equality constraints that are linear in the variables $a_i$ and $\beta_i$. This, plus the nonnegativity constraints $a_i \geq 0$ and $\beta_i \geq 0$, defines a polyhedral cone $P$ over these variables. Now, the projection $P'$ of $P$ over the $m$ variables $a_i$ is also a polyhedral cone. Hence, $P'$ is finitely generated. Finally, since $\partial_f(G; \lambda, K, t) = \{\sum_{i=1}^m a_i g_i : (a_1, \ldots, a_m) \in P'\}$, it holds that $\partial_f(G; \lambda, K, t)$ is finitely generated, concluding the proof.

The constructive proof of Lemma 5.4 provides a way of implementing IsFixedPoint and FiniteRefinement when $K$ is finitely generated. However, with finitely generated cones $K$ is impractical for two reasons: (i) Using a small set of generators $H$ often leads to an overly conservative refinement operator, thereby preventing us from proving invariance even for simple systems; (ii) The number of polynomials in $G_{\tau}$ grows extremely fast (super-exponential in the number of iterations in the worst case), thereby preventing from applying more than a few iteration.

Therefore, in the next section, we consider a different approach using the SOS cone and wherein the size of $G_{\tau}$ is kept constant.

### 5.1 Bounded-Size Iterates

In this approach, we let $K$ be the set of SOS polynomials of degree at most $2d$, denoted by $\text{SOS}_{2d}[x]$. Note that $\text{SOS}_{2d}[x]$ is not finitely generated. The idea of the approach is to compute a finite set $G' = \{g_1', \ldots, g_m'\}$ included in $\partial_f(G; \lambda, K, t)$, wherein $G = \{g_1, \ldots, g_m\}$. Note that $|G'| = |G|$.

Sampling from a convex set is a problem that has received some attention in the literature. We can for instance mention the hit-and-run algorithm which is a Monte-Carlo method to sample random points inside a given set [12, 27]. However, in this work, we consider another approach based on projections.

Concretely, let $\|\cdot\|$ be a norm on $\mathbb{R}[x]$. Given a subset $H \subseteq \mathbb{R}[x]$, define the projection on $H$ by $\text{Proj}(g; H) = \arg\min_{h \in H} \|g - h\|$. The key insight is that using sum-of-squares (SOS) optimization, we can compute $\text{Proj}(g; H)$ where $H = \partial_f(G; \lambda, K, t)$ when $G$ is finitely generated set of polynomials and $K = \text{SOS}_{2d}[x]$. Using the projection operator allows us to implement IsFixedPoint and FiniteRefinement as shown in Algos. 2 and 3. The IsFixedPoint procedure simply checks that $\text{Proj}(g_i; H) = g_i$ for each $g_i \in G$. Similarly, the FiniteRefinement procedure computes the set $G' = \{\text{Proj}(g_i; H) \mid g_i \in G\}$ wherein $G = \partial_f(G; \lambda, K, t)$.

Note that $H = \partial_f(G; \lambda, K, t)$ in Algos. 2 and 3 is a convex set that can be represented by Linear Matrix Inequalities [24, 31]. Hence,

**Algorithm 2:** IsFixedPoint using Projections

**Data:** Fine set $G \subseteq \mathbb{R}[x]$, $\lambda \in \mathbb{R}$, $K \in \mathcal{K}$, $t \in \mathbb{N}$.

1. Let $H = \partial_f(G; \lambda, K, t)$.
2. If for all $g \in G$, $\text{Proj}(g; H) = g$ then return True
3. Else return False

**Algorithm 3:** FiniteRefinement using Projections

**Data:** $G = \{g_1, \ldots, g_m\} \subseteq \mathbb{R}[x]$, $\lambda \in \mathbb{R}$, $K \in \mathcal{K}$, $t \in \mathbb{N}$.

1. Let $H = \partial_f(G; \lambda, K, t)$.
2. For $i = 1, \ldots, m$ do let $g'_i = \text{Proj}(g_i; H)$
3. Return $\{g'_1, \ldots, g'_m\}$

computing $\text{Proj}(g_i)$ can be done efficiently, e.g., using semidefinite programming [11].

