A Sums-of-Squares Extension of Policy Iterations

Assalé Adjé, Pierre-Loïc Garoche, Victor Magron

A Sums-of-Squares Extension of Policy Iterations

Assalé Adjé*[1], Pierre-Loïc Garoche[2], and Victor Magron[3]

[1] Laboratoire de Mathématiques et Physique (LAMPS),
Université de Perpignan, Perpignan, France
assale.adje@univ-perp.fr
[2] Onera, the French Aerospace Lab, France.
Université de Toulouse, F-31400 Toulouse, France.
pierre-loic.garoche@onera.fr
[3] CNRS Verimag; 2 av de Vignate 38610 Gières France.
victor.magron@imag.fr

Abstract

In order to address the imprecision often introduced by widening operators in static analysis, policy iteration based on min-computations amounts to considering the characterization of reachable value set of a program as an iterative computation of policies, starting from a post-fixpoint. Computing each policy and the associated invariant relies on a sequence of numerical optimizations. While the early research efforts relied on linear programming (LP) to address linear properties of linear programs, the current state of the art is still limited to the analysis of linear programs with at most quadratic invariants, relying on semidefinite programming (SDP) solvers to compute policies, and LP solvers to refine invariants.

We propose here to extend the class of programs considered through the use of Sums-of-Squares (SOS) based optimization. Our approach enables the precise analysis of switched systems with polynomial updates and guards. The analysis presented has been implemented in Matlab and applied on existing programs coming from the system control literature, improving both the range of analyzable systems and the precision of previously handled ones.

1 Introduction

A wide set of critical systems software including controller systems, rely on numerical computations. Those systems range from aircraft controllers, car engine control, anti-collision systems for aircrafts or UAVs, to nuclear powerplant monitors and medical devices such a pacemakers or insulin pumps.

In all cases, the software part implements the execution of an endless loop that reads the sensor inputs, updates its internal states and controls actuators. However the analysis of such software is hardly feasible with classical static analysis tools based on linear abstractions. In fact, according to early results in control theory from Lyapunov in the 19th century, a linear system is defined

*The work was done when the author was at IRIT, Université Paul Sabatier at Toulouse and was supported by the CIMI (Centre International de Mathématiques et d'Informatique) Excellence program ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02 during a postdoctoral fellowship.

as asymptotically stable iff it satisfies the Lyapunov criterion, i.e. the existence of a quadratic invariant. In this view, it is important to develop new analysis tools able to support quadratic or richer polynomial invariants.

```
x ∈ X^in;
while (r_1^0(x)⋈0 and ... and r_{n_0}^0(x)⋈0){
  case (r_1^1(x)⋈0 and ... and r_{n_1}^1(x)⋈0): x = T^1(x);
  case ...
  case (r_1^i(x)⋈0 and ... and r_{n_i}^i(x)⋈0): x = T^i(x);
}
```

(a) A one-loop program with a switch-case loop body.

Let $X^0 = \{y \mid r_1^0(y) \bowtie 0 \text{ and } \dots \text{ and } r_{n_0}^0(y) \bowtie 0\}$
and for all cases i, $X^i = \{y \mid r_1^i(y) \bowtie 0 \text{ and } \dots \text{ and } r_{n_i}^i(y) \bowtie 0\}$.
We can define the discrete-time switched system:

$$x_0 \in X^{\text{in}},\ \forall k \in \mathbb{N},\ x_{k+1} = T(x_k),\ \text{where } T(x) = T^i(x) \text{ if } x \in X^i \cap X^0$$

(b) The discrete-time switched system correspondence of the program.

Figure 1: One-loop programs with switch-case loop body and its representation as a switched system.

While most controllers are linear, their actual implementation is always more complex. e.g. in order to cope with safety, additional constructs such as antiwindups or saturations are introduced. Another classical approach is the use of linear parameter varying controllers (LPV) were the gains of the linear controller vary depending on conditions: this becomes piecewise polynomial controllers at the implementation level.

We are interested here in bounding the set of reachable values of such controllers using sound analyses, that is computing a sound over approximation of reachable states. We specifically focus on a class of programs larger than linear systems: constrained piecewise polynomial systems.

A loop is performed while a conjunction of polynomial inequalities[1] is satisfied. Within this loop, different polynomial updates are performed depending on conjunctions of polynomial constraints. This class of programs is represented in Fig. 1a. The program can be analyzed through its switched system representation (see Fig. 1b). In the obtained system, the conditions to switch are only governed by the state variable: at each time k, we consider the dynamics T^i such that $x_k \in X^i$. So, the switch conditions do not depend on the time (we do not consider the time when we reach X^i) and are not defined from random variables.

From the point-of-view of the dynamical systems or control theory, to find an over-approximation of reachable values set of a program of the class described in Fig. 1a can be reduced to find a positive invariant of its switched system representation (see Fig. 1b).

Moreover, the class of switched systems where the switch conditions only depend on the state variable is classical in control theory and includes the nonlinear systems with dead-zone, saturation,

[1] $\bowtie$ is either the strict $<$ or the weak ($\leq$) comparison operator over reals.

resets or hysteresis.

Related Works

Reachability analysis is a long-standing problem in dynamical systems theory, especially when the system dynamics is nonlinear. In the particular case of polynomial systems, this problem has recently attracted several research efforts.

In [WLW13], the authors use a method based on sublevel sets of polynomials to analyze the reachable set of a continuous time polynomial system with initial/general constraints being encoded by semialgebraic sets. Their method relies on a so-called iterative *advection algorithm* based on Sums-of-Squares (SOS) and semidefinite programming (SDP) to compute either inner of outer approximation of the backward reachable set, also named *domain of attraction* (DoA). The stability analysis of continuous-time hybrid systems with SOS certificates was investigated in [PP09]. [PTT16] have recently applied analogous techniques to perform stability analysis and controller synthesis in the context of robotics. Other studies rely on SOS reinforcement and moment relaxations to obtain hierarchies of approximations converging to sets of interest such as the DoA in continuous-time, either from outside in [HK14] or from inside in [KHJ12]. This approach relies on a linearization of the ordinary differential equation involved in the polynomial system, based on Liouville's Equation satisfied by adequate occupation measures. This framework has been extended to hybrid systems in [SVBT14], as well as to synthesis of feedback controllers in [MVTT13]. Liouville's Equation can also be used to approximate other sets of interest, such as the maximal controlled invariant for either discrete- or continuous-time systems (see [KHJ14]).

By contrast with these prior works, our approach focuses on computing over approximation of the forward reachable set of a discrete-time polynomial system with finitely many guards, thanks to an algorithm relying on SOS and template policy iterations.

Template abstractions were introduced in [SSM04] as a way to define an abstraction based on an a-priori known vector of templates, i.e. numerical expressions over the program variables. An abstract element is then defined as a vector of reals defining bounds b_i over the templates p_i: $p_i(x_1, \ldots, x_d) \leq b_i$.

Initially templates were used in the classical abstract interpretation setting to compute Kleene fixpoints with linear functions p_i. Typically, the values of the bound b_i increases during the fixpoint computation until convergence to a post-fixpoint.

Later in [AGG10] the authors proposed to consider generalized templates including quadratic forms and compute directly the fixpoint of these template-based abstractions using numerical optimization. When considering the sub-class of linear programs, fixpoints can be computed by using a finite sequence of linear (LP) and semidefinite (SDP) optimization problems.

Two dual approaches, respectively called Max-policies and Min-policies, can be applied. Max-policies [GS07] iterate from initial states and compute *policies* as relaxations through rewriting of an optimization problem (forgetting about rank conditions). Min-policies [GGTZ07, AGG10] rely on duality principle and determine a policy through the computation of a Lagrange multiplier.

Contributions

The present paper is a followup of [AGM15], in which Sums-of-Squares (SOS) programming is used to analyze properties (such as boundedness or safety) of piecewise discrete polynomial systems. The main contribution is the extension of the Min-policy iteration algorithm to improve the

precision of the analysis of boundedness for such systems. Here, we develop a policy iteration on template domains based on polynomials. The current approach improves the previous frameworks developed in [GGTZ07, AGG12] to handle affine (or very specific) piecewise affine systems with quadratic templates and semidefinite programming. This improvement comes from the use of SOS programming to develop an SOS extension of a relaxed functional, sharing the same properties as the one defined in [AGG12]. In particular, we show, as in [Adj14], that this policy iteration has the desired convergence guarantees.

