

Polynomial Template Generation using Sum-of-Squares Programming

Assalé Adjé^{1,a} and Victor Magron^{2,b}

¹ Onera, the French Aerospace Lab, France.
Université de Toulouse, F-31400 Toulouse, France.
assale.adje@onera.fr

² Circuits and Systems Group, Department of Electrical and Electronic Engineering,
Imperial College London, South Kensington Campus, London SW7 2AZ, UK.
v.magron@imperial.ac.uk

Abstract. Template abstract domains allow to express more interesting properties than classical abstract domains. However, template generation is a challenging problem when one uses template abstract domains for program analysis. In this paper, we relate template generation with the program properties that we want to prove. We focus on one-loop programs with nested conditional branches. We formally define the notion of well-representative template basis with respect to such programs and a given property. The definition relies on the fact that template abstract domains produce inductive invariants. We show that these invariants can be obtained by solving certain systems of functional inequalities. Then, such systems can be strengthened using a hierarchy of sum-of-squares (SOS) problems when we consider programs written in polynomial arithmetic. Each step of the SOS hierarchy can possibly provide a solution which in turn yields an invariant together with a certificate that the desired property holds. The interest of this approach is illustrated on nontrivial program examples in polynomial arithmetic.

Keywords: static analysis, abstract interpretation, template abstract domains, sum-of-squares programming, piecewise discrete-time polynomial systems

1 Introduction

The concept of templates was introduced in a linear setting. They answered to the computational issue of the polyhedra domain, that is, the number of faces and the number of vertices both explode when performing the code analysis. Recently, generalizations of linear templates appeared, such as quadratic Lyapunov functions as nonlinear templates. Nevertheless, no precise characterization of the templates to use have been developed for program analysis purpose. Indeed, depending on the property to show, prefixing a template basis without any

^a The author is supported by the RTRA /STAE Project BRIEFCASE and the ANR ASTRID VORACE Project.

^b The author is supported by EPSRC (EP/I020457/1) Challenging Engineering Grant.

rules can lead to unuseful information on the programs. For instance, suppose that we want to show that the values taken by the variables of the program are bounded. Then, it is natural to use intervals or norm functions as templates. Unfortunately, these functions are not sufficient to show the desired property. In the context of linear systems in optimal control, it is well known that Lyapunov functions provide useful templates to bound the variable values. This result can be extended to polynomial systems using polynomial Lyapunov functions. The crucial notion behind is that these polynomial functions allow to define sublevel sets which are invariant by the dynamics -in our case, the dynamics being the loop body. In static analysis, Lyapunov functions provide inductive invariants, which are precisely the results of computation while using template abstract domains.

Related works. Template domains were introduced by Sankaranarayanan et al. [SSM05], see also [SCSM06]. The latter authors only considered a finite set of linear templates and did not provide an automatic method to generate templates. Linear template domains were generalized to nonlinear quadratic cases by Adjé et al. in [AGG11,AGG10], where the authors used in practice quadratic Lyapunov templates for affine arithmetic programs. These templates are again not automatically generated. Roux et al. [RJGF12] provide an automatic method to compute floating-point certified Lyapunov functions of perturbed affine loop body updates. They use Lyapunov functions with squares of coordinate functions as quadratic template bases in case of single loop programs written in affine arithmetic. The extension proposed in [AGMW13,AGMW14] relies on combining polynomial templates with sum-of-squares (SOS) techniques to certify nonlinear inequalities.

Proving polynomial inequalities is already NP-hard and boils down to show that the infimum of a given polynomial is positive. However, one can obtain lower bounds of the infimum by solving a hierarchy of Moment-SOS relaxations, introduced by Lasserre in [Las01]. Recent advances in SOS optimization allowed to extensively apply these relaxations to various fields, including parametric polynomial optimization, optimal control, combinatorial optimization, *etc.* (see e.g. [Par03,Lau09] for more details). In the context of hybrid systems, certified inductive invariants can be computed by using SOS approximations of parametric polynomial optimization problems [LWYZ14]. In [PJ04], the authors develop an SOS-based methodology to certify that the trajectories of hybrid systems avoid an unsafe region. Recently, Ahmadi et al. [AJ13] investigate necessary or sufficient conditions for SOS-convex Lyapunov functions to stabilize switched systems, either in the linear case or when the switched system is the convex hull of a finite number of nonlinear criteria.

In a static analysis context, polynomial invariants appear in [BRCZ05], where invariants are given by polynomial inequalities (of bounded degree) but the method relies on a reduction to linear inequalities (the polyhedra domain). Template polyhedra domains allow to analyze reachability for polynomial systems: in [STDG12], the authors propose a method that computes linear templates to improve the accuracy of reachable set approximations, whereas the

procedure in [DT12] relies on Bernstein polynomials and linear programming, with linear templates being fixed in advance. Bernstein polynomials also appear in [RG13] as template polynomials but there are not generated automatically. In [SG09], the authors use SMT-based techniques to automatically generate templates which are defined as formulas built with arbitrary logical structures and predicate conjunctions. Other reductions to systems of polynomial *equalities* (by contrast with polynomial inequalities, as we consider here) were studied in [MOS04,RCK07] and more recently in [CJJK14].

Contribution and methodology. In this paper, we generate polynomial templates by combining the approach of SOS approximations extensively used in control theory with template abstract domains originally introduced in static analysis. We focus on analyzing programs composed of a single loop with polynomial conditional branches in the loop body and polynomial assignments. For such programs, our method consists in computing certificates which yield sufficient conditions that a given property holds. We introduce the notion of *well-representative* templates with respect to this property. Computing inductive invariant and polynomial templates boils down to solving a system of functional inequalities. For computational purpose, we strengthen this system as follows:

1. We impose that the functions involved in each inequality of the system belong to a convex cone \mathcal{K} included in the set of nonnegative functions. This allows in turn to define the stronger notion of \mathcal{K} *well-representative* templates.
2. Instantiating \mathcal{K} to the cone of SOS polynomials leads to consider a hierarchy of SOS programs, parametrized by the degrees of the polynomial templates. While solving the hierarchy, we extract polynomial template bases and feasible invariant bounds together with (SOS-based) certificates that the desired property holds.

The potential of the method is demonstrated on several “toy” nonlinear programs, defined with medium-size polynomial conditionals/assignments, involving at most 4 variables and of degree up to 3. Numerical experiments illustrate the hardness of program analysis in this context, as simple nonlinear examples can already yield unexpected behaviors.

Organization of the paper. The paper is organized as follows. In Section 2, we present the programs that we want to analyze and their representation as constrained piecewise discrete-time dynamical system. Next, we recall the collecting semantics that we use and finally remind some required background about abstract semantics for generalized template domains. Section 3 contains the main contribution of the paper, namely the definition of well representative templates and how to generate such templates in practice using SOS programming. Section 4 provides practical computation examples for program analysis.

2 Static analysis context and abstract template domains

In this section, we describe the programs which are considered in this paper. Next, we explain how to analyze them through their representation as discrete-

time dynamical systems. Then, we give details about the special properties which can be inferred on such programs. Finally, we recall mandatory results for abstract template domains that are used in the sequel of the paper.

2.1 Program syntax and constrained piecewise discrete-time dynamical system representations

In this paper, we are interested in analyzing computer science programs. We focus on programs composed of a single loop with a possibly complicated switch-case type loop body. This loop is supposed to be written as a nested sequence of *if* statements. Moreover we suppose that the analyzed programs are written in Static Single Assignment (SSA) form, that is each variable is initialized at most once. We denote by (x_1, \dots, x_d) the vector of the program variables. Finally, we consider assignments of variables using only *parallel assignments* $(x_1, \dots, x_d) = T(x_1, \dots, x_d)$. Tests are either weak inequalities $r(x_1, \dots, x_d) \leq 0$ or strict inequalities $r(x_1, \dots, x_d) < 0$. We assume that assignments are functions from \mathbb{R}^d to \mathbb{R}^d and test functions are functions from \mathbb{R}^d to \mathbb{R} . In the program syntax, the notation \ll will be either \leq or $<$. The form of the analyzed program is described in Figure 1.

```

x ∈ Xin;
while (r10(x) << 0 and ... and rn00(x) << 0) {
  if (r11(x) << 0) {
    ⋮
    if (rn11(x) << 0) {
      x = T1(x);
    }
    else {
      ⋮
      if (rnii(x) << 0) {
        x = Ti(x);
      }
    }
  }
  else {
    ⋮
  }
}

```

Fig. 1. One-loop programs with nested conditional branches

As depicted in Figure 1, an update $T^i : \mathbb{R}^d \rightarrow \mathbb{R}^d$ of the i -th condition branch is executed if and only if the conjunction of tests $r_j^i(x) \ll 0$ holds.

The variable x is updated by $T^i(x)$ if the current value of x belongs to $X^i := \{x \in \mathbb{R}^d \mid \forall j = 1, \dots, n_i, r_j^i(x) \ll 0\}$. Consequently, we interpret programs as *constrained piecewise discrete-time dynamical systems* (CPDS for short). The term *piecewise* means that there exists a partition $\{X^i, i \in \mathcal{I}\}$ of \mathbb{R}^d such that for all $i \in \mathcal{I}$, the dynamics of the system is represented by the following relation, for $k \in \mathbb{N}$:

$$\text{if } x_k \in X^i \cap X^0, x_{k+1} = T^i(x_k). \quad (1)$$

We assume that the initial condition x_0 belongs to some compact set X^{in} . For the program, X^{in} is the set where the variables are supposed to be initialized in. Since the test entry for the loop condition can be nontrivial, we add the term *constrained* and X^0 denotes the set representing the conjunctions of tests for the loop condition. The iterates of the CPDS are constrained to live in X^0 : if for some step $k \in \mathbb{N}$, $x_k \notin X^0$ then the CPDS is stopped at this iterate with the terminal value x_k . We define a partition as a family of nonempty sets such that:

$$\bigcup_{i \in \mathcal{I}} X^i = \mathbb{R}^d, \forall i, j \in \mathcal{I}, i \neq j, X^i \cap X^j \neq \emptyset. \quad (2)$$

From Equation (2), for all $k \in \mathbb{N}^*$ there exists a unique $i \in \mathcal{I}$ such that $x_k \in X^i$. A set X^i can contain both strict and weak inequalities and characterizes the set of the n_i conjunctions of tests functions r_j^i . Let $r^i = (r_1^i, \dots, r_{n_i}^i)$ stands for the vector of tests functions associated to the set X^i . Moreover, for X^i , we denote by $r^{i,s}$ (resp. $r^{i,w}$) the part of r^i corresponding to strict (resp. weak) inequalities. Finally, we obtain the representation of the set X^i given by Equation (3):

$$X^i = \{x \in \mathbb{R}^d \mid r^{i,s}(x) < 0, r^{i,w}(x) \leq 0\}. \quad (3)$$

We insist on the notation: $y < z$ (resp. $y_l < z_l$) means that for all coordinates l , $y_l < z_l$ (resp. $y_l \leq z_l$).