As mentioned before, computing $G_{\tau+1}$ as the output of Algo. 3 is advantageous because it keeps the size of $G_{\tau}$ constant throughout the process. However, it might be slow to make progress, so that a large number of iterations might be needed before finding invariant (if we eventually find one). Another limitation of the approach is the sensitivity to numerical errors. Indeed, when using a numerical solver to compute $\text{Proj}(g, H)$, we might get something close to but not exactly in $H$. In this case, we cannot certify that a “numerical” fixed point is an actual fixed point. To address these limitations, we define a robust version of the above approach.

#### 5.1.1 Robust Projection and Acceleration.

The robust projection relies on an inner-approximation of $\partial_f(G; \lambda, K, t)$ parameterized by a robustness parameter $\varepsilon > 0$:

$$\partial_f(G; \varepsilon, \lambda, K, t) = \{g \in \text{ch}(G) : L_f(g) + \lambda g - \varepsilon \cdot \|g\| \in \text{ch}(K \cdot G^{\leq t})\}.$$ 

The resulting implementation of FiniteRefinement is the same as in Algo. 3 but with $H = \partial_f(G; \varepsilon, \lambda, K, t)$; see Algo. 4.

The main result of this section is that if the output of Algo. 4 is close to $G$, then $S(G)$ is forward invariant.

**Lemma 5.5.** Assume that $S(G)$ is compact and let $\varepsilon > 0$. Denote $H = \partial_f(G; \varepsilon, \lambda, K, t)$. There exists a constant $\kappa > 0$ depending only on $G$, $f$, $\lambda$ and $\varepsilon$, such that for all $g \in G$, if $\|g - \text{Proj}(g, H)\| \leq \kappa \cdot \|g\|$, then $S(G)$ is forward invariant.

**Proof.** Let $D : \mathbb{R}[x] \rightarrow \mathbb{R}[x]$ be defined by $D(g) = L_f(g) + \lambda g$, and let $\delta = \max \{|D(g(x))| : x \in \text{S}(G), \|g\| \leq 1\}$. Without loss of generality, let $g \in G$ be such that $\|g\| = 1$, and assume that there is $g' \in H$ such that $\|g - g'\| \leq \kappa$ wherein we choose $\kappa < \frac{1}{\sqrt{\delta}}$. Let us set $\kappa(G) = \kappa$. First, note that this implies that $\|g'\| \leq 1 - \kappa$. Second, by definition of $\delta$, we note that $|D(\frac{g - g'}{\|g'\|})| \leq \delta$ since $\|g - g'\| \leq \kappa$. $D(\cdot)$ being a linear operator, we obtain $D(g) \geq D(g') - \delta \kappa$ on $S(G)$. Finally, since $g' \in H$, it holds that $D(g') \geq \|g'\| \cdot S(G)$. Hence, $D(g) \geq \varepsilon - \varepsilon \kappa - \delta \kappa$ on $S(G)$. By our choice of $\kappa$, it follows that $D(g) > 0$ on $S(G)$. Hence, $S(G)$ is forward invariant.

Hence, in our algorithmic process, if at some point $G'$ is “close enough” to $G$, then we stop the algorithm and return $G$. However, estimating the value of $\kappa$, in practice, is hard. We will simply set it to be a constant smaller than $\varepsilon$, typically $\frac{\varepsilon}{2}$, and use an SMT solver such as Z3 to validate the final positive invariant.
Algorithm 5: Initial iterate using Sample Points

**Data:** Finite set \( G_{\text{init}} \subseteq \mathbb{R}[x] \), \( K \in \mathcal{K} \), template \( \left\{ p_1, \ldots, p_k \right\} \subseteq \mathbb{R}[x] \), sample points \( x_1, \ldots, x_N \in \mathbb{R}^n \).

1. Let \( B = \left\{ \sum_{j=1}^{k} a_j p_j : a_j \in \mathbb{R} \right\} \)
2. Let \( \mathcal{G} = \left\{ g \in B : g(x_i) \geq 0, \, i = 1, \ldots, N \right\} \)
3. Let \( G_{-1} \subseteq \mathbb{R}[x] \) be a finite set such that \( \mathcal{G} = \text{ch}(G_{-1}) \)
4. Let \( G_0 = \{ \text{Proj}(g; \text{ch}(K \cdot G_{\text{init}})) : g \in G_{-1} \} \)

**Figure 2:** The sets \( S(G_\sigma) \) for \( \sigma = 0, \ldots, 6 \) for the Vanderpol oscillator from Example 2.4 and the template in Example 5.7. The inner most curve corresponds to \( \sigma = 0 \), then \( \sigma = 1 \), etc.