Organization of the paper

The paper is structured as follows: we characterize the class of programs considered – constrained piecewise discrete-time polynomial systems – together with their collecting semantics as a least fixpoint. Then, in Section 3, we recall definitions of generalized templates, their expression as an abstract domain and the definition of the abstract transfer function. Section 4 proposes an abstraction of the transfer function using an SOS reinforcement, while Section 5 relies on this abstraction to perform policy iteration. Finally Section 6 presents experiments.

2 Constrained Piecewise Discrete-Time Polynomial Systems: Definition and Collecting Semantics

In this paper, we are interested in proving automatically that the set of values taken by the variables of the analyzed program is bounded. In the rest of the paper, we analyze the program by decomposing it using its dynamic system representation. The boundedness problem is thus reduced to prove that the set of all the possible trajectories of a dynamical system is bounded. Since the analyzed program has the form depicted by Figure 1a, we consider the special class of discrete-time dynamical systems introduced in Figure 1b that is:

(i) their dynamic law T is a piecewise polynomial function, and

(ii) the state variable x is constrained to live in some given basic semi-algebraic set[2].

We recall that a set is a basic semi-algebraic set if and only if it can be represented as a conjunction of strict or weak polynomial inequalities ("basic" means that no disjunction occurs). Note that T is a piecewise polynomial function with respect to a given partition, meaning that if we restrict T to be an element of the partition then T is a polynomial function.

We now give a formal definition of *constrained piecewise discrete-time polynomial system* (PPS for short).

First to define a PPS, we need a partition. Let $\mathcal{I}$ be a finite set of partition labels and $\mathcal{X} = \{X^i \subseteq \mathbb{R}^d \mid i \in \mathcal{I}\}$ be a partitioning, that is a given family of basic semi-algebraic sets satisfying the following:

$$\bigcup_{i\in\mathcal{I}} X^i = \mathbb{R}^d,\ \forall\, i,j \in \mathcal{I},\ i \neq j \Rightarrow X^i \cap X^j = \emptyset\ . \tag{1}$$

Each set X^i of the partition corresponds to a case in the loop body of the program given in Figure 1a as the cases are assumed to be disjoint. By definition of basic semi-algebraic sets, it follows that

[2]For instance membership of the sub-level set $\{x \in \mathbb{R}^d \mid 1 - \|x\|_2^2 \leq 0\}$, thus this does not entail boundedness of variable values.

for all $i \in \mathcal{I}$, there exists a family of n_i polynomials $\{r_j^i, j \in [n_i]\}$ such that:

$$X^i = \left\{ x \in \mathbb{R}^d \mid r_j^i(x) \bowtie 0 \ \forall\, j \in [n_i] \right\} . \tag{2}$$

where $\bowtie$ is either $<$ or $\leq$ and $[n_i]$ denotes the set $\{1, \ldots, n_i\}$. Eq. (2) matches with the program given in Figure 1a. Guards are defined as conjunctions of polynomial inequalities and thus are basic semi-algebraic sets.

The second tool needed to define a PPS is the piecewise polynomial dynamic relative to the partition. Let $T : \mathbb{R}^d \mapsto \mathbb{R}^d$ be a piecewise polynomial function w.r.t. to the partitioning $\mathcal{X}$. By definition, there exists a family of polynomials $\{T^i : \mathbb{R}^d \mapsto \mathbb{R}^d, i \in \mathcal{I}\}$ such that for all $i \in \mathcal{I}$:

$$T(x) = T^i(x), \ \forall x \in X^i \ . \tag{3}$$

Eq. (3) matches with the polynomial updates in Figure 1a.

Finally, it remains to define the initialization and the set where the state variable lies. Let X^{in} and X^0 be two basic semi-algebraic sets of $\mathbb{R}^d$, X^{in} supposed to be compact, i.e. closed and bounded. The two sets can be represented as in Eq. (2) using their respective family of n_{in} and n_0 polynomials:

$$X^{\text{in}} = \left\{ x \in \mathbb{R}^d \mid r_j^{\text{in}}(x) \bowtie 0 \ \forall\, j \in [n_{\text{in}}] \right\} \text{ and } X^0 = \left\{ x \in \mathbb{R}^d \mid r_j^0(x) \bowtie 0 \ \forall\, j \in [n_0] \right\} \ ,$$

where for all $j \in [n_{\text{in}}]$, $r_j^{\text{in}} : \mathbb{R}^d \mapsto \mathbb{R}$ and for all $k \in [n_0]$, $r_k^0 : \mathbb{R}^d \mapsto \mathbb{R}$ is a polynomial.

The set X^{in} and X^0 respectively denote the set of initial states of the program and the set which defines the loop condition in Figure 1a.

Let $\mathcal{X}$ be the family of sets $\{X^i, i \in \mathcal{I}\}$ satisfying Eq. (1) and $\mathcal{T}$ be the family of functions $\{T^i, i \in \mathcal{I}\}$ satisfying Eq. (3). We define the PPS associated to the quadruple $(X^{\text{in}}, X^0, \mathcal{X}, \mathcal{T})$ as the system satisfying the following dynamic:

$$x_0 \in X^{\text{in}}, \text{ and } \forall\, k \in \mathbb{N}, \text{if } x_k \in X^0, \ x_{k+1} = T(x_k) \, . \tag{4}$$

In the rest of the paper, a PPS dynamical system is identified by a quadruple $(X^{\text{in}}, X^0, \mathcal{X}, \mathcal{T})$.

Example 2.1 (Running example)**.** *We consider the following running example corresponding to a slightly modified version of [AJ13, Example 3]. By comparison with [AJ13, Example 3], the semi-algebraic set X^1 (resp. X^2) is introduced to represent conditions under which we use the polynomial update T^1 (resp. T^2). The PPS is the quadruple $(X^{\text{in}}, X^0, \{X^1, X^2\}, \{T^1, T^2\})$, where:*

$$\begin{array}{l} X^{\text{in}} = [-1, 1] \times [-1, 1] \\ X^0 = \mathbb{R}^2 \end{array} \quad \textit{and} \quad \left\{ \begin{array}{l} X^1 = \{x \in \mathbb{R}^2 \mid -x_1^2 + 1 \leq 0\} \\ X^2 = \{x \in \mathbb{R}^2 \mid x_1^2 - 1 < 0\} \end{array} \right.$$

and the family of functions $\{T^1, T^2\}$, defined as follows:

$$T^1(x_1, x_2) = \begin{pmatrix} 0.687x_1 + 0.558x_2 - 0.0001x_1x_2 \\ -0.292x_1 + 0.773x_2 \end{pmatrix}$$

and

$$T^2(x_1, x_2) = \begin{pmatrix} 0.369x_1 + 0.532x_2 - 0.0001x_1^2 \\ -1.27x_1 + 0.12x_2 - 0.0001x_1x_2 \end{pmatrix}$$

Its (discretized) reachable value set is simulated and depicted at Figure 2.

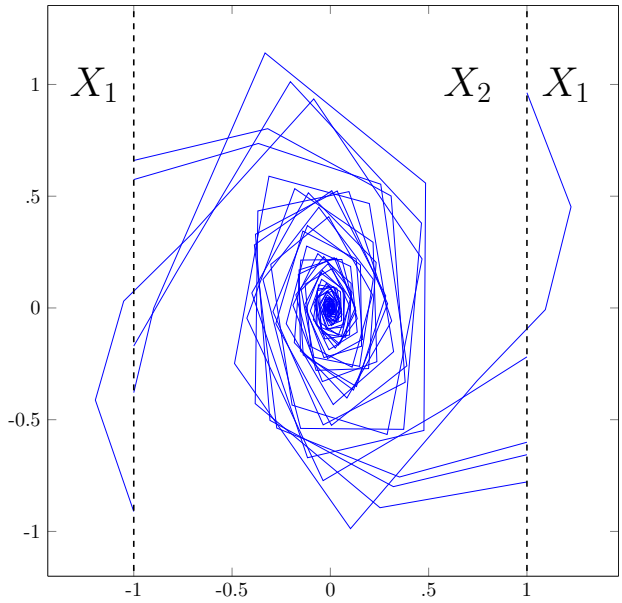


Figure 2: Running example simulation

We recall that our objective is to prove automatically that the set of the possible trajectories of the system is bounded. This set, also called the reachable values set or the collecting semantics of the system, is defined as follows:

$$\mathfrak{R} = \bigcup_{k \in \mathbb{N}} T^{(k)}_{|_{X^0}}(X^{\text{in}}) \tag{5}$$

where $T_{|_{X^0}}$ is the restriction of T over X^0.The notation $T^{(k)}_{|_{X^0}}$ stands for composing k-times the map $T_{|_{X^0}}$.