We suppose that the sets X^{in} and X^0 also admits the representation given by Equation (3) and we denote by r^0 the vector of tests functions $(r_1^0, \dots, r_{n_0}^0)$ and by r^{in} the vector of tests functions $(r_1^{\text{in}}, \dots, r_{n_{\text{in}}}^{\text{in}})$. We also decompose r^0 and r^{in} as strict and weak inequality parts denoted respectively by $r^{0,s}$, $r^{0,w}$, $r^{\text{in},s}$ and $r^{\text{in},w}$. To sum up, we give a formal definition of CPDS.

Definition 1 (CPDS). *A constrained piecewise discrete-time dynamical system (CPDS) is the quadruple $(X^{\text{in}}, X^0, \mathcal{X}, \mathcal{L})$ with:*

- $X^{\text{in}} \subseteq \mathbb{R}^d$ is the compact of the possible initial conditions;
- $X^0 \subseteq \mathbb{R}^d$ is the set of the constraints which must be respected by the state variable;
- $\mathcal{X} := \{X^i, i \in \mathcal{I}\}$ is a partition as defined in Equation (2);
- $\mathcal{L} := \{T^i, i \in \mathcal{I}\}$ is the family of the functions from \mathbb{R}^d to \mathbb{R}^d , w.r.t. the partition \mathcal{X} satisfying Equation (1).

From now on, we associate a CPDS representation to each program of the form described at Figure 1. Since a program admits several CPDS representations,

we choose one of them, but this arbitrary choice does not change the results provided in this paper. In the sequel, we will often refer to the running example described in Example 1.

Example 1 (Running example). The program below involves four variables and contains an infinite loop with a conditional branch in the loop body. The update of each branch is polynomial. The parameters c_{ij} (resp. d_{ij}) are given parameters. During the analysis, we only keep the variables x_1 and x_2 since $oldx_1$ and $oldx_2$ are just memories.

```

 $x_1, x_2 \in [a_1, a_2] \times [b_1, b_2];$ 
 $oldx_1 = x_1;$ 
 $oldx_2 = x_2;$ 
while  $(-1 \leq 0)$  {
  if  $(oldx_1^2 + oldx_2^2 \leq 1)$  {
     $oldx_1 = x_1;$ 
     $oldx_2 = x_2;$ 
     $x_1 = c_{11} * oldx_1^2 + c_{12} * oldx_2^3;$ 
     $x_2 = c_{21} * oldx_1^3 + c_{22} * oldx_2^2;$ 
  }
  else {
     $oldx_1 = x_1;$ 
     $oldx_2 = x_2;$ 
     $x_1 = d_{11} * oldx_1^3 + d_{12} * oldx_2^2;$ 
     $x_2 = d_{21} * oldx_1^2 + d_{22} * oldx_2^2;$ 
  }
}

```

Its constrained piecewise discrete-time dynamical system representation corresponds to the quadruple $(X^{\text{in}}, X^0, \{X^1, X^2\}, \{T^1, T^2\})$, where the set of initial conditions is:

$$X^{\text{in}} = [a_1, a_2] \times [b_1, b_2],$$

the set X^0 in which the variable $x = (x_1, x_2)$ lies is:

$$X^0 = \mathbb{R}^d,$$

the partition verifying Equation (2) is:

$$X^1 = \{x \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1\}, \quad X^2 = \{x \in \mathbb{R}^2 \mid -x_1^2 - x_2^2 < -1\},$$

and the functions relative to the partition $\{X^1, X^2\}$ are:

$$T^1(x) = \begin{pmatrix} c_{11}x_1^2 + c_{12}x_2^3 \\ c_{21}x_1^3 + c_{22}x_2^2 \end{pmatrix} \text{ and } T^2(x) = \begin{pmatrix} d_{11}x_1^3 + d_{12}x_2^2 \\ d_{21}x_1^2 + d_{22}x_2^2 \end{pmatrix}.$$

2.2 Program invariants

The main goal of the paper is to decide automatically if a given property holds for the analyzed program. We are interested in numerical properties and more precisely in properties on the values taken by the d -uplet of the variables of the program. Hence, in our point-of-view, a property is just the membership of some set $P \subset \mathbb{R}^d$. In particular, we study properties which are valid after an arbitrary number of loop iterates. Such properties are called *loop invariants* of the program. Formally, we use the CPDS representation of a given program and we say that P is a loop invariant of this program if:

$$\forall k \in \mathbb{N}, x_k \in P,$$

where x_k is defined at Equation (1) as the state variable at step $k \in \mathbb{N}$ of the CPDS representation of the program.

Now, let us consider a program of the form described in Figure 1 and let us denote by \mathcal{S} the CPDS representation of this program. The set $\mathcal{R}(\mathcal{S})$ of *reachable values* is the set of all possible values taken by the state variable along the running of \mathcal{S} . We define $\mathcal{R}(\mathcal{S})$ as follows:

$$\mathcal{R}(\mathcal{S}) = \{y \in \mathbb{R}^d \mid \exists k \in \mathbb{N}, \exists i \in \mathcal{I}, x_k \in X^i \cap X^0, y = T^i(x_k)\} \cup X^{\text{in}}. \quad (4)$$

To prove that a set P is a loop invariant of the program is equivalent to prove that $\mathcal{R}(\mathcal{S}) \subseteq P$. We can rewrite $\mathcal{R}(\mathcal{S})$ by introducing auxiliary variables \mathcal{R}^i , $i \in \mathcal{I}$:

$$\mathcal{R}(\mathcal{S}) = \bigcup_{i \in \mathcal{I}} \mathcal{R}^i \cup X^{\text{in}}, \quad \mathcal{R}^i = T^i(\mathcal{R}(\mathcal{S}) \cap X^i \cap X^0). \quad (5)$$

Let us denote by $\wp(\mathbb{R}^d)$ the set of subsets of \mathbb{R}^d and introduce the map $F : (\wp(\mathbb{R}^d))^{|I|+1} \rightarrow (\wp(\mathbb{R}^d))^{|I|+1}$ defined by:

$$F_i(C_1, \dots, C_{|I|+1}) = \begin{cases} T^i(C_{|I|+1} \cap X^i \cap X^0) & \text{if } j \neq |I| + 1, \\ \bigcup_{k \in \mathcal{I}} C_k \cup X^{\text{in}} & \text{otherwise.} \end{cases} \quad (6)$$

We equip $\wp(\mathbb{R}^d)$ with the partial order of inclusion and $(\wp(\mathbb{R}^d))^{|I|+1}$ by the standard component-wise partial order. The infimum is understood in this sense i.e. as the greatest lower bound with respect to this order. The smallest fixed point problem is:

$$\inf \left\{ \mathbf{C} = (C_1, \dots, C_{|I|+1}) \in (\wp(\mathbb{R}^d))^{|I|+1} \mid \forall i = 1, \dots, |I| + 1, C_i = F_i(\mathbf{C}) \right\}.$$

It is well-known from Tarski's theorem that the solution of this problem exists, is unique and in this case, it corresponds to $(\mathcal{R}^1, \mathcal{R}^2, \dots, \mathcal{R}(\mathcal{S}))$ where $\mathcal{R}^1, \mathcal{R}^2, \dots, \mathcal{R}^{|I|}$ are defined in Equation (5). Tarski's theorem also states that $(\mathcal{R}^1, \mathcal{R}^2, \dots, \mathcal{R}(\mathcal{S}))$ is the smallest solution of the following Problem:

$$\inf \left\{ \mathbf{C} = (C_1, \dots, C_{|I|+1}) \in (\wp(\mathbb{R}^d))^{|I|+1} \mid \forall i = 1, \dots, |I| + 1, F_i(\mathbf{C}) \subseteq C_i \right\}.$$

We warn the reader that the construction of F is completely determined by the data of the CPDS \mathcal{S} . But for the sake of conciseness, we do not make it explicit on the notations. Note also that the map F corresponds to a standard transfer function (or collecting semantics functional) applied to the CPDS representation of a program.

Example 2 (Transfer function of the running example). Since $X^0 = \mathbb{R}^d$, the transfer function F associated to the CPDS of Example 1 is given by:

$$\begin{aligned} F_1(C_1, C_2, C_3) &= T^1(C_3 \cap X^1), \\ F_2(C_1, C_2, C_3) &= T^2(C_3 \cap X^2), \\ F_3(C_1, C_2, C_3) &= C_1 \cup C_2 \cup X^{\text{in}}. \end{aligned}$$

To prove that a subset P is a loop invariant, it suffices to show that $\mathbf{P} = (T^1(P \cap X^1 \cap X^0), \dots, P)$ satisfies $F_{|\mathcal{I}|+1}(\mathbf{P}) \subseteq P$. Nevertheless, F is still not computable and we use abstract interpretation [CC77] to provide safe over-approximations of F . Next, we use generalized abstract template domains as abstract domains and we construct a safe over-approximation of F using a Galois connection. In this paper, we consider invariants defined from properties which are encoded with sublevel sets of given functions. A loop invariant is supposed to be the union of sublevel sets of a given function from \mathbb{R}^d to \mathbb{R} .