5.2 Initial Iterate from Simulations

The only constraint we have on the initial iterate \( G_0 \) (line 1) is that \( I \subseteq S(G_0) \). Hence, one could for instance just set \( G_0 = G_{\text{init}} \). However, the more functions in \( G_0 \), the more expressive the subsequent iterates since they are all subsets of \( \text{ch}(G_0) \). This is illustrated in the example below.

**Example 5.6.** Let \( G_{\text{init}} = \{1-x_1^2-x_2^2\} \), so that \( I \) is the ball of radius one around the origin. If \( G_0 = G_{\text{init}} \), then all sets \( G_\sigma \) are subsets of \( \text{ch}(G_{\text{init}}) = \{\alpha + \beta - \alpha x_1^2 - \alpha x_2^2 : \alpha, \beta \geq 0\} \). Hence, the iterates can only describe balls of radius at least one around the origin. However, for instance, if we let \( G_0 = \{1-x_1^2, 1-x_2^2, 1-2x_1x_2\} \), then we still have that \( I \subseteq S(G_0) \). However, the iterates can describe a larger variety of sets such as ellipsoids.

In order to define a rich initial iterate, we use sample points from simulations. Then, we define the initial iterate as the set of all polynomials in a given template that are nonnegative at the sample points. Finally, we project on \( \text{ch}(K \cdot G_{\text{init}}) \). This is implemented in Algo. 5. Note that \( B \) is a linear subspace; then, \( \mathcal{G} \) is a subset of \( B \) obtained by imposing linear inequality constraints; hence, \( \mathcal{G} \) is a finitely generated cone; this allows to compute \( G_{-1} \) in line 3.

**Example 5.7.** Consider the system of Example 2.4. The sample points are given by the trajectories in Fig. 1. We use \( \{1, x_1^2, x_1x_2, x_2^2\} \) as template to allow any homogeneous quadratic curves. The first seven iterates of the overall procedure are depicted in Fig. 2.

5.3 Finite Termination

Finitely, we discuss the termination of the algorithm. Unfortunately, the approach in Sec. 5.1 does not guarantee that the iteration will converge in finite time to a fixed point. Therefore, we introduce a widening of the refinement operator, that can be applied only a finite number of times before obtaining the empty set, thereby ensuring termination of the algorithm in finite time.

The widened operator removes the generators that are not in the refinement. This is implemented in Algo. 6. Note that the condition in line 4 can be checked efficiently, e.g., if \( K = \text{SOS}_d[x] \) (see Sec. 5.1). It holds that the output \( G' \) of Algo. 6 has cardinality strictly lower than the input \( G \) if \( G \) is not a fixed point.

**Lemma 5.8.** If \( G \) is not a fixed point, i.e., \( G \not\subseteq \partial_f(G; \lambda, K, \ell) \), then the output \( G' \) of Algo. 6 satisfies \( |G'| < |G| \).

**Proof.** If \( G \not\subseteq \partial_f(G; \lambda, K, \ell) \), then there is some \( g_i \in G \) such that \( \text{Proj}(g_i; H) \neq g_i \) so that concluding the proof.

As a corollary, Algo. 6 can be applied recursively at most \( |G_0| \) times before reaching a (possibly trivial) fixed point. Note that the widening operator is applied only after several iterations of the less conservative projection-based refinement operator is applied.

6 NUMERICAL EXPERIMENTS

We applied the algorithmic process from Sec. 5 to compute invariant BSA sets for several polynomial systems.

**Implementation Details.** We implemented the algorithm in Julia. To compute the projections in Secs. 5.1 and 5.2, we used the package SumOfSquares.jl [53] with the SDP solver Mosek [9]. To compute a finite set of initial iterates in Sec. 5.2, we used Polyhedra.jl [25] with CDDLib [18]. We used only the robust projection approach (Sec. 5.1.1); in particular, no widening as in Sec. 5.3 was needed. The parameters we used for the robust projection were \( \lambda = 1, K = \text{SOS}_d[x] \) with \( d \) inferred automatically by the solver, \( \ell = 1, \epsilon = 0.1, \text{and } \kappa = 10^{-8} \). We believe that \( \epsilon = 0.1 \) is sufficiently large and \( \kappa = 10^{-8} \) is sufficiently small to ensure sound invariants despite possible numerical inaccuracies inherent to SDP solvers. However, a rigorous analysis of the robustness to numerical errors is beyond the scope of this paper. Therefore, whenever possible, we verified the returned invariant using the SMT solver Z3 [17]. We compared our approach, which uses multiple polynomials \( g_1, \ldots, g_m \), with the one using a single polynomial \( g \) possibly of higher degree.