To prove this boundedness property, we can compute $\mathfrak{R}$ and do some analysis to prove that $\mathfrak{R}$ is bounded. Nevertheless, the computation of $\mathfrak{R}$ cannot be done in general and instead, we have to compute an over-approximation of $\mathfrak{R}$ and show that this approximation is bounded.

The usual abstract interpretation methodology to characterize and to construct this over-approximation relies on the representation of $\mathfrak{R}$ as the smallest fixed point of a monotone map over a complete lattice. In other words, $\mathfrak{R}$ satisfies:

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

Let us define $\wp(\mathbb{R}^d)$ the set of subsets of $\mathbb{R}^d$ and introduce the function $F : \wp(\mathbb{R}^d) \mapsto \wp(\mathbb{R}^d)$ defined for all $C \in \wp(\mathbb{R}^d)$ by:

$$F(C) = T(C \cap X^0) \cup X^{\text{in}} = \bigcup_{i \in \mathcal{I}} T^i(C \cap X^0 \cap X^i) \cup X^{\text{in}} \ . \tag{6}$$

Thus, $\mathfrak{R}$ is the smallest fixed point of F and from Tarski's Theorem, since F is monotone and $\wp(\mathbb{R}^d)$ is a complete lattice:

$$\mathfrak{R} = \min\{C \in \wp(\mathbb{R}^d) \mid F(C) \subseteq C\} \ . \tag{7}$$

Finally to compute an over-approximation of $\mathfrak{R}$ it suffices to compute a set C such that $F(C) \subseteq C$. A set C which satisfies $F(C) \subseteq C$ is called an *inductive invariant*[3].

The rest of the paper addresses the computation of a sound over-approximation of $\mathfrak{R}$ using its definition as the smallest fixpoint of F (Eq. (7)).

3 Basic Semi-algebraic Inductive Invariants Set

An easy way to over-approximate the set of reachable values is to restrict the set of inductive invariants that we consider. We propose to restrict the class of such invariants to basic semi-algebraic sets using template abstractions. A template abstraction consists of representing a given set as the intersection of sublevel sets of *a-priori fixed* functions depending on the state variables. Such functions are called *templates*. Then computing an inductive invariant in the templates domain boils down to providing, for each template p, a bound $w(p)$ such that the intersection over the templates p of sublevel sets $\{x \in \mathbb{R}^d \mid p(x) \leq w(p)\}$ is an inductive invariant. In our context, a template is simply an *a-priori* fixed multivariate polynomial.

The overall method is not new and corresponds to a specialization of the templates abstraction (see [AGG10, AGG12]) to polynomial templates. However, in practice, the method developed in [AGG10, AGG12] is restricted to template polynomials of degree 2 (quadratic forms) and affine systems or a very restricted class of piecewise affine systems.

Next we give formal details about the polynomial template abstraction and the equations that must satisfy the template bounds vector w to generate an inductive invariant. From now on, we denote by $\mathbb{P}$ the set of templates and by $\mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ the set of functions from $\mathbb{P}$ to $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. We equip $\mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ with the functional partial order $\leq_{\mathcal{F}}$ i.e. $v \leq_{\mathcal{F}} w$ iff $v(p) \leq w(p)$ for all $p \in \mathbb{P}$. Let $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$. The sets that we consider are of the form:

$$w^\star = \{x \in \mathbb{R}^d \mid p(x) \leq w(p), \forall\, p \in \mathbb{P}\}\,. \tag{8}$$

Example 3.1. *Let us define $q_1(x) = q_1(x_1, x_2) = x_1^2$, $q_2(x) = q_2(x_1, x_2) = x_2^2$ and let us consider a well-chosen polynomial p of degree 6. We will explain in Subsection 5.3 how to generate automatically this template p. Let us define $\mathbb{P}_{\text{run}} := \{q_1, q_2, p\}$.*

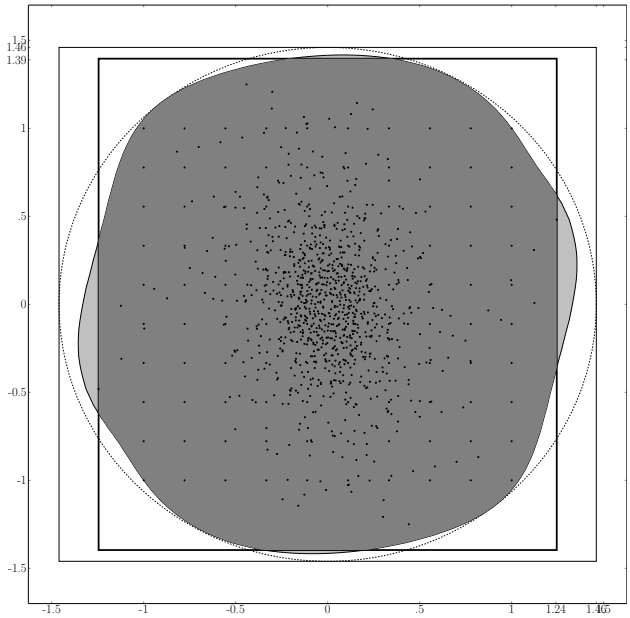
Consider the function w^0 over $\mathbb{P}_{\text{run}}$, $w^0(q_1) = w^0(q_2) = 2.1391$ and $w^0(p) = 0$, the set $w^{0\star} = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 \leq w^0(q_1),\ x_2^2 \leq w^0(q_2),\ p(x) \leq w^0(p)\}$ is presented in Figure 3.

Now let us take the function w^1 over $\mathbb{P}_{\text{run}}$ defined by $w^1(q_1) = 1.5503$, $w^1(q_2) = 1.9501$ and $w^1(p) = 0$, the set $w^{1\star} = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 \leq w^1(q_1),\ x_2^2 \leq w^1(q_2),\ p(x) \leq w^1(p)\}$ is also presented in Figure 3.

Since the set of black dots in Figure 3 belongs to $w^{0\star}$ and $w^{1\star}$, we guess that $w^{0\star}$ and $w^{1\star}$ contain both the reachability set of the system from Example 2.1. To formally prove it, one way is to show that they are also inductive invariants for this system.

We restrict the class of inductive invariants to those of the form (8) and characterize the inductiveness for such sets. Since each polynomial template p is fixed, the considered variables that we handle are the template bounds $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$. Therefore, we need to translate the

[3] In the dynamical systems theory, the inductive invariant sets are called positive invariant.



The semialgebraic set of gray denotes the set $w^{0\star}$ of Example 3.1 since $p(x) \leq 0$ is included in $x_i^2 \leq 2.1391$, for $i \in [1,2]$. The dark gray region denotes the semialgebraic set $w^{0\star}$. Black dots are actual reachable states of $\mathfrak{R}$ obtained by simulation.

Figure 3: Semialgebraic sets $w^\star$ for Example 3.1

inductiveness of the sets $w^\star$ into inequalities on w. By definition the set $w^\star$ is an inductive invariant iff $F(w^\star) \subseteq w^\star$, that is:

$$\bigcup_{i\in\mathcal{I}} T^i(w^\star \cap X^0 \cap X^i) \cup X^{\text{in}} \subseteq w^\star \ .$$

By definition, $w^\star$ is an inductive invariant iff:

$$\forall\, p \in \mathbb{P},\ \forall x \in \bigcup_{i\in\mathcal{I}} T^i(w^\star \cap X^0 \cap X^i) \cup X^{\text{in}},\ p(x) \leq w(p) \ .$$

Using the definition of the supremum, $w^\star$ is an inductive invariant iff:

$$\forall\, p \in \mathbb{P},\quad \sup_{x \in \bigcup_{i\in\mathcal{I}} T^i(w^\star \cap X^0 \cap X^i) \cup X^{\text{in}}} p(x) \leq w(p) \ .$$

Now, let us consider $p \in \mathbb{P}$. Using the fact that for all $A, B \subseteq \mathbb{R}^d$ and for all functions f, $\sup_{A\cup B} f = \sup\{\sup_A f, \sup_B f\}$:

$$\sup_{x\in\bigcup_{i\in\mathcal{I}} T^i(w^\star\cap X^0\cap X^i)\cup X^{\text{in}}} p(x) = \sup\left\{\sup_{i\in\mathcal{I}} \sup_{x\in T^i(w^\star\cap X^0\cap X^i)} p(x),\ \sup_{x\in X^{\text{in}}} p(x)\right\} \ .$$

By definition of the image:

$$\sup_{x\in\bigcup_{i\in\mathcal{I}} T^i(w^\star\cap X^0\cap X^i)\cup X^{\text{in}}} p(x) = \sup\left\{\sup_{i\in\mathcal{I}} \sup_{y\in w^\star\cap X^0\cap X^i} p(T^i(y)),\ \sup_{x\in X^{\text{in}}} p(x)\right\} \ .$$

Next, we introduce the following notation, for all $p \in \mathbb{P}$:

$$F_i^\sharp(w)(p) := \sup_{x \in w^\star \cap X^i \cap X^0} p(T^i(x)) \quad \text{and} \quad X^{\text{in}\dagger}(p) := \sup_{x \in X^{\text{in}}} p(x) \ .$$

Finally, we define the function from $\mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ to itself, for all $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$:

$$F^\sharp(w) := \sup \left\{ \sup_{i \in \mathcal{I}} F_i^\sharp(w), X^{\text{in}\dagger} \right\} .$$

Note that $^\sharp$, † correspond exactly to the notations used in [AGG10]. By construction we obtain the following proposition:

Proposition 3.1. *Let $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$. Then $w^\star$ is an inductive invariant (i.e. $F(w^\star) \subseteq w^\star$) iff $F^\sharp(w) \leq_{\mathcal{F}} w$.*

From Prop. 3.1, $\inf\{w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n \mid F^\sharp(w) \leq_{\mathcal{F}} w\}$ identifies the smallest inductive invariant $w^\star$ of the form (8).