Definition 2 (Sublevel property). *Given a function κ from \mathbb{R}^d to \mathbb{R} , we define the sublevel property \mathcal{P}_κ as follows:*

$$\mathcal{P}_\kappa := \bigcup_{\alpha \in \mathbb{R}} \{x \in \mathbb{R}^d \mid \kappa(x) \leq \alpha\}.$$

Example 3 (Sublevel property examples).

1. Let κ be a norm on \mathbb{R}^d , then \mathcal{P}_κ is the property “the values taken by the variables are bounded”.
2. Let $\kappa : x \mapsto x_i$, then \mathcal{P}_κ is the property “the values taken by the variable x_i are bounded from above”.
3. We can ensure that the set of possible values taken by the program variables avoids an unsafe region with a fixed level sublevel property. For example, if the property to show consists in proving that the square norm of the variable is still greater than 1, we can set $\kappa(x) = 1 - \|x\|_2^2$ and restrict the sublevel sets to those for which $\alpha \leq 0$.

A sublevel property is called *sublevel invariant* when this property is a loop invariant. We describe how to construct template bases, so that we can prove that a sublevel property is a sublevel invariant.

2.3 Abstract template domains

The concept of generalized templates was introduced in [AGG10,AGG11]. Let $\mathbf{F}(\mathbb{R}^d, \mathbb{R})$ stands for the set of functions from \mathbb{R}^d to \mathbb{R} .

Definition 3 (Generalized templates). A generalized template p is a function from \mathbb{R}^d to \mathbb{R} over the vector of variables (x_1, \dots, x_d) .

Templates can be viewed as implicit functional relations on variables to prove certain properties on the analyzed program. We denote by \mathbb{P} the set of templates. First, we suppose that \mathbb{P} is given by some oracle and say that \mathbb{P} forms a template basis. Here, we recall the required background about generalized templates (see [AGG10,AGG11] for more details).

Basic notions We replace the classical concrete semantics by meaning of sublevel sets i.e. we have a functional representation of numerical invariants through the functions of \mathbb{P} . An invariant is determined as the intersection of sublevel sets. The problem is thus reduced to find optimal level sets on each template p . Let $\mathbf{F}(\mathbb{P}, \overline{\mathbb{R}})$ stands for the set of functions from \mathbb{P} to $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$.

Definition 4 (\mathbb{P} -sublevel sets). For $w \in \mathbf{F}(\mathbb{P}, \overline{\mathbb{R}})$, we associate the \mathbb{P} -sublevel set $w^* \subseteq \mathbb{R}^d$ given by:

$$w^* = \{x \in \mathbb{R}^d \mid p(x) \leq w(p), \forall p \in \mathbb{P}\} = \bigcap_{p \in \mathbb{P}} \{x \in \mathbb{R}^d \mid p(x) \leq w(p)\}.$$

In convex analysis, a closed convex set can be represented by its support function i.e. the supremum of linear forms on the set (e.g. [Roc96, § 13]). Here, we use the generalization by Moreau [Mor70] (see also [Rub00,Sin97]) which consists in replacing the linear forms by the functions $p \in \mathbb{P}$.

Definition 5 (\mathbb{P} -support functions). To $X \subseteq \mathbb{R}^d$, we associate the abstract support function denoted by $X^\dagger : \mathbb{P} \mapsto \overline{\mathbb{R}}$ and defined by:

$$X^\dagger(p) = \sup_{x \in X} p(x).$$

Let C and D be two ordered sets equipped respectively by the order \leq_C and \leq_D . Let ψ be a map from C to D and φ be a map from D to C . We say that the pair (ψ, φ) defines a Galois connection between C and D if and only if ψ and φ are monotonic and the equivalence $\psi(c) \leq_D d \iff \varphi(d) \leq_C c$ holds for all $c \in C$ and all $d \in D$.

We equip $\mathbf{F}(\mathbb{P}, \overline{\mathbb{R}})$ with the partial order of real-valued functions i.e. $w \leq v \iff w(p) \leq v(p) \forall p \in \mathbb{P}$. The set $\varphi(\mathbb{R}^d)$ is equipped with the inclusion order.

Proposition 1. The pair of maps $w \mapsto w^*$ and $X \mapsto X^\dagger$ defines a Galois connection between $\mathbf{F}(\mathbb{P}, \overline{\mathbb{R}})$ and the set of subsets of \mathbb{R}^d .

In the terminology of abstract interpretation, $(\cdot)^\dagger$ is the abstraction function, and $(\cdot)^*$ is the concretisation function. The Galois connection result provides the correctness of the semantics. We also remind the following property:

$$(((w^*)^\dagger)^* = w^*, \quad ((X^\dagger)^*)^\dagger = X^\dagger. \quad (7)$$

The lattices of \mathbb{P} -convex sets and \mathbb{P} -convex functions Now, we are interested in closed elements (in term of Galois connection), called \mathbb{P} -convex elements.

Definition 6 (\mathbb{P} -convexity). Let $w \in \mathbf{F}(\mathbb{P}, \overline{\mathbb{R}})$, we say that w is a \mathbb{P} -convex function if $w = (w^*)^\dagger$. A set $X \subseteq \mathbb{R}^d$ is a \mathbb{P} -convex set if $X = (X^\dagger)^*$. We respectively denote by $\text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$ and $\text{Vex}_{\mathbb{P}}(\mathbb{R}^d)$ the set of \mathbb{P} -convex functions of $\mathbf{F}(\mathbb{P}, \overline{\mathbb{R}})$ and the set of \mathbb{P} -convex sets of \mathbb{R}^d .

The family of functions $\text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$ is ordered by the partial order of real-valued functions. The family of sets $\text{Vex}_{\mathbb{P}}(\mathbb{R}^d)$ is ordered by the inclusion order. Galois connection allows to construct lattice operations on \mathbb{P} -convex elements.

Definition 7 (The meet and join). Let v and w be in $\mathbf{F}(\mathbb{P}, \overline{\mathbb{R}})$. We denote by $\inf(v, w)$ and $\sup(v, w)$ the functions defined respectively by, $p \mapsto \inf(v(p), w(p))$ and $p \mapsto \sup(v(p), w(p))$. We equip $\text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$ with the join operator $v \vee w = \sup(v, w)$ and the meet operator $v \wedge w = (\inf(v, w)^*)^\dagger$. Similarly, we equip $\text{Vex}_{\mathbb{P}}(\mathbb{R}^d)$ with the join operator $X \sqcup Y = ((X \cup Y)^\dagger)^*$ and the meet operator $X \sqcap Y = X \cap Y$.

The next theorem follows readily from the fact that the pair of $v \mapsto v^*$ and $C \mapsto C^\dagger$ defines a Galois connection (see e.g. [DP02, § 7.27]).

Theorem 1. The complete lattices $(\text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}}), \wedge, \vee)$ and $(\text{Vex}_{\mathbb{P}}(\mathbb{R}^d), \sqcap, \sqcup)$ are isomorphic.

Abstract semantics Since the pair of maps $w \mapsto w^*$ and $X \mapsto X^\dagger$ is a Galois connection (Proposition 1), we can construct abstract semantics functional from this pair and the map F defined at Equation (6). We obtain a map F^\sharp from $\text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})^{|\mathcal{I}|+1}$ to itself defined for $w \in \text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})^{|\mathcal{I}|+1}$ and $p \in \mathbb{P}$ by:

$$(F_i^\sharp(w))(p) = \begin{cases} \sup_{y \in T^i(w_{|\mathcal{I}|+1}^* \cap X^i \cap X^0)} p(y) = \sup_{\substack{x \in w_{|\mathcal{I}|+1}^* \\ r_s^i(x) < 0, r_w^i(x) \leq 0 \\ r_s^0(x) < 0, r_w^0(x) \leq 0}} p(T^i(x)) \\ \sup_{y \in \bigcup_{j \in \mathcal{I}} w_j^* \cup X^{\text{in}}} p(y) = \left(\bigcup_{j \in \mathcal{I}} w_j^* \cup X^{\text{in}} \right)^\dagger(p) \end{cases}$$

Since F is conditioned by the data of the CPDS \mathcal{S} , it is also the case for F^\sharp . As a corollary of Theorem 1, the best abstraction of $\mathcal{R}(\mathcal{S})$ in the lattice $\text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$ is the smallest fixed point of Equation (8).

$$\inf \left\{ \mathbf{w} = (w_1, \dots, w_{|\mathcal{I}|+1}) \in \text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})^{|\mathcal{I}|+1} \left\{ \begin{array}{l} \text{s. t. } \forall i = 1, \dots, |\mathcal{I}| + 1, F_i^\sharp(\mathbf{w}) \leq w_i. \end{array} \right. \right\} \quad (8)$$

The infimum is understood in the sense of the order of the component-wise order of the complete lattice $\text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})^{|\mathcal{I}|+1}$. Using Tarski's theorem, the solution of Equation (8) exists and is unique and is usually called the abstract semantics. This latter solution is optimal but any feasible solution could provide an answer to decide whether a sublevel property is an invariant of the program.

Definition 8 (Feasible invariant bound). *The function $w \in \text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$ is a feasible invariant bound w.r.t. to the CPDS $\mathcal{S} = (X^{\text{in}}, X^0, \{X^i, i \in \mathcal{I}\}, \{T^i, i \in \mathcal{I}\})$ iff it exists $(w_1, \dots, w_{|\mathcal{I}|}) \in \text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})^{|\mathcal{I}|}$ such that:*

$$w \geq \sup\{X^{\text{in}\dagger}, \sup_{i \in \mathcal{I}} w_i\} \wedge \left(\forall i \in \mathcal{I}, w_i \geq (T^i(w^* \cap X^i \cap X^0))^{\dagger} \right) \quad (9)$$

In the sequel, we denote by $\mathcal{F}(\mathcal{S})$ the set of feasible invariant bounds.