**Results.** All computations were made on a laptop with processor Intel Core i7-7600u and 16GB RAM running Windows. The timing results, number of iterations and number of polynomials in the invariants for the different numerical experiments are reported in Table. 1. The invariants are reported in the Appendix.
We consider the dynamical system given by

\[ \frac{\text{d}x}{\text{d}t} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \]

with initial set \( I = \{ x_1^2 + x_2^2 \leq \frac{1}{2} \} \), depicted in Fig. 3a. We considered two templates to compute an invariant set for this system: \( H_1 = \{ 1, x_1, x_2 \} \) and \( H_2 = \{ 1, x_1^2, x_1 x_2, x_2^2 \} \). The invariants obtained by the algorithm using each template are represented in Fig. 3c. Both invariants were verified using Z3.

### 6.4 Unstable 2D Nonlinear System

We consider the dynamical system given by

\[ x_1 = 1 - x_1^3 + x_2, \quad x_2 = -\frac{1}{2} + x_1^3 - x_2, \]

with initial set \( I = \{ x_1^2 + x_2^2 \leq \frac{1}{4} \} \), depicted in Fig. 3d. We considered two templates to compute an invariant set for this system: \( H_1 = \{ 1, x_1, x_2 \} \) and \( H_2 = \{ 1, x_1^2, x_1 x_2, x_2^2 \} \). The invariants obtained by the algorithm using each template are represented in Fig. 3d. Note that in this case, the invariants are not compact. Thus, Lemma 5.5 does not apply. Hence, it is important to verify the invariants with Z3. The first invariant (with \( H_1 \)) was easily verified using Z3. However, the verification of the second invariant (with \( H_2 \)) using Z3 timed out. Nevertheless, by manually removing four polynomials from the invariant, we obtained a new set (depicted in Fig. 3d), for which we could prove the invariance with Z3. We synthesized a forward invariant described by a single SOS polynomial \( q \). For degree \( d \in \{ 2, 4, 6 \} \), no such \( q \) could be found. For \( d = 8 \), the solver returned the polynomial \( g \) (given in the Appendix and depicted in Fig. 3a). However, due to the high degree of the polynomial, Z3 timed out in the verification of the invariant (>12 hours).

### 6.5 3D Nonlinear System

We consider the dynamical system, inspired by [2, Example 7]:

\[
\begin{align*}
    \dot{x}_1 &= 1 - x_1 x_2 - x_1^3 + x_2^3 - x_1 x_1^2 x_2 - x_2^3, \\
    \dot{x}_2 &= -x_2 - x_2 x_1^3 - x_1^2 x_2 - x_1 x_2^2, \\
    \dot{x}_3 &= -4x_3 - x_1^3 + 3x_1^2 x_2 + 3x_1 x_2^2.
\end{align*}
\]

with initial set \( I = \{ x_1^2 + x_2^2 + x_3^2 \leq \frac{1}{4} \} \). We considered the template \( H = \{ 1, x_1^2, x_2^2, x_3^2 \} \). The algorithm generated a BSA set described by 729 polynomials. The verification using Z3 timed out. However, by manually selecting and keeping only one polynomial, we obtained a larger set, for which we could verify the invariance with Z3.

### 6.6 4D Nonlinear System

We consider the dynamical system, inspired by [2, Example 9]:

\[
\begin{align*}
    \dot{x}_1 &= -x_1 - 3x_1 x_4 + x_3^3, \\
    \dot{x}_2 &= -x_2 - x_1 x_4, \\
    \dot{x}_3 &= -x_3 - x_1 x_4, \\
    \dot{x}_4 &= x_3 - x_1^2 - x_4.
\end{align*}
\]

with initial set \( I = \{ x_1^2 + x_2^2 + x_3^2 + x_4^2 \leq \frac{1}{4} \} \). We considered the template \( H = \{ 1, x_1^2, x_2^2, x_3^2, x_4^2 \} \). The algorithm generated a BSA set described by 162 polynomials. The verification using Z3 timed out.