Example 3.2. *Let us consider the system defined at Example 2.1. Let us consider the same templates basis $\mathbb{P}_{\text{run}}$ from Example 3.1 i.e. $\mathbb{P}_{\text{run}} = \{q_1, q_2, p\}$ where $q_1(x) = x_1^2$, $q_2(x) = x_2^2$ and p is a well-chosen polynomial of degree 6. Let $w \in \mathcal{F}\left(\mathbb{P}_{\text{run}}, \overline{\mathbb{R}}\right)$. For $i = 1$ and the templates q_1, we have:*

$$F_1^\sharp(w)(q_1) = \sup_{\substack{-x_1^2+1 \leq 0 \\ x_1^2 \leq w(q_1),\ x_2^2 \leq w(q_2),\ p(x) \leq w(p)}} (0.687x_1 + 0.558x_2 - 0.0001x_1x_2)^2 .$$

Indeed, $X^1 = \{x \in \mathbb{R}^2 \mid -x_1^2 + 1 \leq 0\}$ and $X^0 = \mathbb{R}^2$ and the dynamics associated with X^1 is the polynomial function T^1 defined for all $x \in \mathbb{R}^2$ by: $T^1(x) = \begin{pmatrix} 0.687x_1+0.558x_2-0.0001x_1x_2 \\ -0.292x_1+0.773x_2 \end{pmatrix}$. Since q_1 computes the square of the first coordinates, this yields $q_1(T^1(x)) = (0.687x_1 + 0.558x_2 - 0.0001x_1x_2)^2$.

With $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, computing $F^\sharp(w)$ boils down to solving a finite number of nonconvex polynomial optimization problems. General methods do not exist to solve such problems. In Section 4, we propose a method based on Sums-of-Squares (SOS) to over-approximate $F^\sharp(w)$.

4 SOS-based Relaxed Semantics

In this section, we introduce the relaxed functional on which we will compute a fixpoint, yielding a further over-approximation of the set $\mathfrak{R}$ of reachable values. This relaxed functional is constructed from a Lagrange relaxation of maximization problems involved in the evaluation of $F^\sharp$ and Sums-of-Squares strengthening of polynomial nonnegativity constraints. First, we recall mandatory background related to Sums-of-Squares and their application in polynomial optimization. The interested reader is referred to [Las09] for more details.

4.1 Sums-of-Squares Programming

Let $\mathbb{R}[x]_{2m}$ stands for the set of polynomials of degree at most $2m$ and $\Sigma[x] \subset \mathbb{R}[x]$ be the cone of Sums-of-Squares (SOS) polynomials, that is $\Sigma[x] := \{ \sum_i q_i^2, \text{ with } q_i \in \mathbb{R}[x] \}$.

Our work will use the simple fact that for all $q \in \Sigma[x]$, then $q(x) \geq 0$ for all $x \in \mathbb{R}^d$ as the set $\Sigma[x]$ contains only nonnegative polynomials. In other words, for any given polynomial q, we can strengthen the constraint of q being nonnegative into the existence of an SOS decomposition of q.

For $q \in \mathbb{R}[x]_{2m}$, finding an SOS decomposition $\sum_i q_i^2 = q$ valid over $\mathbb{R}^d$ is equivalent to solve the following matrix linear feasibility problem:

$$q(x) = b_m(x)^T \, Q \, b_m(x) \,, \quad \forall x \in \mathbb{R}^d, \tag{9}$$

where $b_m(x) := (1, x_1, \ldots, x_d, x_1^2, x_1x_2, \ldots, x_d^m)$ (the vector of all monomials in x up to degree m) and the Gram matrix Q, being a *semidefinite positive* matrix (i.e. all the eigenvalues of Q are nonnegative). The size of Q (as well as the length of b_m) is $\binom{d+m}{d}$.

Example 4.1. *Consider the bi-variate polynomial $q(x) := 1 + x_1^2 - 2x_1x_2 + x_2^2$. With $b_1(x) = (1, x_1, x_2)$, one looks for a semidefinite positive matrix Q such that the polynomial equality $q(x) = b_1(x)^T \, Q \, b_1(x)$ holds for all $x \in \mathbb{R}^2$. The matrix*

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

satisfies this equality and has three nonnegative eigenvalues, which are 0, 1, and 2, respectively associated to the three eigenvectors $e_0 := (0, 1/\sqrt{2}, 1/\sqrt{2})^\intercal$, $e_1 := (1, 0, 0)^\intercal$ and $e_2 := (0, 1/\sqrt{2}, -1/\sqrt{2})^\intercal$. Defining the matrices $L := (e_1 \, e_2 \, e_0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$ and $D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, one obtains the decomposition $Q = L^\intercal \, D \, L$ and the equality $q(x) = (L \, b_1(x))^T \, D \, (L \, b_1(x)) = \sigma(x) = 1 + (x_1 - x_2)^2$, for all $x \in \mathbb{R}^2$. The polynomial σ is called an SOS certificate *and guarantees that q is nonnegative.*

In practice, one can solve the general problem (9) by using semidefinite programming (SDP) solvers (e.g. Mosek [AA00], SDPA [YFN+10], CSDP [Bor99]). For more details about SDP, we refer the interested reader to [VB94].

The SOS reinforcement of polynomial optimization problems consists of restricting polynomial nonnegativity to being an element of $\Sigma[x]$. In case of polynomial maximization problems, the SOS reinforcement boils down to computing an upper bound of the real optimal value. For example let $p \in \mathbb{R}[x]$ and consider the unconstrained polynomial maximization problem $\sup\{p(x), \; x \in \mathbb{R}^d\}$. Applying SOS reinforcement, we obtain:

$$\sup\{p(x), \; x \in \mathbb{R}^d\} = \inf\{\eta \mid \eta - p(x) \geq 0\} \leq \inf\{\eta \mid \eta - p(x) \in \Sigma[x]\} \;. \tag{10}$$

Now, let $p, q \in \mathbb{R}[x]$ and consider the constrained polynomial maximization problem: $\sup\{p(x) | q(x) \leq 0, \; x \in \mathbb{R}^d\}$. Let $\lambda \in \Sigma[x]$, then:

$$\sup_{q(x) \leq 0, \; x \in \mathbb{R}^d} p(x) \leq \sup_{x \in \mathbb{R}^d} \; p(x) - \lambda(x) \cdot q(x) \;.$$

Indeed, suppose $q(x) \leq 0$, then $-\lambda(x)q(x) \geq 0$ and $p(x) \leq p(x) - \lambda(x)q(x)$. Finally taking the supremum over $\{x \in \mathbb{R}^d \mid q(x) \leq 0\}$ provides the above inequality. Since $\sup\{p(x) - \lambda(x) \cdot q(x), \; x \in$

$\mathbb{R}^d\}$ is an unconstrained polynomial maximization problem then we apply an SOS reinforcement (as in Eq. (10)) and we obtain:

$$\sup_{q(x)\leq 0,\ x\in\mathbb{R}^d} p(x) \leq \sup_{x\in\mathbb{R}^d} p(x) - \lambda(x)\cdot q(x) \leq \inf\{\eta \mid \eta - p - \lambda q \in \Sigma[x]\}\,.$$

Finally, note that this latter inequality is valid whatever $\lambda \in \Sigma[x]$ and so we can take the infimum over $\lambda \in \Sigma[x]$ which leads to:

$$\sup_{q(x)\leq 0,\ x\in\mathbb{R}^d} p(x) \leq \inf_{\lambda\in\Sigma[x]} \sup_{x\in\mathbb{R}^d} p(x) - \lambda(x)\cdot q(x) \leq \inf_{\substack{\eta-p-\lambda q\in\Sigma[x]\\ \lambda\in\Sigma[x]}} \eta\,. \tag{11}$$

In Eq. (11), λ is an SOS polynomial but to exploit linear programming solvers in policy iterations (see the fourth assertion of Prop. 5.2) we restrict λ to be a nonnegative scalar and in this case, since positive scalars are sum-of-squares polynomials of degree 0, we obtain a safe over-approximation of the right-hand-side of Eq. (11).