From the definition of feasible invariant bound, we state the following proposition.

Proposition 2. *Let us consider a CPDS $\mathcal{S} = (X^{\text{in}}, X^0, \{X^i, i \in \mathcal{I}\}, \{T^i, i \in \mathcal{I}\})$. The following statements are true:*

1. *Let $(w_1, \dots, w_{|\mathcal{I}|+1})$ be a solution of Problem (8), then $w_{|\mathcal{I}|+1}$ is the smallest feasible invariant bound w.r.t. \mathcal{S} ;*
2. *For all $w \in \mathcal{F}(\mathcal{S})$, $\mathcal{R}(\mathcal{S}) \subseteq w^*$.*

For a given program represented by the CPDS \mathcal{S} , we recall that an invariant $P \subseteq \mathbb{R}^d$ is to said be an *inductive invariant* of this program if for all $k \in \mathbb{N}$, the implication $x_k \in P \implies x_{k+1} \in P$ holds for the state variable x_k . Next, for a given function $w \in \text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$, we give a simple condition in term of inductive invariants (up to test functions) for w to be a feasible invariant bound.

Proposition 3 (Loop head invariants in template domains). *Let us consider the CPDS $\mathcal{S} = (X^{\text{in}}, X^0, \{X^i, i \in \mathcal{I}\}, \{T^i, i \in \mathcal{I}\})$ and $w \in \text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$. Suppose that:*

$$X^{\text{in}} \subseteq w^* \wedge (\forall i \in \mathcal{I} (x \in w^* \wedge x \in X^i \wedge x \in X^0 \implies T^i(x) \in w^*)) . \quad (10)$$

Then $w \in \mathcal{F}(\mathcal{S})$.

Proof. From the definition of the $(\cdot)^{\dagger}$ operator and Proposition 1, Conjunction (9) holds with $w_i = w$ for all $i \in \mathcal{I}$. \square

We recalled that abstract template domains produce invariants, i.e. \mathbb{P} -sublevel sets of feasible invariant bounds. It is not surprising since abstract template domains are abstract domains. The main issue is that \mathbb{P} is supposed to be given. The question is which templates basis \mathbb{P} can produce a nontrivial (strictly smaller than \mathbb{R}^d) feasible invariant bound? This question can be refined when we want to show that some sublevel property is an invariant: which templates basis can ensure that the sublevel property is an invariant of the program? We propose an answer by considering Equation (10) as a system of equations, where unknowns are the template basis \mathbb{P} and $w \in \text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$. Given a sublevel \mathcal{P}_{κ} , we also impose that w and \mathbb{P} satisfy $w^* \subseteq \mathcal{P}_{\kappa}$. This latter constraint leads to the computation of a level α for which $\{x \in \mathbb{R}^d \mid \kappa(x) \leq \alpha\}$ is an invariant of the program.

3 Proving program properties using sum-of-squares

Here, we describe how to certify that a sublevel property is a loop invariant using sum-of-squares (SOS) approximations. In Section 3.1, we provide a formal definition of the set of template bases that we shall use to the latter certification. Then we describe how to construct template bases so that we can prove sublevel properties (Section 3.2). In the end, we explain how to compute such bases in practice, by solving a hierarchy of SOS programs (Section 3.3).

3.1 The general setting

Definition 9 (Well-representative template basis w.r.t. a CPDS and a sublevel property). *Let \mathcal{P}_κ be a sublevel property and $\mathcal{S} = (X^{\text{in}}, X^0, \{X^i, i \in \mathcal{I}\}, \{T^i, i \in \mathcal{I}\})$ be a CPDS. The template basis \mathbb{P} is well-representative w.r.t. \mathcal{S} and \mathcal{P}_κ iff there exists $w \in \mathcal{F}(\mathcal{S})$ such that $w^* \subseteq \mathcal{P}_\kappa$.*

In the sequel, we fix a CPDS $\mathcal{S} = (X^{\text{in}}, X^0, \{X^i, i \in \mathcal{I}\}, \{T^i, i \in \mathcal{I}\})$ and a sublevel property \mathcal{P}_κ .

Well-representative template bases explicit the sets of implicit functional relations on the program variables, needed to prove that a sublevel property is an invariant. Next, we define a cone structure to strengthen the notion of well-representative bases.

Definition 10 (Convex cones containing the scalars in $\mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$). *A non-empty subset \mathcal{K} of $\mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$ is a convex cone containing the scalars iff:*

1. for all $f \in \mathcal{K}$, for all $t \geq 0$, $tf \in \mathcal{K}$;
2. for all $f, g \in \mathcal{K}$, $f + g \in \mathcal{K}$;
3. for all $c \in \mathbb{R}_+$, $x \mapsto c \in \mathcal{K}$;

In the sequel, we write $c \in \mathcal{K}$ instead of $x \mapsto c \in \mathcal{K}$, for each $c \in \mathbb{R}_+$. For a convex cone containing the scalars \mathcal{K} , \mathcal{K}^k stands for the set of vectors of k elements of \mathcal{K} and $\mathcal{K}^{n \times k}$ stands for the set of tableaux of $n \times k$ elements of \mathcal{K} . For $\lambda \in \mathcal{K}^{n \times k}$, we denote the “row m ” of λ by $\lambda_{m,\cdot}$ and the “column j ” of λ by $\lambda_{\cdot,j}$. Thus $\lambda_{m,j}$ refers to the m, j element of the tableau λ .

We derive a stronger notion of well-representative template bases, namely \mathcal{K} well-representative template bases. This notion is more restrictive, as a \mathcal{K} well-representative template basis deals with a system of inequalities instead of conjunctions of implications.

Definition 11 (\mathcal{K} well-representative template basis). *A finite template basis $\mathbb{P} = \{p_1, \dots, p_k\}$ is a \mathcal{K} well-representative template basis w.r.t. \mathcal{S} and \mathcal{P}_κ iff there exist $w \in \mathbb{R}^k$, $\alpha \in \mathbb{R}$, $\nu \in \mathcal{K}^k$ and for all $i \in \mathcal{I}$, there exist $\lambda^i \in \mathcal{K}^{k \times k}$, $\mu^i \in \mathcal{K}^{k \times n_i}$, $\gamma^i \in \mathcal{K}^{k \times n_0}$ such that:*

1. Initial condition satisfiability: $\forall l = 1, \dots, k$,

$$w_l \geq \sup_{y \in X^{\text{in}}} p_l(y).$$

2. “Local” branch satisfiability: $\forall l = 1, \dots, k, \forall i \in \mathcal{I}$:

$$w_l - \sum_{j=1}^k \lambda_{l,j}^i(x)(w_j - p_j(x)) - p_l(T^i(x)) + \sum_{j=1}^{n_i} \mu_{l,j}^i(x)r_j^i(x) + \sum_{j=1}^{n_0} \gamma_{l,j}^i(x)r_j^0(x) \in \mathcal{K}.$$

3. Property satisfiability:

$$\alpha - \kappa(x) - \sum_{t=1}^k \nu_t(x)(w_t - p_t(x)) \in \mathcal{K}.$$

For the sake of presentation, let us define for all $l = 1, \dots, k$, for all $i \in \mathcal{I}$:

$$\begin{cases} S_l^i : x \mapsto \\ w_l - \sum_{j=1}^k \lambda_{l,j}^i(x)(w_j - p_j(x)) - p_l(T^i(x)) + \sum_{j=1}^{n_i} \mu_{l,j}^i(x)r_j^i(x) + \sum_{j=1}^{n_0} \gamma_{l,j}^i(x)r_j^0(x), \\ S^\kappa : x \mapsto \alpha - \kappa(x) - \sum_{t=1}^k \nu_t(x)(w_t - p_t(x)). \end{cases} \quad (11)$$

Example 4 (\mathcal{K} well-representative template basis). Consider Example 1. We are interested in proving the boundedness of the values taken by the variables of the program. For $x = (x_1, x_2)$, let consider $\kappa(x) = \|x\|_2^2 = x_1^2 + x_2^2$. Recall that $X^{\text{in}} = [a_1, a_2] \times [b_1, b_2]$, $X^0 = \mathbb{R}^d$, $X^1 = \{x \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1\}$, $X^2 = \{x \in \mathbb{R}^2 \mid -x_1^2 - x_2^2 < -1\}$, $T^1(x_1, x_2) = (c_{11}x_1^2 + c_{12}x_2^3, c_{21}x_1^3 + c_{22}x_2^2)$ and $T^2(x_1, x_2) = (d_{11}x_1^3 + d_{12}x_2^2, d_{21}x_1^2 + d_{22}x_2^2)$. Let $\mathcal{K} = \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$ and $\{p\}$ be a singleton template basis. Then $\{p\}$ is \mathcal{K} well-representative w.r.t. the CPDS $(X^{\text{in}}, X^0, \{X^1, X^2\}, \{T^1, T^2\})$ and \mathcal{P}_κ iff there exists $w \in \mathbb{R}$, $\alpha \in \mathbb{R}_+$, $\nu \in \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$, $\lambda^1, \lambda^2 \in \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$ and $\gamma^1, \gamma^2 \in \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$ such that:

$$\begin{cases} w \geq \sup_{y \in [a_1, a_2] \times [b_1, b_2]} p(y), \\ \forall x \in \mathbb{R}^2, w - \lambda^1(x)(w - p(x)) - p(T^1(x)) + \gamma^1(x)(\|x\|_2^2 - 1) \geq 0, \\ \forall x \in \mathbb{R}^2, w - \lambda^2(x)(w - p(x)) - p(T^2(x)) + \gamma^2(x)(1 - \|x\|_2^2) \geq 0, \\ \forall x \in \mathbb{R}^2, \alpha - \|x\|_2^2 - \nu(x)(w - p(x)) \geq 0. \end{cases}$$

Note that generating inductive invariants is well known to yield undesirable non-linear optimization problems (e.g. bilinearity, as in [CSS03]). Here nonlinearity is avoided by fixing the parameters $\{\lambda^i, i \in \mathcal{I}\} \subseteq \mathcal{K}^{k \times k}$ and $\nu \in \mathcal{K}^k$ to 1, so that the two last inequalities of Definition 11 become linear in the variables $p_1, \dots, p_k, w_1, \dots, w_k, \alpha$ and the parameters $\{\mu^i, i \in \mathcal{I}\}, \{\gamma^i, i \in \mathcal{I}\} \in \mathcal{K}^k$.