### 6.7 Switched System with Limit Cycle

We consider the switched dynamical system

\[
\begin{align*}
    \dot{x}_1 &= -\frac{1}{2} - x_2 - \frac{1}{3} x_1 + \frac{1}{2} x_2^2 + \frac{1}{2} x_1 x_2^2 - \frac{1}{2} x_1^2 x_2 - \frac{1}{2} x_1^3, & \text{if } x_1 \geq 0, \\
    \dot{x}_2 &= 1 + \frac{1}{2} x_2 + x_1 + \frac{1}{2} x_1 x_2 - \frac{1}{2} x_1^2 x_2, & \text{if } x_1 < 0.
\end{align*}
\]

with initial set \( I = \{ x_1^2 + x_2^2 \leq \frac{1}{4} \} \), depicted in Fig. 3e. We considered the template \( H = \{ 1, x_1^2, x_1 x_2 \} \). Our approach to compute an invariant for this system was: (i) compute an invariant \( P_1 \) for the
first mode containing $\mathcal{I}$; (ii) compute the intersection $\mathcal{I}_{12}$ between $\mathcal{P}_1$ and the guard $\mathcal{G}_{12}$ from the first to the second mode, which is a BSA set; (iii) compute an invariant $\mathcal{P}_2$ for the second mode containing $\mathcal{I}$ and $\mathcal{I}_{12}$; (iv) compute the intersection $\mathcal{I}_{21}$ between $\mathcal{P}_2$ and the guard $\mathcal{G}_{21}$ from the second to the first mode, which is a BSA set; (v) check that $\mathcal{I}_{21}$ is contained in $\mathcal{P}_1$; since it was the case, we stopped. The resulting invariant was verified using Z3.

6.8 Switched Bistable System

We consider the switched dynamical system

\[
\begin{align*}
\dot{x}_1 &= 0.1 - x_2 - 0.3x_1 + 0.3x_1^2 - 0.1x_1^3, & \text{if } x_1 \geq 0, \\
\dot{x}_2 &= -2 + 2x_1 - 0.1x_2, \\
\dot{x}_1 &= -0.125 - 2x_2 - 0.25x_1, & \text{if } x_1 \leq 0, \\
\dot{x}_2 &= 0.25 - 0.25x_2 + 0.5x_1,
\end{align*}
\]

with initial set $\mathcal{I} = \{x_1^2 + x_2^2 \leq \frac{1}{2}\}$, depicted in Fig. 3f. We considered the template $H = \{1, x_1, x_2, x_2^2\}$. We split the first mode into two “sub-”modes, one for $x_2 \geq 0$ and one for $x_2 \leq 0$ and applied the procedure described in Sec. 6.7 to compute invariants that were successfully verified using Z3.

7 CONCLUSIONS

We provided an algorithmic framework to compute semi-algebraic invariants for polynomial systems. We expressed the invariance condition as a fixed point of a refinement operator over cones of polynomials. A key element of the framework is the introduction of the projection operator to compute the refinement, which allows to keep the complexity and convergence of the refinement process under control. In future work, we plan to apply this refine-by-projection approach to other fixed-point approaches in abstract interpretation. We also plan to improve the approach by allowing to detect when the size of the iterates can be reduced because some polynomials have become redundant.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their detailed comments and suggestions. This research was funded in part by the Belgian-American Education Foundation (BAEF) and the US National Science Foundation (NSF) under award numbers 1836900 and 1932189.
A COMPUTED INVARIANTS FOR THE NUMERICAL EXPERIMENTS

A.1 Invariants in Sec. 6.1

Computed invariant:

\[-0.99301954986876 + 0.0377815384988998* x_2^2 - 0.0666256032861187* x_1 + x_2 - 0.08956070963519557* x_1^2,\]
\[-0.9992011592628854 + 0.007955071759038876* x_2^2 + 0.0041018047194553466* x_1 + x_2 + 0.0389487854357559354* x_1^2,\]
\[-0.9991994975155781 + 0.007562020589211854* x_2^2 + 0.0042485198887888844* x_1 + x_2 + 0.0390529139500008275* x_1^2,\]
\[-0.9992137430607909 + 0.008771034196869584* x_2^2 + 0.0036680198441712666* x_1 + x_2 + 0.0388488210965402* x_1^2,\]
\[-0.9992827480108409 + 0.011526972259621198* x_2^2 + 0.00166399042857424* x_1 + x_2 + 0.03603262057205512* x_1^2,\]
\[-0.9993785554131969 + 0.01537369337658755* x_2^2 - 0.0017643808674354974* x_1 + x_2 + 0.03167079837241498* x_1^2,\]
\[-0.9985171163578682 + 0.032865540164738846* x_2^2 - 0.00579264202647794* x_1 + x_2 - 0.02454214265693337* x_1^2,\]
\[-0.9988802218290721 + 0.031494103922956214* x_2^2 - 0.03135885750959462* x_1 + x_2 - 0.01621872347865698* x_1^2.\]

Invariant after removing seven polynomial:

\[-0.993031954986876 + 0.0377815384988998* x_2^2 - 0.0666256032861187* x_1 + x_2 - 0.08956070963519557* x_1^2,\]
\[-0.9991994975155781 + 0.007562020589211854* x_2^2 + 0.0042485198887888844* x_1 + x_2 + 0.0390529139500008275* x_1^2,\]
\[-0.9993785554131969 + 0.01537369337658755* x_2^2 - 0.0017643808674354974* x_1 + x_2 + 0.03167079837241498* x_1^2.\]

Single SOS polynomial:

\[-14.147180353695786 + 0.3119346001218656e-16* x_2^2 - 2.72203506254444e-16* x_1 - 1.144466139062474* x_2^2 + 2.364989150845086* x_1 + x_2 - 0.5856502928090498* x_1^2 - 3.43153414322688e-16* x_2^3 + 1.1896379017199832e-15* x_1 + x_2^2 - 1.0658662861406021e-15* x_1^2 + x_2 + 1.430034012737953e-16* x_1^3 + 0.08136329385159524* x_2^4 - 0.2704342780642278* x_1 + x_2^3 + 0.9431646915713372* x_1^2 + x_2^2 - 1.10223318894562621* x_1^3 + x_2 + 0.10534350132410865* x_1 + x_2^4 + 3.7852391977715565e-17* x_2^5 - 1.0899073204583052e-16* x_1 + x_2^4 + 2.7178771719675984e-16* x_1^2 + x_2^3 - 4.647037756088389e-16* x_1^3 + x_2^2 + 1.547447895294992e-16* x_1 + x_2^4 + 8.37092411653621e-20* x_1^5 - 0.028676934757914243* x_2^6 + 0.12282021340742454* x_1 + x_2^5 - 0.3819521807112903* x_1 + x_2^4 + 0.02627925175346396* x_1 + x_2^3 - 0.34524587137427354* x_1 + x_2^2 + 0.1077157353563978* x_1 + x_2^2 + 3.97813088993918e-6* x_1 - 9.770977101795868e-19* x_2^7 + 6.6663742436859206e-18* x_1 + x_2^6 - 1.3603234786515431e-17* x_1 + x_2^5 + 3.414552747312814e-17* x_1 + x_2^4 - 1.6599099673281645e-17* x_1 + x_2^3 + 4.065547544821152e-17* x_1 + x_2^2 + 2.4742367640606324e-20* x_1 + x_2^2 + 1.0723541315989710e-22* x_1^7 + 0.002680016807049046* x_2^8 - 0.014204639190999435* x_1 + x_2^7 + 0.044234100069372326* x_1 + x_2^6 - 0.0914071879176629* x_1 + x_2^5 + 0.06135054792366527* x_1 + x_2^4 - 0.00992063283303217* x_1 + x_2^3 + 0.0270457326139765* x_1 + x_2^2 - 8.062124627215044e-9* x_1^8.\]
A.3 Invariants in Sec. 6.3