In presence of several constraints, we assign to each constraint an element $\sigma \in \Sigma[x]$, and we consider the product of σ with its associated constraint and then the sum of all such products. This sum is finally added to the objective function.

The use of such SOS polynomials for constrained polynomial optimization problem can be seen as a generalization of the S-procedure from [Yak77]. We refer to [Las01] or [Par03] for applications in control. Note that the existence of SOS decompositions of positive polynomials over compact sets is ensured by the Putinar Positivstellensatz from [Put93].

4.2 Relaxed semantics

The computation of $F^\sharp$ as a polynomial maximization problem cannot be directly performed using numerical solvers. We use the SOS reinforcement mechanisms described above to relax the computation and characterize an abstraction of $F^\sharp$.

We still assume the knowledge of the template basis $\mathbb{P}$, involving polynomials of degree at most $2m$. Let us define $\mathcal{F}(\mathbb{P},\mathbb{R}_+)$ the set of nonnegative functions over $\mathbb{P}$ i.e. $g \in \mathcal{F}(\mathbb{P},\mathbb{R}_+)$ iff for all $p\in\mathbb{P}$, $g(p)\in\mathbb{R}_+$. Let $p\in\mathbb{P}$ and $w \in \mathcal{F}\left(\mathbb{P},\overline{\mathbb{R}}\right)$. Starting from the definition of $F_i^\sharp$, one obtains the

following:

$$
\begin{aligned}
(F_i^\sharp(w))(p) &= \sup_{\substack{q(x)\le w(q),\ \forall q\in\mathbb{P}\\ r_j^i(x)\le 0,\ \forall j\in[n_i]\\ r_k^0(x)\le 0,\ \forall k\in[n_0]}} p(T^i(x)) \\
&\le \inf_{\substack{\lambda\in\mathcal{F}(\mathbb{P},\mathbb{R}_+)\\ \sigma\in\Sigma[x],\mu_l\in\Sigma[x],\gamma_l\in\Sigma[x]\\ \deg(\sigma)\le 2m\deg T^i\\ \deg(\mu_l r_l^i)\le 2m\deg T^i\\ \deg(\gamma_l r_l^0)\le 2m\deg T^i}} \sup_{x\in\mathbb{R}^d} p(T^i(x)) + \sum_{\mathrm{q}\in\mathbb{P}} \lambda(q)(w(q)-q(x)) \\
&\qquad\qquad - \sum_{\mathrm{l}=1}^{\mathrm{n^i}} \mu_l(x) r_l^i(x) - \sum_{\mathrm{l}=1}^{\mathrm{n^0}} \gamma_l(x) r_l^0(x) \\
&\le \inf_{\lambda,\sigma,\mu_l,\gamma_l,\eta} \eta \\
&\qquad \text{s. t.} \begin{cases} \eta - p\circ T^i - \sum_{\mathrm{q}\in\mathbb{P}} \lambda(q)(w(q)-q) + \sum_{\mathrm{l}=1}^{\mathrm{n^i}} \mu_l r_l^i + \sum_{\mathrm{l}=1}^{\mathrm{n^0}} \gamma_l r_l^0 = \sigma\,, \\ \lambda\in\mathcal{F}(\mathbb{P},\mathbb{R}_+)\,,\ \sigma\in\Sigma[x],\ \mu_l\in\Sigma[x],\ \gamma_l\in\Sigma[x],\ \eta\in\mathbb{R}\,, \\ \deg(\sigma)\le 2m\deg T^i\,, \\ \deg(\mu_l r_l^i)\le 2m\deg T^i,\ \deg(\gamma_l r_l^0)\le 2m\deg T^i\,. \end{cases} \\
&\qquad\qquad \text{(using an SOS reinforcement to remove the sup)}
\end{aligned}
$$

We denote by $\Sigma[x]^n$ the set of n-tuples of SOS polynomials. For clarity purpose, the dependency on i is omitted within the notations of the multipliers μ_l and γ_l. Moreover, let us write $\sum_{l=1}^{n^i} \mu_l r_l^i$ (resp. $\sum_{\mathrm{l}=1}^{\mathrm{n^0}} \gamma_l r_l^0$) as $\langle \mu, r^i\rangle$ (resp. $\langle \gamma, r^0\rangle$). Finally, we write $(F_i^{\mathcal{R}}(w))(p)$ the over-approximation of $(F_i^\sharp(w))(p)$, defined as follows:

$$
\begin{aligned}
(F_i^{\mathcal{R}}(w))(p) = \inf_{\lambda,\sigma,\mu,\gamma,\eta} \eta \\
\text{s. t.} \begin{cases} \eta - p\circ T^i - \sum_{\mathrm{q}\in\mathbb{P}} \lambda(q)(w(q)-q) + \langle \mu, r^i\rangle + \langle \gamma, r^0\rangle = \sigma \\ \lambda\in\mathcal{F}(\mathbb{P},\mathbb{R}_+)\,,\ \sigma\in\Sigma[x],\ \mu\in\Sigma[x]^{n_i},\ \gamma\in\Sigma[x]^{n_0},\ \eta\in\mathbb{R}\,, \\ \deg(\sigma)\le 2m\deg T^i,\ \deg(\langle \mu, r^i\rangle + \langle \gamma, r^0\rangle)\le 2m\deg T^i\,. \end{cases}
\end{aligned}
\tag{12}
$$

In Equation (12), the notation λ is a vector of Lagrange multipliers. Each multiplier is associated with a constraint constructed from a template i.e. a constraint $q(x) - w(q) \le 0$. We also introduce the vector of SOS polynomials μ and γ. Their role is to take into account the presence of the constraints $x \in X^i$ and $x \in X^0$ in the computation of $(F_i^{\mathcal{R}}(w))(p)$. Recall that X^i and X^0 are basic semi-algebraic sets, then the size of the vectors μ and γ are equal to the number of polynomials defining X^i and X^0.

We conclude that, for all $i \in \mathcal{I}$, the evaluation of $F_i^{\mathcal{R}}$ can be done using SOS programming, since it is reduced to solve a minimization problem with a linear objective function and linear combination of polynomials constrained to be sum-of-squares.

Note that $F_i^{\mathcal{R}}$ defined at Eq. (12) is the SOS extension of the relaxed function defined in [AGG12]. Indeed, considering the special case where T^i is affine, the templates p, q and the test functions r^i, r^0 are quadratic, the vectors μ^i and γ^i are restricted to be nonnegative scalars, then $F_i^{\mathcal{R}}$ corresponds to the relaxed function defined in [AGG12] at Eq. (3.12).

Example 4.2. *We still consider the running example defined at Example 2.1 and take again the same templates basis* $\mathbb{P}_{\mathrm{run}}$ *of Example 3.1 composed of* $q_1 : x \mapsto x_1^2$ *and* $q_2 : x \mapsto x_2^2$. *and a*

well-chosen polynomial p of degree 6. For the index of the partition $i = 1$. Recall that $T^1(x) = \begin{pmatrix} 0.687x_1+0.558x_2-0.0001x_1x_2 \\ -0.292x_1+0.773x_2 \end{pmatrix}$ *and* $X^1 = \{x \in \mathbb{R}^2 \mid -x_1^2 + 1 \leq 0\}$ *and thus* $r_1^1(x) = -x_1^2 + 1$. *Let* $w \in \mathcal{F}\left(\mathbb{P}_{\text{run}}, \overline{\mathbb{R}}\right)$, *then:*

$$
\begin{array}{l}
(F_1^{\mathcal{R}}(w))(q_1) = \\
\inf\limits_{\lambda,\sigma,\mu,\eta} \eta \\
\text{s. t.} \left\{ \begin{array}{l} \eta - (0.687x_1 + 0.558x_2 - 0.0001x_1x_2)^2 - \lambda(q_1)(w(q_1) - x_1^2) \\ -\lambda(q_2)(w(q_2) - x_2^2) - \lambda(p)(w(p) - p(x)) + \mu(x)(1 - x_1^2) = \sigma(x) \\ \lambda \in \mathcal{F}(\mathbb{P}, \mathbb{R}_+),\ \sigma \in \Sigma[x],\ \mu \in \Sigma[x],\ \eta \in \mathbb{R}, \\ \deg(\sigma) \leq 6,\ \deg(\mu) \leq 6\,. \end{array} \right.
\end{array}
$$

In practice, one cannot find any feasible solution of degree less than 6, thus we replace the degree constraint by the more restrictive one: $\deg(\sigma) \leq 6,\ \deg(\mu) \leq 6$.