The next lemma states that \mathcal{K} well-representative templates bases are well-representative template bases. This result is an application of S-Lemma with “nonnegative functions multipliers”.

Lemma 1 (Functional S-Lemma). *Let $\delta, \beta_1, \dots, \beta_n \in \mathbb{R}$ and $h, g_1, \dots, g_n \in \mathbf{F}(\mathbb{R}^d, \mathbb{R})$. If there exists $\lambda \in \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)^n$ such that*

$$\delta - h(x) - \sum_{i=1}^n \lambda_i(x)(\beta_i - g_i(x)) \geq 0, \quad (12)$$

then

$$\forall x \in \mathbb{R}^d, (g_1(x) \leq \beta_1 \wedge \dots \wedge g_n(x) \leq \beta_n \implies h(x) \leq \delta). \quad (13)$$

Proof. Assuming that the inequality (13) holds for some $\lambda \in \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)^n$, we obtain $\delta - h(x) \geq \sum_{i=1}^n \lambda_i(x)(\beta_i - g_i(x))$. The positivity of λ_i yields the desired result. \square

Theorem 2 (\mathcal{K} well-representative is well-representative). *Assume that a finite template basis \mathbb{P} is \mathcal{K} well-representative w.r.t. \mathcal{S} and \mathcal{P}_κ . Then \mathbb{P} is well-representative w.r.t. \mathcal{S} and \mathcal{P}_κ .*

Proof. $\mathbb{P} = \{p_1, \dots, p_k\}$ is \mathcal{K} well-representative. Then there exists $w \in \mathbb{R}^k$, $\alpha \in \mathbb{R}$ and $\nu \in \mathcal{K}^k$ and for all $i \in \mathcal{I}$, $\lambda^i \in \mathcal{K}^{k \times k}$, $\mu^i \in \mathcal{K}^{k \times n_i}$, $\gamma^i \in \mathcal{K}^{k \times n_0}$ such that, for all $l = 1, \dots, k$, for all $i \in \mathcal{I}$, $S_l^i \in \mathcal{K}$, $S^\kappa \in \mathcal{K} \subseteq \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$ ($S_l^i \in \mathcal{K}$ and S^κ defined at Equation (11)) and $w_l \geq \sup\{p_l(x) \mid x \in X^{\text{in}}\}$. We set, for all $l = 1, \dots, k$, $v(p_l) := w_l$. From Proposition 1, $v(p_l) \geq \sup\{p_l(x) \mid x \in X^{\text{in}}\}$ for all $l = 1, \dots, k$ is equivalent to $X^{\text{in}} \subseteq v^*$ and $S_l^i \in \mathcal{K} \subseteq \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$ for all $l = 1, \dots, k$ and for all $i \in \mathcal{I}$ imply respectively, by Lemma 1 for all $i \in \mathcal{I}$, $(x \in v^* \wedge r^i(x) \leq 0 \wedge r^0(x) \leq 0 \implies T^i(x) \in v^*)$. Taking $\bar{v} = (v^*)^\dagger$, we have from Equation (7), $\bar{v} \in \text{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \bar{\mathbb{R}})$ and $\bar{v}^* = v^*$. By Proposition 3, $v \in \mathcal{F}(\mathcal{S})$. Finally $S^\kappa \in \mathcal{K} \subseteq \mathbf{F}(\mathbb{R}^d, \mathbb{R}_+)$ implies that $v^* \subseteq \{x \in \mathbb{R}^d \mid \kappa(x) \leq \alpha\} \subseteq \mathcal{P}_\kappa$ by Lemma 1. \square

This proof exhibits a feasible invariant bound which is given by the variable w of the system of inequalities in Definition 11.

3.2 Simple construction of \mathcal{K} well-representative template bases

In this subsection, we discuss how to simply construct \mathcal{K} well-representative template bases.

Proposition 4 (With one \mathcal{K} well-representative template). *Let $\{p\}$ be a \mathcal{K} well-representative template basis w.r.t. \mathcal{S} and \mathcal{P}_κ and \mathcal{Q} be a finite subset of $\mathbf{F}(\mathbb{R}^d, \mathbb{R})$ s.t. for all $q \in \mathcal{Q}$, $p - q \in \mathcal{K}$, for all $i \in \mathcal{I}$, $(p - q) \circ T^i \in \mathcal{K}$. Then $\mathbb{P} = \{p\} \cup \mathcal{Q}$ is a \mathcal{K} well-representative template basis w.r.t. \mathcal{S} and \mathcal{P}_κ .*

Proof. Suppose that $\{p\}$ is \mathcal{K} well-representative w.r.t. \mathcal{P}_κ . By definition, there exists $w \in \mathbb{R}$, $\alpha \in \mathbb{R}$ and $\nu \in \mathcal{K}$ and for all $i \in \mathcal{I}$, $\lambda^i \in \mathcal{K}$, $\mu^i \in \mathcal{K}^{1 \times n_i}$, $\gamma^i \in \mathcal{K}^{1 \times n_0}$, $\nu \in \mathcal{K}$ such that the functions for all $i \in \mathcal{I}$, $S^i := S_1^i$, S^κ belong to \mathcal{K} ($S_1^i \in \mathcal{K}$ and S^κ defined at Equation (11)) and $w \geq \sup\{p(x) \mid x \in X^{\text{in}}\}$. Let us take q such that $p - q \in \mathcal{K}$. It follows that $p \geq q$ and thus: $w \geq \sup\{p(x) \mid$

$x \in X^{\text{in}}\} \geq \sup\{q(x) \mid x \in X^{\text{in}}\}$. Now let $i \in \mathcal{I}$, since $(p - q) \circ T^i \in \mathcal{K}$ then there exists $f \in \mathcal{K}$ such that $f(x) = p(T^i(x)) - q(T^i(x))$ for all $x \in \mathbb{R}^d$, we have $w(1 - \lambda^i(x)) - q(T(x)) + \lambda^i(x)p(x) + \sum_{j=1}^{n_i} \mu_j^i(x)r_j^i(x) + \sum_{j=1}^{n_0} \gamma_j^i(x)r_j^0(x) = S^i(x) + f(x)$ for all $x \in \mathbb{R}^d$. Since \mathcal{K} is closed under addition then $S^i + f \in \mathcal{K}$. Now $S^\kappa \in \mathcal{K}$ implies that $S^\kappa + 0(w - q) \in \mathcal{K}$. It follows that $\{p, q\}$ is \mathcal{K} well-representative w.r.t. \mathcal{S} and \mathcal{P}_κ by taking $(w, w) \in \mathbb{R}^2$, $\alpha \in \mathbb{R}$, $(\nu, 0) \in \mathcal{K}^2$ and for all $i \in \mathcal{I}$, $\{(\lambda^i, 0), (\lambda^i, 0)\} \in \mathcal{K}^{2 \times 2}$, $(\mu^i, \mu^i) \in \mathcal{K}^{2 \times n_i}$, $(\gamma^i, \gamma^i) \in \mathcal{K}^{2 \times n_0}$ (following the order of the parameters of Definition 11). We conclude by induction on the elements q . \square

Example 5 (With Quadratic Lyapunov Functions). Let us consider the following program:

$$\begin{aligned} & x \in X^{\text{in}}; \\ & \text{while } (-1 \leq 0) \{ \\ & \quad x = Ax; \\ & \} \end{aligned}$$

where X^{in} is a bounded set, A is a $d \times d$ matrix. Its CPDS representation is $S = (X^{\text{in}}, \mathbb{R}^d, \mathbb{R}^d, Ax)$. Suppose there exists a symmetric matrix P such that:

$$P - \text{Id} \succeq 0 \quad P - A^\top P A \succeq 0 \quad (14)$$

where $B - C \succeq 0$ for two symmetric matrices means that $x^\top(B - C)x \geq 0$ for all x and Id is the identity matrix. Let $k = 1, \dots, d$ and let us denote by I_k the $d \times d$ matrix such that $I_k(i, j) = 1$ if $i = j = k$ and 0 otherwise. Remark that $\text{Id} - I_k \succeq 0$ for all $k = 1, \dots, d$.

Let $\mathcal{K} = \{x \mapsto x^\top Q x + c \mid c \in \mathbb{R}_+, Q \succeq 0\}$. Then $\mathbb{P} = \{x \mapsto x^\top P x\} \cup \{x \mapsto x^\top I_k x, k = 1, \dots, d\}$ is a \mathcal{K} well-representative template basis w.r.t. S and $\mathcal{P}_{\|\cdot\|_2^2}$.

We write $\beta := \sup\{x^\top P x \mid x \in X^{\text{in}}\} \in \mathbb{R}$ (since X^{in} is bounded and $x \mapsto x^\top P x$ is continuous). We have to exhibit $w, \alpha \in \mathbb{R}$ and $\lambda, \nu \in \mathcal{K}$ such that: $w \geq \beta$, $x \mapsto w - \lambda(x)(w - x^\top P x) - x^\top A^\top P A x \in \mathcal{K}$ and $x \mapsto \alpha - \|x\|_2^2 - \nu(x)(w - x^\top P x) \in \mathcal{K}$. Taking $\lambda = \nu = 1$ and $\alpha = w = \beta$, the latter inequalities become $P - A^\top P A x \succeq 0$ and $x \mapsto -\|x\|_2^2 + x^\top P x \geq 0$. So $-\|x\|_2^2 + x^\top P x = x^\top(P - \text{Id})x \in \mathcal{K}$. Thus, $\{x \mapsto x^\top P x\}$ is a \mathcal{K} well-representative template basis w.r.t. S and $\mathcal{P}_{\|\cdot\|_2^2}$. Now $P - \text{Id} \succeq 0$ implies that $P - I_k \succeq 0$ and then $x^\top P x - x^\top I_k x \in \mathcal{K}$. For all $k = 1, \dots, d$, for all $x \in \mathbb{R}^d$, $x^\top A^\top P A x - x^\top A^\top I_k A x = x^\top A^\top P - I_k A x \in \mathcal{K}$. By Proposition 4, a \mathcal{K} well-representative template basis w.r.t. S and $\mathcal{P}_{\|\cdot\|_2^2}$.