Computed invariant with $H_1$:

$$-0.5367809350302005 + 0.7364709881468191 x_2 + 0.41167549284132915 x_1 x_2 + 0.5826353338500777 + 0.005118306539458857 x_2 + 0.8127175835968954 x_1,$$

$$0.326076086303661 x_1^2.$$

A.4 Invariants in Sec. 6.4

Computed invariant with $H_1$:

$$-0.5494478511158494 - 0.0096797866389297 x_2 - 0.8352962138678838 x_1,$$

$$-0.7084226824177525 - 0.010491865778762787 x_2 - 0.705677324805271 x_1,$$

$$-0.9981365603664998 - 7.156732576621838 x_1 - 0.61019686320016036 x_1,$$

$$-0.5724744460351411 - 0.8063258458465716 x_2 - 0.1486998283675083 x_1,$$

$$-0.9219036041387807 - 0.3873549679929587 x_2 - 0.00015317146607383 x_1,$$

$$-0.561241962399377 - 0.811320177029977 x_2 - 0.16360363268810975 x_1,$$

$$-0.525474119971067 - 0.8143011604485618 x_2 - 0.24655743617412532 x_1,$$

$$-0.5254687744128949 - 0.7146370028416534 x_2 - 0.46171043012532076 x_1,$$

$$-0.5803704215825563 - 0.01859674827176363 x_2 - 0.814104246522119 x_1,$$

$$-0.5254709394084618 - 0.2784999641909676 x_2 - 0.8039390908413512 x_1,$$

$$-0.5406994519179427 - 0.0099582787652706 x_2 - 0.8409781623580773 x_1.$$
Invariant after removing four polynomials:

Single SOS polynomial:
A.5 Invariants in Sec. 6.5

Computed invariant:

\[-0.28675786256423603 - 0.18280086902711773 \cdot x_2 + 0.12140102417765512 \cdot x_1 - 0.15460754380061761 \cdot x_2^2 + 0.1309402373957572 \cdot x_1 \cdot x_2 + 0.0179862099898157 \cdot x_1^2 - 2.2518593469162838 \cdot x_2^3 - 5.2635398890486801 \cdot x_1^2 \cdot x_2 + 5.32456770702873 \cdot x_1^2 \cdot x_2^2 - 2.11908286925807 \cdot x_1^3 - 4.230287442095737 \cdot x_2^4 + 14.88947023666147 \cdot x_1^3 \cdot x_2 - 3.833726478458332 \cdot x_1^4 - 0.38895473765656946 \cdot x_2^5 + 0.6830316635335733 \cdot x_1^3 \cdot x_2^2 - 2.799091160948933 \cdot x_1^2 \cdot x_2^3 + 3.6150946275380034 \cdot x_1^3 \cdot x_2^4 - 1.809841753837602 \cdot x_1^4 \cdot x_2 + 0.595344336340465 \cdot x_1^5 + 6.3660968391039425 \cdot x_2^6 - 26.921765938983317 \cdot x_1 \cdot x_2^5 - 51.0900207154662 \cdot x_1 \cdot x_2^4 + 60.67932780655055 \cdot x_1^3 \cdot x_2^3 - 50.30016099504679 \cdot x_1^2 \cdot x_2^4 - 26.462677835173247 \cdot x_1^5 \cdot x_2 + 6.039783458388755 \cdot x_1^6.\]

A.6 Invariant in Sec. 6.6

Computed invariant:

\[-0.329552584068616 + 0.3916113015966564 \cdot x_3 + 0.2899739413846004 \cdot x_2^2 + 0.1428297229913382 \cdot x_1^2 - 0.871872297547406 - 0.21177461319034813 \cdot x_3^2 + 0.10501545362584444 \cdot x_2^2 + 0.4289078411249493 \cdot x_1^2 - 0.8535189933667975 \cdot 0.17888097824420462 \cdot x_3 + 0.007807118789072992 \cdot x_2^2 + 0.489332168912969 \cdot x_1^2 - 0.865051279681401 \cdot 0.2000960676408763 \cdot x_3 + 0.0877118073301994 \cdot x_2^2 + 0.45161541013547407 \cdot x_1^2 - 0.6906501877035409 \cdot 0.6121202891787498 \cdot x_3^2 + 0.0030550649675006266 \cdot x_2^2 + 0.38509964473994507 \cdot x_1^2 - 0.3329552584068616 + 0.9316113015966564 \cdot x_3^2 + 0.02899739413846004 \cdot x_2^2 + 0.1428297229913382 \cdot x_1^2 - 0.3337674127559513 + 0.9332427750444002 \cdot x_3^2 - 0.0032308722117315894 \cdot x_2^2 + 0.13381865590657924 \cdot x_1^2 - 0.3330753280420715 + 0.9321483462528075 \cdot x_3^2 - 0.002287385488984797 \cdot x_2^2 + 0.14196849744707027 \cdot x_1^2.\]

Invariant after removing all but one polynomial:

\[-0.329552584068616 + 0.9316113015966564 \cdot x_3 + 0.2899739413846004 \cdot x_2^2 + 0.1428297229913382 \cdot x_1^2.\]
A.7 Invariants in Sec. 6.7

Computed invariant for mode 1:

Not shown because too long.