The computation of $F^\sharp$ requires the approximation of $X^{\text{in}^\dagger} := \sup\{p(x), x \in X^{\text{in}}\}$. Since X^{in} is a basic semi-algebraic set and each template p is a polynomial, then the evaluation of $X^{\text{in}^\dagger}$ boils down to solving a polynomial maximization problem. Next, we use SOS reinforcement described above to over-approximate $X^{\text{in}^\dagger}$ with the set $X^{\text{in}^{\mathcal{R}}}$, defined as follows:

$$
X^{\text{in}^{\mathcal{R}}}(p) := \inf \left\{ \eta \;\middle|\; \begin{array}{c} \eta - p + \langle \nu^{n_{\text{in}}}, r^{n_{\text{in}}} \rangle = \sigma_0, \\ \eta \in \mathbb{R},\ \sigma_0 \in \Sigma[x], \nu^{\text{in}} \in \Sigma[x]^{n_{\text{in}}}, \\ \deg(\sigma_0) \leq 2m, \deg(\langle \nu^{n_{\text{in}}}, r^{n_{\text{in}}} \rangle) \leq 2m \end{array} \right\}\ .
$$

Thus, the value of $X^{\text{in}^{\mathcal{R}}}(p)$ is obtained by solving an SOS optimization problem. Since X^{in} is a nonempty compact basic semi-algebraic set, this problem has a feasible solution (see the proof of [Las01, Th. 4.2]), ensuring that $X^{\text{in}^{\mathcal{R}}}(p)$ is finite valued.

Example 4.3. *The initialization set* X^{in} *of Example 2.1 is* $[-1, 1] \times [-1, 1]$. *It can be written as:* $\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 - 1 \leq 0,\ x_2^2 - 1 \leq 0\}$. *Then, considering the same template basis of Example 4.2 and the template* q_1*:*

$$
X^{\text{in}^{\mathcal{R}}}(q_1) := \inf \left\{ \eta \;\middle|\; \begin{array}{c} \eta - x_1^2 + \nu_1^{n_{\text{in}}}(x)(x_1^2 - 1) + \nu_2^{n_{\text{in}}}(x)(x_2^2 - 1) = \sigma_0(x), \\ \eta \in \mathbb{R},\ \sigma_0 \in \Sigma[x], \nu_1^{\text{in}}, \nu_2^{\text{in}} \in \Sigma[x], \\ \deg(\sigma_0) \leq 6, \deg(\langle \nu_1^{n_{\text{in}}} \rangle \leq 6, \deg(\langle \nu_2^{n_{\text{in}}} \rangle \leq 6 \end{array} \right\}\ .
$$

It is easy to see that taking for all $x \in \mathbb{R}^2$, $\nu_1^{n_{\text{in}}}(x) = 1$ *and for all* $x \in \mathbb{R}^2$, $\nu_2^{n_{\text{in}}}(x) = 0$ *leads to* $\eta - x_1^2 + \nu_1^{n_{\text{in}}}(x)(x_1^2 - 1) + \nu_2^{n_{\text{in}}}(x)(x_2^2 - 1) = \eta - 1 = \sigma_0(x)$. *Thus for* $\eta = 1$ *and for all* $x \in \mathbb{R}^2$, $\sigma_0(x) = 0$, *we obtain* $X^{\text{in}^{\mathcal{R}}}(q_1) \leq 1$. *We will see at Prop. 4.1, that* $X^{\text{in}^\dagger} \leq_{\mathcal{F}} X^{\text{in}^{\mathcal{R}}}$. *Thus, since* $X^{\text{in}^\dagger}(q_1) = \sup\{x_1^2 \mid (x_1, x_2) \in [-1, 1] \times [-1, 1]\} = 1$, *we conclude that* $1 \leq X^{\text{in}^{\mathcal{R}}}(q_1)$ *and* $X^{\text{in}^{\mathcal{R}}}(q_1) = 1$.

Finally, we define the relaxed functional $F^{\mathcal{R}}$ for all $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ for all $p \in \mathbb{P}$ as follows:

$$
(F^{\mathcal{R}}(w))(p) = \sup \left\{ \sup_{i \in \mathcal{I}} (F_i^{\mathcal{R}}(w))(p), X^{\text{in}^{\mathcal{R}}}(p) \right\}\ . \tag{13}
$$

As we followed the construction proposed in Section 4.1, the relaxed functional $F^{\mathcal{R}}$ provides a safe over-approximation of the abstract semantics $F^\sharp$.

Proposition 4.1 (Safety)**.** *The following statements hold:*

1. $X^{\mathrm{in}^\dagger} \leq_{\mathcal{F}} X^{\mathrm{in}_{\mathcal{R}}}$;
2. *For all* $i \in \mathcal{I}$, *for all* $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, $F_i^\sharp(w) \leq_{\mathcal{F}} F_i^{\mathcal{R}}(w)$;
3. *For all* $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, $F^\sharp(w) \leq_{\mathcal{F}} F^{\mathcal{R}}(w)$.

An important property that we will use to prove some results on policy iteration algorithm is the monotonicity of the relaxed functional.

Proposition 4.2 (Monotonicity)**.** 1. *For all* $i \in \mathcal{I}$, $w \mapsto F_i^{\mathcal{R}}(w)$ *is monotone on* $\mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$;

2. *The function* $w \mapsto F^{\mathcal{R}}(w)$ *is monotone on* $\mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$.

Proof. Let $p \in \mathbb{P}$. For $v \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, $\lambda \in \mathcal{F}\left(\mathbb{P}, \mathbb{R}_+\right)$, $\mu \in \Sigma[x]^{n_i}$, $\gamma \in \Sigma[x]^{n_0}$ and $\eta \in \mathbb{R}$ such that $\deg(\langle \mu, r^i \rangle + \langle \gamma, r^0 \rangle) \leq 2m \deg T^i$, we define the polynomial in x, $\psi^{\lambda,\mu,\gamma,\eta}(v) := \eta - p \circ T^i - \sum_{\mathrm{q} \in \mathbb{P}} \lambda(q)(v(q) - q) + \langle \mu, r^i \rangle + \langle \gamma, r^0 \rangle$. We define for $v \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ the set $R(v) = \{(\lambda, \mu, \gamma, \eta) \mid \psi^{\lambda,\mu,\gamma,\eta}(v) \in \Sigma[x],\ \lambda \in \mathcal{F}\left(\mathbb{P}, \mathbb{R}_+\right), \mu \in \Sigma[x]^{n_i}, \gamma \in \Sigma[x]^{n_0}, \eta \in \mathbb{R}\}$. Now, let us take $w, w' \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ such that $w \leq_{\mathcal{F}} w'$. We have:

$$
\begin{array}{rl}
& \psi^{\lambda,\mu,\gamma,\eta}(w) \\
= & \eta - p \circ T^i - \sum_{\mathrm{q} \in \mathbb{P}} \lambda(q)(w(q) - q) + \langle \mu, r^i \rangle + \langle \gamma, r^0 \rangle \\
= & \eta - p \circ T^i - \sum_{\mathrm{q} \in \mathbb{P}} \lambda(q)(w(q) - w'(q) + w'(q) - q) + \langle \mu, r^i \rangle + \langle \gamma, r^0 \rangle \\
= & \eta - p \circ T^i - \sum_{\mathrm{q} \in \mathbb{P}} \lambda(q)(w'(q) - q) + \langle \mu, r^i \rangle + \langle \gamma, r^0 \rangle - \sum_{\mathrm{q} \in \mathbb{P}} \lambda(q)(w(q) - w'(q)) \\
= & \psi^{\lambda,\mu,\gamma,\eta}(w') + \sum_{\mathrm{q} \in \mathbb{P}} \lambda(q)(w'(q) - w(q)) \,.
\end{array}
$$

Then, from $w \leq_{\mathcal{F}} w'$ and the fact that $\lambda(q)$ are nonnegative scalars, if $\psi^{\lambda,\mu,\gamma,\eta}(w')$ is an SOS polynomial, so is $\psi^{\lambda,\mu,\gamma,\eta}(w)$ as a sum of a SOS polynomial and a nonnegative scalar. Hence, we have $R(w') \subseteq R(w)$. Finally, we recall that if $A \subseteq B$, then $\inf_B \leq \inf_A$. We conclude that $(F_i^{\mathcal{R}}(w))(p) \leq (F_i^{\mathcal{R}}(w'))(p)$.