This example shows that the quadratic forms (Lyapunov functions for discrete-time linear systems) $x \mapsto x^\top P x$ for P satisfying Equation (14) combined with $x \mapsto x_k^2$ are used in the setting of quadratic templates.

Another possibility consists in constructing a \mathcal{K} well-representative template basis w.r.t. \mathcal{S} and \mathcal{P}_κ from a vector of templates p_1, \dots, p_k such that for all $i = 1, \dots, k$, $\{p_i\}$ is a \mathcal{K} well-representative templates w.r.t. \mathcal{S} and \mathcal{P}_κ (Proposition 5).

Proposition 5 (From two single \mathcal{K} well-representative templates). *Let $\{p\}$ and \mathcal{Q} two \mathcal{K} well-representative template bases w.r.t. \mathcal{S} and \mathcal{P}_κ . Then $\{p\} \cup \mathcal{Q}$ is a \mathcal{K} well-representative template basis w.r.t. \mathcal{S} and \mathcal{P}_κ .*

Proof. By induction, it suffices to prove the result for $\mathcal{Q} = \{q\}$. We write $p_1 = p$ and $p_2 = q$. By definition, for $l = 1, 2$, there exist $w_l \in \mathbb{R}$, $\alpha_l \in \mathbb{R}$, $\nu \in \mathcal{K}^k$ and for all $i \in \mathcal{I}$ $\lambda_l^i \in \mathcal{K}$, $\mu_l^i \in \mathcal{K}^{1 \times n_i}$, $\gamma_l^i \in \mathcal{K}^{1 \times n_0}$ such that $S_l^i, S_l^\kappa \in \mathcal{K}$ ($S_l^i \in \mathcal{K}$ and S^κ defined at Equation (11)) and $w_l \geq \sup\{p_l(x) \mid x \in X^{\text{in}}\}$. It follows that $\{p, q\}$ is \mathcal{K} well-representative w.r.t. \mathcal{S} and \mathcal{P}_κ by taking $(w_1, w_2) \in \mathbb{R}^2$, $\alpha = (\alpha_1 + \alpha_2)/2 \in \mathbb{R}$, $(\nu_1/2, \nu_2/2) \in \mathcal{K}^2$ and for all $i \in \mathcal{I}$, $\{(\lambda_1^i, 0), (0, \lambda_2^i)\} \in \mathcal{K}^{2 \times 2}$, $(\mu_1^i, \mu_2^i) \in \mathcal{K}^{2 \times n_i}$, $(\gamma_1^i, \gamma_2^i) \in \mathcal{K}^{2 \times n_0}$ (following the order of the parameters in Definition 11). To conclude, we use the fact that \mathcal{K} is closed under nonnegative scalar multiplications. \square

3.3 Practical computation using sum-of-squares programming

Let $\mathbb{R}[x]$ stands for the set of d -variate polynomials and $\mathbb{R}_{2m}[x]$ be its subspace of polynomials of degree at most $2m$. We instantiate \mathcal{K} by the cone of sum-of-squares (SOS), that is $\mathcal{K} = \Sigma[x] := \left\{ \sum_i q_i^2, \text{ with } q_i \in \mathbb{R}[x] \right\}$.

In the sequel, we assume that the data of the CPDS representation \mathcal{S} of some analyzed program are polynomials, that is for all $j = 1, \dots, n_0$, $r_j^{\text{in}} \in \mathbb{R}[x]$, for all $j = 1, \dots, n_0$, $r_j^0 \in \mathbb{R}[x]$, for all $i \in \mathcal{I}$, $T^i \in \mathbb{R}[x]$ and for all $j = 1, \dots, n_i$, $r_j^i \in \mathbb{R}[x]$. We look for a single polynomial template $p \in \mathbb{R}_{2m}[x]$ ($k = 1$) such that the basis $\{p\}$ is $\Sigma[x]$ well-representative w.r.t. \mathcal{S} and \mathcal{P}_κ , thus satisfies the three conditions of Definition 11. One way to strengthen the three conditions of Definition 11 is to take $\lambda^i = 1$, for all $i \in \mathcal{I}$, $\nu = 1$, $\alpha = w$, then to consider the following *hierarchy* of SOS constraints, parametrized by the integer m :

$$\left\{ \begin{array}{l} w - p(x) + \sum_{j=1}^{n_{\text{in}}} \sigma_j(x) r_j^{\text{in}}(x) = \sigma_0(x) \ , \\ \forall i \in \mathcal{I}, -p(T^i(x)) + p(x) + \sum_{j=1}^{n_i} \mu_j^i(x) r_j^i(x) + \sum_{j=1}^{n_0} \gamma_j^i(x) r_j^0(x) = \sigma^i(x) \ , \\ -\kappa(x) + p(x) = \psi(x) \ , \\ p \in \mathbb{R}_{2m}[x] \ , w \in \mathbb{R} \ , \\ \sigma_0 \in \Sigma[x] \ , \deg \sigma_0 \leq 2m \ , \\ \forall j = 1, \dots, n_{\text{in}} \ , \sigma_j \in \Sigma[x] \ , \deg(\sigma_j g_j) \leq 2m \ , \\ \forall i \in \mathcal{I} \ , \sigma^i \in \Sigma[x] \ , \deg(\sigma^i) \leq 2m \deg T^i \ , \\ \forall i \in \mathcal{I} \ , \forall j = 1, \dots, n_i \ , \mu_j^i \in \Sigma[x] \ , \deg(\mu_j^i r_j^i) \leq 2m \deg T^i \ , \\ \forall i \in \mathcal{I} \ , \forall j = 1, \dots, n_0 \ , \gamma_j^i \in \Sigma[x] \ , \deg(\gamma_j^i r_j^0) \leq 2m \deg T^i \ , \\ \psi \in \Sigma[x] \ , \deg(\psi) \leq 2m \ . \end{array} \right. \quad (15)$$

For an integer m , we denote by \mathcal{C}_m the set of constraints on the decision variables $w, p, \sigma^0, \{\sigma_j, j = 1, \dots, n_{\text{in}}\}, \{\sigma^i, i \in \mathcal{I}\}, \{\mu_j^i, i \in \mathcal{I}, j = 1, \dots, n_i\}, \{\gamma_j^i, i \in \mathcal{I}, j = 1, \dots, n_0\}$ and ψ depicted at Equation (15).

As objective function, we choose to minimize w . The intuition behind this choice is that w is enforced to be equal to α which defines the level for which $\{x \in \mathbb{R}^d \mid \kappa(x) \leq \alpha\}$ is an invariant of the program associated to the CPDS \mathcal{S} . When κ is the norm, a minimal value w (and thus α) would be the smallest computable bound on the norm of the state variable x_k . Thus we synthesize a polynomial template of degree at most $2m$ by solving the following minimization problem:

$$\inf \left\{ w \in \mathbb{R} \mid \begin{pmatrix} w, p, \sigma^0, \{\sigma_j, j = 1, \dots, n_{\text{in}}\}, \\ \{\sigma^i, i \in \mathcal{I}\}, \{\mu_j^i, i \in \mathcal{I}, j = 1, \dots, n_i\}, \\ \{\gamma_j^i, i \in \mathcal{I}, j = 1, \dots, n_0\}, \psi \end{pmatrix} \in \mathcal{C}_m \right\}. \quad (16)$$

Hence, computing the polynomial template $p \in \mathbb{R}_{2m}[x]$ boils down to solving an SOS minimization problem. From an optimal solution of Program (16), one can extract the polynomials $\sigma_0, \sigma_1, \dots, \sigma_{n_{\text{in}}}, \psi \in \Sigma[x]$ and for all $i \in \mathcal{I}$, the polynomials $\mu^i, \gamma^i, \sigma^i \in \Sigma[x]$, which are called *SOS certificates*. In practice, one can use the Matlab toolbox YALMIP [LÖ4], which includes a high-level parser for nonlinear optimization and has a built-in module for such SOS calculations. YALMIP reduces SOS programming to semidefinite programming (SDP) (see e.g. [VB94] for more details about SDP), which in turn can be handled with efficient SDP solvers, such as MOSEK [AA00]. In our setting the choice $\alpha = w$ avoids numerical issues while solving SDP programs.

Computational considerations Define $t := \max\{\deg T^i, i \in \mathcal{I}\}$. At step m of this hierarchy, the number of SDP variables is proportional to $\binom{d+2mt}{d}$ and the number of SDP constraints is proportional to $\binom{d+mt}{d}$. Thus, one expects tractable approximations when the number d of variables (resp. the degree $2m$ of the template p) is small. However, one can handle bigger instances of Problem (16) by taking into account the system properties. For instance one could exploit sparsity as in [WKKM06] by considering the variable sparsity correlation pattern of the polynomials $\{T^i, i \in \mathcal{I}\}, \{r_j^i, i \in \mathcal{I}, j = 1, \dots, n_i\}, \{r_j^0, j = 1, \dots, n_0\}, \{r_j^{\text{in}}, j = 1, \dots, n_{\text{in}}\}$ and κ .

Recall that $\mathcal{R}(\mathcal{S})$ is the set of possible values taken by the CPDS \mathcal{S} , which are also the possible values taken by the variables of the program represented by \mathcal{S} .