Computed invariant for mode 2:

-0.9834456725348417 + 0.112332316310927176 + x2^2 + 0.025778192014470788 + x1 + x2 + 0.1398269304006194 + x1^2
-0.9834072352067391 + 0.11296145964890551 + x2^2 + 0.0269975265812748 + x1 + x2 + 0.13935434423717987 + x1^2
-0.9834072342091401 + 0.11295935658755152 + x2^2 + 0.02699537509948993 + x1 + x2 + 0.1393614690687895 + x1^2
-0.9850066759345267 + 0.11835293336418753 + x2^2 + 0.03755265898642679 + x1 + x2 + 0.11976739678040782 + x1^2
-0.9842843646648725 + 0.1157526692019295 + x2^2 + 0.0332808902290435 + x1 + x2 + 0.12937189415352954 + x1^2.

Computed invariant for mode 3:

-0.9817527300995846 + 0.141268427788666962 + x2^2 + 0.0221130626628126854 + x1 + x2 + 0.1253627657646799704 + x1^2
-0.9430521512391482 + 0.1274811491934949 + x2^2 + 0.2829021209811443 + x1 + x2 + 0.1198546863031727 + x1^2
-0.95675273017751 + 0.06963989317132586 + x2^2 + 0.24338182490184636 + x1 + x2 + 0.122229685818302 + x1^2
-0.9460438614272869 + 0.11497499259730252 + x2^2 + 0.2780506258895415 + x1 + x2 + 0.12028970353040660 + x1^2
-0.9447551218522421 + 0.12013698286746648 + x2^2 + 0.2803195319354359 + x1 + x2 + 0.1201075563678729 + x1^2
-0.9664456245483404 + 0.04994292733097908 + x2^2 + 0.21980562893445577 + x1 + x2 + 0.12318297077806359 + x1^2
-0.9715222141302825 + 0.1495897116723925 + x2^2 + 0.1357599376728176 + x1 + x2 + 0.123841611229666077 + x1^2
-0.96537970478111981 + 0.1389581820496891 + x2^2 + 0.1833301507589829 + x1 + x2 + 0.12297440815348881 + x1^2
-0.957136169395247 + 0.12862903115612198 + x2^2 + 0.22913553228295144 + x1 + x2 + 0.12182706167771154 + x1^2
-0.9401064342228134 + 0.14261960375083502 + x2^2 + 0.28564367648818584 + x1 + x2 + 0.1194451494775421 + x1^2.

A.8 Invariants in Sec. 6.8

Computed invariant for mode 1:

-0.9857936879547902 + 0.09909691865399034 + x2^2 + 0.000387618927225279 + x1 + x2 + 0.13561141269578034 + x1^2
-0.986140544056003 + 0.0972059879302515 + x2^2 + 0.000327492093934748 + x1 + x2 + 0.13441970678087045 + x1^2
-0.9847674411170849 + 0.10467633738192324 + x2^2 + 0.01082587540543865 + x1 + x2 + 0.13841514270030641 + x1^2,

Computed invariant for mode 2:

-0.9834456725348417 + 0.112332316310927176 + x2^2 + 0.025778192014470788 + x1 + x2 + 0.1398269304006194 + x1^2
-0.9834072352067391 + 0.11296145964890551 + x2^2 + 0.0269975265812748 + x1 + x2 + 0.13935434423717987 + x1^2
-0.9834072342091401 + 0.11295935658755152 + x2^2 + 0.02699537509948993 + x1 + x2 + 0.1393614690687895 + x1^2
-0.9850066759345267 + 0.11835293336418753 + x2^2 + 0.03755265898642679 + x1 + x2 + 0.11976739678040782 + x1^2
-0.9842843646648725 + 0.1157526692019295 + x2^2 + 0.0332808902290435 + x1 + x2 + 0.12937189415352954 + x1^2.

Computed invariant for mode 3:

-0.9902072381403451 + 0.13563419510408165 + x2^2 + 0.018812787726133103 + x1 + x2 + 0.0271860531884196 + x1^2.