2. The mapping $F^{\mathcal{R}}$ is monotone as supremum of monotone maps. □

From the third assertion of Prop. 4.1, if w satisfies $F^{\mathcal{R}}(w) \leq_{\mathcal{F}} w$ then $F^\sharp(w) \leq_{\mathcal{F}} w$ and from Prop. 3.1, $w^\star$ is an inductive invariant and thus $\mathfrak{R} \subseteq w^\star$. This result is formulated as the following corollary.

Corollary 4.1 (Over-approximation)**.** *For all* $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ *such that* $F^{\mathcal{R}}(w) \leq_{\mathcal{F}} w$ *then* $\mathfrak{R} \subseteq w^\star$.

5 Policy Iteration in Polynomial Templates Abstract Domains

We are interested in computing the least fixpoint $\mathfrak{R}^{\mathcal{R}}$ of $F^{\mathcal{R}}$, $\mathfrak{R}^{\mathcal{R}}$ being an over-approximation of $\mathfrak{R}$ (least fixpoint of F). As for the definition of $\mathfrak{R}$, it can be reformulated using Tarski's theorem as the minimal post-fixpoint:

$$\mathfrak{R}^{\mathcal{R}} = \min\{w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right) \mid F^{\mathcal{R}}(w) \leq_{\mathcal{F}} w\} \,.$$

The idea behind policy iteration is to over-approximate $\mathfrak{R}^{\mathcal{R}}$ using successive iterations which are composed of

- the computation of polynomial template bounds using linear programming,
- the determination of new policies using SOS programming,

until a fixpoint is reached. Policy iteration navigates in the set of post-fixpoints of $F^{\mathcal{R}}$ and needs to start from a post-fixpoint w^0 known *a-priori*. It acts like a narrowing operator and can be interrupted at any time. For further information on policy iteration, the interested reader can consult [CGG+05, GGTZ07].

5.1 Policies

Policy iteration can be used to compute a fixpoint of a monotone self-map defined as an infimum of a family of affine monotone self-maps. In this paper, we propose to design a policy iteration algorithm to compute a fixpoint of $F^{\mathcal{R}}$. In this subsection, we give the formal definition of policies in the context of polynomial templates and define the family of affine monotone self-maps. We do not apply the concept of policies on $F^{\mathcal{R}}$ but on the functions $F_i^{\mathcal{R}}$ exploiting the fact that for all $i \in \mathcal{I}$, $F_i^{\mathcal{R}}$ is the optimal value of a minimization problem.

Policy iteration needs a *selection property*, that is, when an element $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ is given, there exists a policy which achieves the infimum. In our context, since we apply the concept of policies to $F_i^{\mathcal{R}}$, it means that the minimization problem involved in the computation of $F_i^{\mathcal{R}}$ has an optimal solution. In our case, for $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and $p \in \mathbb{P}$, an optimal solution is a vector $(\lambda, \sigma, \mu, \gamma) \in \mathcal{F}\left(\mathbb{P}, \mathbb{R}_+\right) \times \Sigma[x] \times \Sigma[x]^{n_i} \times \Sigma[x]^{n_0}$ such that, using (12), we obtain:

$$\begin{array}{c} \left(F_i^{\mathcal{R}}(w)\right)(p) = p \circ T^i + \displaystyle\sum_{q \in \mathbb{P}} \lambda(q)(w(q) - q) - \langle \mu, r^i \rangle - \langle \gamma, r^0 \rangle + \sigma \\ \text{and } \deg(\sigma) \leq 2m \deg T^i,\ \deg(\langle \mu, r^i \rangle + \langle \gamma, r^0 \rangle) \leq 2m \deg T^i \end{array} . \tag{14}$$

Observe that in Eq. (14), $\left(F_i^{\mathcal{R}}(w)\right)(p)$ is a scalar whereas the right-hand-side is a polynomial. The equality in this equation means that this polynomial is a constant polynomial. Then we introduce the set of feasible solutions for the SOS problem $\left(F_i^{\mathcal{R}}(w)\right)(p)$:

$$\mathrm{Sol}(w, i, p) = \{(\lambda, \sigma, \mu, \gamma) \in \mathcal{F}\left(\mathbb{P}, \mathbb{R}_+\right) \times \Sigma[x] \times \Sigma[x]^{n_i} \times \Sigma[x]^{n_0} \mid \text{Eq. (14) holds}\} . \tag{15}$$

Since policy iteration algorithm can be stopped at any step and still provides a sound over-approximation, we stop the iteration when $\mathrm{Sol}(w, i, p) = \emptyset$. Now, we are interested in the elements $w \in \mathcal{F}\left(\mathbb{P}, \mathbb{R}\right)$ such that $\mathrm{Sol}(w, i, p)$ is non-empty:

$$\mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right) = \{w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right) \mid \forall\, i \in \mathcal{I},\ \forall\, p \in \mathbb{P},\ \mathrm{Sol}(w, i, p) \neq \emptyset\} . \tag{16}$$

The notation $\mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ was introduced in [AGG12] to define the elements $w \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ satisfying $\mathrm{Sol}(w, i, p) \neq \emptyset$. In [AGG12, Section 4.3], we could ensure that $\mathrm{Sol}(w, i, p) \neq \emptyset$ using Slater's constraint qualification condition. In the current nonlinear setting, we cannot use the same condition, which yields a more complicated definition for $\mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$.

Finally, we can define a policy as a map which selects, for all $w \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, for all $i \in \mathcal{I}$ and for all $p \in \mathbb{P}$ a vector of $\mathrm{Sol}(w, i, p)$. More formally, we have the following definition:

Definition 5.1 (Policies in the policy iteration SOS based setting)**.** *A policy is a map* $\pi : \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right) \mapsto ((\mathcal{I} \times \mathbb{P}) \mapsto \mathcal{F}(\mathbb{P}, \mathbb{R}_+) \times \Sigma[x] \times \Sigma[x]^{n_i} \times \Sigma[x]^{n_0})$ *such that:* $\forall\, w \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, $\forall\, i \in \mathcal{I}$, $\forall\, p \in \mathbb{P}$, $\pi(w)(i,p) \in \mathrm{Sol}(w,i,p)$.

We denote by Π the set of policies. For $\pi \in \Pi$, let us define π_λ as the map from $\mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ to $(\mathcal{I} \times \mathbb{P}) \mapsto \mathcal{F}(\mathbb{P}, \mathbb{R}_+)$ which associates with $w \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and $(i,p) \in \mathcal{I} \times \mathbb{P}$ the first element of $\pi(w)(i,p)$ i.e. if $\pi(w)(i,p) = (\lambda, \sigma, \mu, \gamma)$ then $\pi_\lambda(w)(i,p) = \lambda$. The equality $\pi_\lambda(w)(i,p) = \lambda$ means that when we perform the policy iterations algorithm, we select the vector of Lagrange multipliers λ associated with the constraints of the form $q(x) \leq w(q)$. The purpose of this selection is to update the value of w using the direction λ. The other coordinates composing $\pi(w)(i,p)$ that is σ, μ, γ do not serve the policy iterations algorithm but are only used to take in consideration the sets X^i and X^0 in the computation of $F_i^{\mathcal{R}}(w)(p)$.