Proposition 6. *Assume that step m of Problem (16) yields a feasible solution and denote by $p^{(m)} \in \mathbb{R}_{2m}[x]$ (resp. $w^{(m)}$) the polynomial template (resp. the upper bound of $p^{(m)}$ over X^{in}) associated to this solution. Let $v(p^{(m)}) = w^{(m)}$ and thus $v^* := \{x \in \mathbb{R}^d \mid p^{(m)}(x) \leq w^{(m)}\}$. Then $\mathcal{R}(\mathcal{S}) \subseteq v^*$ and $x \in v^* \implies \kappa(x) \leq w^{(m)}$.*

Proof. As a consequence of the first equality constraint of Problem (15), one has $w^{(m)} \geq \sup_{x \in X^{\text{in}}} p^{(m)}(x)$. Then, the finite template basis $\{p^{(m)}\}$ is $\Sigma[x]$ well-representative w.r.t. \mathcal{S} and \mathcal{P}_κ . By Theorem 2, this basis is well-representative

w.r.t. \mathcal{S} and \mathcal{P}_κ . In the proof of Theorem 2, we also proved that $v \in \mathcal{F}(\mathcal{S})$ and $v^\star \subseteq \mathcal{P}_\kappa$. Thus from the second statement of Proposition 2, $\mathcal{R}(\mathcal{S}) \subseteq v^\star$ and \mathcal{P}_κ is sublevel invariant. \square

The next corollary follows directly from Proposition 5 and Proposition 6.

Corollary 1. *Given some integers k and m , assume that steps $m, \dots, m+k$ of Problem (16) yield respective feasible polynomial solutions $p^{(m)}, \dots, p^{(m+k)}$. Then, $\{p^{(m)}, \dots, p^{(m+k)}\}$ is a $\Sigma[x]$ well-representative template basis w.r.t. \mathcal{S} and \mathcal{P}_κ .*

4 Benchmarks

Here, we perform some numerical experiments while solving Problem (16) (given in Section 3.3) on several examples. In Section 4.1, we verify that the program of Example 1 satisfies some boundedness property. We also provide examples involving higher dimensional cases. Then, Section 4.2 focuses on checking that the set of variable values avoids an unsafe region. Numerical experiments are performed on an Intel Core i5 CPU (2.40 GHz) with YALMIP being interfaced with the SDP solver MOSEK. For the sake of simplicity, we write w_m^\star instead of $v(p^{(m)}) = w^{(m)}$.

4.1 Checking boundedness of the set of variables values

Example 6. Following Example 1, we consider the constrained piecewise discrete-time dynamical system $\mathcal{S} = (X^{\text{in}}, X^0, \{X^1, X^2\}, \{T^1, T^2\})$ with $X^{\text{in}} = [0.9, 1.1] \times [0, 0.2]$, $X^0 = \{x \in \mathbb{R}^2 \mid r^0(x) \leq 0\}$ with $r^0 : x \mapsto -1$, $X^1 = \{x \in \mathbb{R}^2 \mid r^1(x) \leq 0\}$ with $r^1 : x \mapsto \|x\|^2 - 1$, $X^2 = \{x \in \mathbb{R}^2 \mid r^2(x) < 0\}$ with $r^2 = -r^1$ and $T^1 : (x_1, x_2) \mapsto (c_{11}x_1^2 + c_{12}x_2^3, c_{21}x_1^3 + c_{22}x_2^2)$, $T^2 : (x_1, x_2) \mapsto (d_{11}x_1^3 + d_{12}x_2^2, d_{21}x_1^2 + d_{22}x_2^2)$. We are interested in proving the boundedness property which a sublevel property \mathcal{P}_κ with $\kappa : x \mapsto \|x\|_2^2$.

Here we illustrate the method by instantiating the program of Example 1 with the following input: $a_1 = 0.9$, $a_2 = 1.1$, $b_1 = 0$, $b_2 = 0.2$, $c_{11} = c_{12} = c_{21} = c_{22} = 1$, $d_{11} = 0.5$, $d_{12} = 0.4$, $d_{21} = -0.6$ and $d_{22} = 0.3$. We represent the possible initial values taken by the program variables (x_1, x_2) by picking uniformly N points $(x_1^{(i)}, x_2^{(i)})$ ($i = 1, \dots, N$) inside the box $X^{\text{in}} = [0.9, 1.1] \times [0, 0.2]$ (see the corresponding square of dots on Figure 2). The other dots are obtained after successive updates of each point $(x_1^{(i)}, x_2^{(i)})$ by the program of Example 1. The sets of dots in Figure 2 are obtained with $N = 100$ and six successive iterations.

At step $m = 3$, Program (16) already yields a feasible solution, from which one can extract the polynomial template $p^{(3)}$ and $w_3 \in \mathcal{F}(\mathcal{S})$. The SOS certificates extracted from this solution guarantee the boundedness property, that is $x \in \mathcal{R}(\mathcal{S}) \implies x \in w_3^\star \implies \|x\|_2^2 \leq w^{(3)}$. Figure 2 displays in light gray outer approximations of the set of possible values X_1 taken by the program of Example 6 as follows: (a) the degree six sublevel set w_3^\star , (b) the degree eight

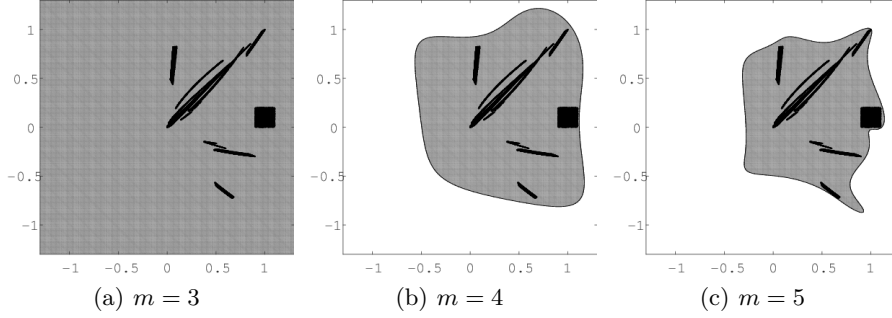


Fig. 2. A hierarchy of sublevel sets w_m^* for Example 6

sublevel set w_4^* and (c) the degree ten sublevel set w_5^* . The outer approximation w_3^* is coarse as it contains the box $[-1.5, 1.5]^2$. However, solving Problem (16) at higher steps yields tighter outer approximations of $\mathcal{R}(\mathcal{S})$ together with more precise bounds $w^{(4)}$ and $w^{(5)}$. Finally, $\{p^{(3)}, p^{(4)}, p^{(5)}\}$ is a $\Sigma[x]$ well-representative template basis w.r.t. to \mathcal{S} and $\mathcal{P}_{\|\cdot\|_2}$ for the program of Example 6.

We also succeeded to certify that the same property holds for higher dimensional programs, described in Example 7 ($d = 3$) and Example 8 ($d = 4$).

Example 7. Here we consider $X^{\text{in}} = [0.9, 1.1] \times [0, 0.2]^2$, $r^0 : x \mapsto -1$, $r^1 : x \mapsto \|x\|_2^2 - 1$, $r^2 = -r^1$, $T^1 : (x_1, x_2, x_3) \mapsto 1/4(0.8x_1^2 + 1.4x_2 - 0.5x_3^2, 1.3x_1 + 0.5x_3^2, 1.4x_2 + 0.8x_3^2)$, $T^2 : (x_1, x_2, x_3) \mapsto 1/4(0.5x_1 + 0.4x_2^2, -0.6x_2^2 + 0.3x_3^2, 0.5x_3 + 0.4x_1^2)$ and $\kappa : x \mapsto \|x\|_2^2$.

Example 8. Here we consider $X^{\text{in}} = [0.9, 1.1] \times [0, 0.2]^3$, $r^0 : x \mapsto -1$, $r^1 : x \mapsto \|x\|_2^2 - 1$, $r^2 = -r^1$, $T^1 : (x_1, x_2, x_3, x_4) \mapsto 0.25(0.8x_1^2 + 1.4x_2 - 0.5x_3^2, 1.3x_1 + 0.5x_2^2 - 0.8x_4^2, 0.8x_3^2 + 1.4x_4, 1.3x_3 + 0.5x_4^2)$, $T^2 : (x_1, x_2, x_3, x_4) \mapsto 0.25(0.5x_1 + 0.4x_2^2, -0.6x_1^2 + 0.3x_2^2, 0.5x_3 + 0.4x_4^2, -0.6x_3 + 0.3x_4^2)$ and $\kappa : x \mapsto \|x\|_2^2$.

Table 1 reports several data obtained while solving Problem (16) at step m , ($2 \leq m \leq 5$), either for Example 6, Example 7 or Example 8. Each instance of Problem (16) is recast as an SDP program, involving a total number of “Nb. vars” SDP variables, with an SDP matrix of size “Mat. size”. We indicate the CPU time required to compute the optimal solution of each SDP program with MOSEK.

The symbol “–” means that the corresponding SOS program could not be solved within one day of computation. These benchmarks illustrate the computational considerations mentioned in Section 3.3 as it takes more CPU time to analyze higher dimensional programs. Note that it is not possible to solve Problem (16) at step 5 for Example 8. A possible workaround to limit this computational blow-up would be to exploit the sparsity of the system.

Table 1. Comparison of timing results for Example 6, 7 and 8

Degree $2m$		4	6	8	10
Example 6 ($d = 2$)	Nb. vars	1513	5740	15705	35212
	Mat. size	368	802	1404	2174
	Time	0.82 s	1.35 s	4.00 s	9.86 s
Example 7 ($d = 3$)	Nb. vars	2115	11950	46461	141612
	Mat. size	628	1860	4132	7764
	Time	0.84 s	2.98 s	21.4 s	109 s
Example 8 ($d = 4$)	Nb. vars	7202	65306	18480	–
	Mat. size	1670	6622	373057	–
	Time	2.85 s	57.3 s	1534 s	–

4.2 Avoiding unsafe regions for the set of variables values

Here we consider the program given in Example 8. One is interested in showing that the set X_1 of possible values taken by the variables of this program does not meet the ball B of center $(-0.5, -0.5)$ and radius 0.5.

Example 9. Let consider the CPDS $\mathcal{S} = (X^{\text{in}}, X^0, \{X^1, X^2\}, \{T^1, T^2\})$ with $X^{\text{in}} = [0.5, 0.7] \times [0.5, 0.7]$, $X^0 = \{x \in \mathbb{R}^2 \mid r^0(x) \leq 0\}$ with $r^0 : x \mapsto -1$, $X^1 = \{x \in \mathbb{R}^2 \mid r^1(x) \leq 0\}$ with $r^1 : x \mapsto \|x\|_2^2 - 1$, $X^2 = \{x \in \mathbb{R}^2 \mid r^2(x) \leq 0\}$ with $r^2 = -r^1$ and $T^1 : (x_1, x_2) \mapsto (x_1^2 + x_2^3, x_1^3 + x_2^2)$, $T^2 : (x, y) \mapsto (0.5x_1^3 + 0.4x_2^2, -0.6x_1^2 + 0.3x_2^2)$. With $\kappa : (x_1, x_2) \mapsto 0.25 - (x_1 + 0.5)^2 - (x_2 + 0.5)^2$, one has $B := \{x \in \mathbb{R}^2 \mid 0 \leq \kappa(x)\}$ and one shall prove that $x \in \mathcal{R}(\mathcal{S}) \implies \kappa(x) < 0$. Note that κ is not a norm, by contrast with the previous examples.

At steps $m = 3, 4$, Program (16) yields feasible solutions with nonnegative bounds $w^{(3)}, w^{(4)}$. Hence, it does not allow to certify that $\mathcal{R}(\mathcal{S}) \cap B$ is empty. This is illustrated in both Figure 3 (a) and Figure 3 (b), where the light grey region does not avoid the ball B . However, solving the SOS feasibility program at step $m = 5$ yields a negative bound $w^{(5)}$ together with a certificate that $\mathcal{R}(\mathcal{S})$ avoids the ball B (see Figure 3 (c)). Finally, $\{p^{(5)}\}$ is a single polynomial template basis w.r.t. \mathcal{S} and \mathcal{P}_κ with the restriction that $\{x \in \mathbb{R}^d \mid p^{(5)}(x) \leq w^{(5)}\} \subseteq \{x \in \mathbb{R}^d \mid \kappa(x) \leq \alpha\}$ for some $\alpha < 0$ for the program of Example 9.

5 Conclusion and Future Works

In this paper, we give a formal framework to relate the template generation problem to the property to prove on analyzed program : well-representative templates. We proposed a practical method to compute well-representative template bases in the case of polynomial arithmetic using sum-of-squares programming. This method is able to handle non trivial examples, as illustrated through the numerical experiments.

Topics of further investigation include refining the invariant bounds generated for a specific sublevel property, by applying the policy iteration algorithm. Such

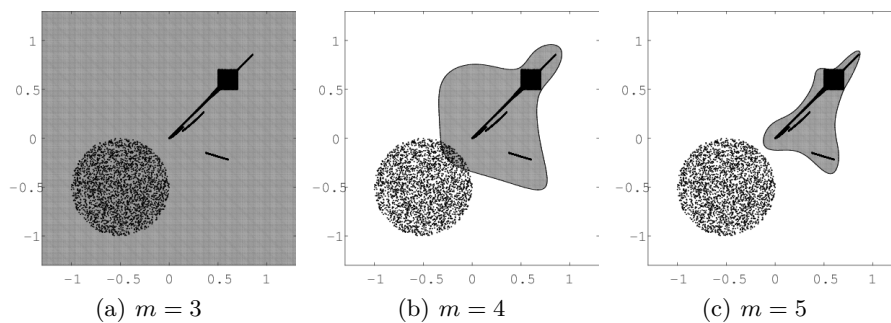


Fig. 3. A hierarchy of sublevel sets w_m^* for Example 9

a refinement would be of particular interest if one can not decide whether the set of variables values avoids an unsafe region when the feasible invariant bound yields a negative value for α . For the case of boundedness property, it would allow to decrease the value of the bounds on the variables. Finally, our method could be generalized to a larger class of programs, involving semialgebraic or transcendental assignments, by using the same polynomial reduction techniques as in [AGMW14].

References

- AA00. Erling D. Andersen and Knud D. Andersen. The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In Hans Frenk, Kees Roos, Tamás Terlaky, and Shuzhong Zhang, editors, *High Performance Optimization*, volume 33 of *Applied Optimization*, pages 197–232. Springer US, 2000.
- AGG10. A. Adjé, S. Gaubert, and E. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In A. D. Gordon, editor, *ESOP*, volume 6012 of *Lecture Notes in Computer Science*, pages 23–42. Springer, 2010.
- AGG11. A. Adjé, S. Gaubert, and E. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. *Logical Methods in Computer Science*, 8(1), 2011.
- AGMW13. Xavier Allamigeon, Stéphane Gaubert, Victor Magron, and Benjamin Werner. Certification of bounds of non-linear functions: the templates method. In *Intelligent Computer Mathematics*, volume 7961 of *Lecture Notes in Computer Science*, pages 51–65. Springer Berlin Heidelberg, 2013.
- AGMW14. Xavier Allamigeon, Stéphane Gaubert, Victor Magron, and Benjamin Werner. Certification of real inequalities – templates and sums of squares. 2014. Submitted for publication.
- AJ13. Amir Ali Ahmadi and Raphael M. Jungers. Switched stability of nonlinear systems via sos-convex lyapunov functions and semidefinite programming. In *CDC'13*, pages 727–732, 2013.

- BRCZ05. R. Bagnara, E. Rodríguez-Carbonell, and E. Zaffanella. Generation of basic semi-algebraic invariants using convex polyhedra. In C. Hankin, editor, *Static Analysis: Proceedings of the 12th International Symposium*, volume 3672 of *LNCS*, pages 19–34. Springer, 2005.
- CC77. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fix-points. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.
- CJJK14. David Cachera, Thomas Jensen, Arnaud Jobin, and Florent Kirchner. Inference of polynomial invariants for imperative programs: A farewell to gröbner bases. *Science of Computer Programming*, 2014.
- CSS03. Michael A. Colón, Sriram Sankaranarayanan, and Henny B. Sipma. Linear invariant generation using non-linear constraint solving. In Jr. Hunt, Warren A. and Fabio Somenzi, editors, *Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 420–432. Springer Berlin Heidelberg, 2003.
- DP02. B. A. Davey and H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, New York, second edition, 2002.
- DT12. Thao Dang and Romain Testylier. Reachability analysis for polynomial dynamical systems using the bernstein expansion. *Reliable Computing*, 17(2):128–152, 2012.
- LÖ4. J. Löfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.
- Las01. Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.
- Lau09. Monique Laurent. Sums of squares, moment matrices and optimization over polynomials. In Mihai Putinar and Seth Sullivant, editors, *Emerging Applications of Algebraic Geometry*, volume 149 of *The IMA Volumes in Mathematics and its Applications*, pages 157–270. Springer New York, 2009.
- LWYZ14. Wang Lin, Min Wu, ZhengFeng Yang, and ZhenBing Zeng. Exact safety verification of hybrid systems using sums-of-squares representation. *Science China Information Sciences*, 57(5):1–13, 2014.
- Mor70. J. J. Moreau. Inf-convolution, sous-additivité, convexité des fonctions numériques. *Journal de Mathématiques Pures et Appliquées*, 49:109–154, 1970.
- MOS04. M. Müller-Olm and H. Seidl. Computing polynomial program invariants. *Inf. Process. Lett.*, 91(5):233–244, 2004.
- Par03. Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003.
- PJ04. Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In Rajeev Alur and George J. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 477–492. Springer Berlin Heidelberg, 2004.
- RCK07. E. Rodríguez-Carbonell and D. Kapur. Automatic generation of polynomial invariants of bounded degree using abstract interpretation. *Sci. Comput. Program.*, 64(1):54–75, 2007.
- RG13. Pierre Roux and Pierre-Loïc Garoche. A polynomial template abstract domain based on bernstein polynomials. In *Sixth International Workshop on Numerical Software Verification (NSV'13)*, 2013.

- RJGF12. P. Roux, R. Jobredeaux, P-L. Garoche, and E. Feron. A generic ellipsoid abstract domain for linear time invariant systems. In T. Dang and I. M. Mitchell, editors, *HSCC*, pages 105–114. ACM, 2012.
- Roc96. R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1996.
- Rub00. A. M. Rubinov. *Abstract Convexity and Global optimization*. Kluwer Academic Publishers, 2000.
- SCSM06. S. Sankaranarayanan, M. Colon, H. Sipma, and Z. Manna. Efficient strongly relational polyhedral analysis. In E. Allen Emerson and Kedar S. Namjoshi, editors, *Verification, Model Checking, and Abstract Interpretation: 7th International Conference, (VMCAI)*, volume 3855 of *LNCS*, pages 111–125, Charleston, SC, January 2006. Springer.
- SG09. Saurabh Srivastava and Sumit Gulwani. Program verification using templates over predicate abstraction. *SIGPLAN Not.*, 44(6):223–234, June 2009.
- Sin97. I. Singer. *Abstract Convex Analysis*. Wiley-Interscience Publication, 1997.
- SSM05. S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *Sixth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *LNCS*, pages 25–41, January 2005.
- STDG12. Mohamed Amin Ben Sassi, Romain Testylier, Thao Dang, and Antoine Girard. Reachability analysis of polynomial systems using linear programming relaxations. In *Automated Technology for Verification and Analysis*, pages 137–151. Springer, 2012.
- VB94. Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1994.
- WKKM06. Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Mura-matsu. Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006.