As said before, policy iteration exploits the linearity of maps when a policy is fixed. We have to define the affine maps we will use in a policy iteration step. With $\pi \in \Pi$, $w \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, $i \in \mathcal{I}$ and $p \in \mathbb{P}$ and $\lambda = \pi_\lambda(w)(i,p)$, let us define the map $\varphi^\lambda_{w,i,p} : \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right) \mapsto \overline{\mathbb{R}}$ as follows:

$$v \mapsto \varphi^\lambda_{w,i,p}(v) = \sum_{q \in \mathbb{P}} \lambda(q) v(q) + (F_i^{\mathcal{R}}(w))(p) - \sum_{q \in \mathbb{P}} \lambda(q) w(q)\,. \tag{17}$$

Then, for $\pi \in \Pi$, we define for all $w \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, the map $\Phi_w^{\pi(w)}$ from $\mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right) \mapsto \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$. Let $v \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and $p \in \mathbb{P}$:

$$\Phi_w^{\pi(w)}(v)(p) = \sup\left\{ \sup_{i \in \mathcal{I}} \varphi^\lambda_{w,i,p}(v), X^{\mathrm{in}\mathcal{R}}(p) \right\}. \tag{18}$$

Example 5.1. *Let us consider Example 4.2 and the function* $w^0(q_1) = w^0(q_2) = 2.1391$ *and* $w^0(p) = 0$*. Then there exists two SOS polynomials* $\overline{\mu}$ *and* $\overline{\sigma}$ *such that, for all* $x \in \mathbb{R}^d$*:*

$$\begin{aligned}(F_1^{\mathcal{R}}(w))(q_1) = \;& (0.687x_1 + 0.558x_2 - 0.0001x_1x_2)^2 + \overline{\lambda}(q_1)(2.1391 - x_1^2) \\ & + \overline{\lambda}(q_2)(2.1391 - x_2^2) - \overline{\lambda}(p)p(x) - \overline{\mu}(x)(1 - x_1^2) + \overline{\sigma}(x) \\ = \;& 1.5503\,,\end{aligned}$$

with $\overline{\lambda}(q_1) = \overline{\lambda}(q_2) = 0$ *and* $\overline{\lambda}(p) = 2.0331$*. It means that* $\overline{\lambda}$*,* $\overline{\mu}$ *and* $\overline{\sigma}$ *are computed such that* $(0.687x_1 + 0.558x_2 - 0.0001x_1x_2)^2 + \overline{\lambda}(q_1)(2.1391 - x_1^2) + \overline{\lambda}(q_2)(2.1391 - x_2^2) - \overline{\lambda}(p)p(x) - \overline{\mu}(x)(1 - x_1^2) + \overline{\sigma}(x)$ *is actually a constant polynomial.*

Then $(\overline{\lambda}, \overline{\mu}, \overline{\sigma}) \in \mathrm{Sol}(w^0, 1, q_1)$ *and we can define a policy* $\pi(w^0)$ *such that* $\pi(w^0)(1,q_1) = (\overline{\lambda}, \overline{\mu}, \overline{\sigma})$ *and thus* $\pi_\lambda(w^0)(1,q_1) = (0,0,2.0331)$*. We can thus define for* $v \in \mathcal{F}(\mathbb{P}_{\mathrm{run}}, \mathbb{R})$*, the affine mapping:* $\varphi^\lambda_{w^0,1,q_1}(v) = \lambda(q_1)v(q_1) + \lambda(q_2)v(q_2) + \lambda(p)v(p) + (F_1^{\mathcal{R}}(w))(q_1) - \lambda(q_1)w(q_1) - \lambda(q_2)w(q_2) - \lambda(p)w(p) = 2.1391v(p) + 1.5503$.

Let us denote by $\mathcal{F}(\mathbb{P}, \mathbb{R})$ the set of finite valued function on $\mathbb{P}$ i.e $g \in \mathcal{F}(\mathbb{P}, \mathbb{R})$ iff $g(p) \in \mathbb{R}$ for all $p \in \mathbb{P}$.

Proposition 5.1 (Properties of $\varphi^\lambda_{i,w,p}$)**.** *Let* $\pi \in \Pi$*,* $w \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ *and* $(i,p) \in \mathcal{I} \times \mathbb{P}$*. Let us write* $\lambda = \pi_\lambda(w)(i,p)$*. The following properties are true:*

1. $\varphi^{\lambda}_{w,i,p}$ *is affine on* $\mathcal{F}(\mathbb{P},\mathbb{R})$ *;*

2. $\varphi^{\lambda}_{w,i,p}$ *is monotone on* $\mathcal{F}\left(\mathbb{P},\overline{\mathbb{R}}\right)$ *;*

3. $\forall\, v \in \mathcal{F}\left(\mathbb{P},\overline{\mathbb{R}}\right)$, $F^{\mathcal{R}}_i(v)(p) \leq \varphi^{\lambda}_{w,i,p}(v)$ *;*

4. $\varphi^{\lambda}_{w,i,p}(w) = F^{\mathcal{R}}_i(w)(p)$ *.*

Proof. Let $w \in \mathcal{FS}\left(\mathbb{P},\overline{\mathbb{R}}\right)$, $i \in \mathcal{I}$, $p \in \mathbb{P}$ and $\pi \in \Pi$.

1. The fact that $\varphi^{\pi_\lambda(w)(i,p)}_{w,i,p}$ is affine follows readily from the definition (Eq. (17)).

2. The monotonicity of $\varphi^{\pi_\lambda(w)(i,p)}_{w,i,p}$ follows from the nonnegativity of $\pi_\lambda(w)(i,p)$.

3. Let $v \in \mathcal{F}\left(\mathbb{P},\overline{\mathbb{R}}\right)$. Since $w \in \mathcal{FS}\left(\mathbb{P},\overline{\mathbb{R}}\right)$, there exists $(\lambda,\sigma,\mu,\gamma) \in \mathcal{F}(\mathbb{P},\mathbb{R}_+) \times \Sigma[x] \times \Sigma[x]^{n_i} \times \Sigma[x]^{n_0}$ such that $\deg(\sigma) \leq 2m \deg T^i,\ \deg(\langle \mu, r^i\rangle + \langle \gamma, r^0\rangle) \leq 2m \deg T^i$:

$$\left(F^{\mathcal{R}}_i(w)\right)(p) = p \circ T^i + \sum_{q\in\mathbb{P}} \lambda(q)(w(q)-q) - \langle \mu, r^i\rangle - \langle \gamma, r^0\rangle + \sigma\,.$$

Writing $\lambda = \pi_\lambda(w)(i,p)$, we get:

$$\begin{aligned}\varphi^{\lambda}_{w,i,p}(v) &= \sum_{q\in\mathbb{P}} \lambda(q)v(q) - \sum_{q\in\mathbb{P}} \lambda(q)w(q) + p\circ T^i + \sum_{q\in\mathbb{P}} \lambda(q)(w(q)-q)\\ &\quad -\langle \mu, r^i\rangle - \langle \gamma, r^0\rangle + \sigma\\ &= p\circ T^i + \sum_{q\in\mathbb{P}} \lambda(q)(v(q)-q) - \langle \mu, r^i\rangle - \langle \gamma, r^0\rangle + \sigma\,.\end{aligned}$$

Finally,

$$\varphi^{\lambda}_{w,i,p}(v) - p\circ T^i - \sum_{q\in\mathbb{P}} \lambda(q)(v(q)-q) + \langle \mu, r^i\rangle + \langle \gamma, r^0\rangle = \sigma\,, \tag{19}$$

and recall that (Eq. (14))

$$\left(F^{\mathcal{R}}_i(w)\right)(p) = \inf_{\lambda,\sigma,\mu,\gamma,\eta} \eta \quad \text{s.t.} \begin{cases} \eta - p\circ T^i - \sum_{q\in\mathbb{P}} \lambda(q)(w(q)-q) + \langle \mu, r^i\rangle + \langle \gamma, r^0\rangle = \sigma \\ \lambda \in \mathcal{F}(\mathbb{P},\mathbb{R}_+)\,,\ \sigma\in\Sigma[x],\ \mu\in\Sigma[x]^{n_i},\ \gamma\in\Sigma[x]^{n_0},\ \eta\in\mathbb{R}\,, \\ \deg(\sigma) \leq 2m\deg T^i,\ \deg(\langle \mu, r^i\rangle + \langle \gamma, r^0\rangle) \leq 2m\deg T^i\,. \end{cases}$$

From Eq. (19), $(\lambda,\sigma,\mu,\gamma,\varphi^{\lambda}_{w,i,p}(v))$ is a feasible solution of the latter minimization problem and we conclude that $\left(F^{\mathcal{R}}_i(v)\right)(p) \leq \varphi^{\pi_\lambda(w)(i,p)}_{w,i,p}(v)$.

4.
$$\varphi^{\pi_\lambda(w)(i,p)}_{w,i,p}(w) = \sum_{q\in\mathbb{P}} \lambda(q)w(q) + \left(F^{\mathcal{R}}_i(w)\right)(p) - \sum_{q\in\mathbb{P}} \lambda(q)w(q) = \left(F^{\mathcal{R}}_i(w)\right)(p)\ .$$

□

The properties presented in Prop. 5.1 imply some useful properties for the maps $\Phi^{\pi(w)}_w$.

Proposition 5.2 (Properties of $\Phi^{\pi(w)}_w$)**.** *Let* $\pi \in \Pi$ *and* $w \in \mathcal{FS}\left(\mathbb{P},\overline{\mathbb{R}}\right)$*. The following properties are true:*

1. $\Phi_w^{\pi(w)}$ *is monotone on* $\mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ *;*

2. $F^{\mathcal{R}} \leq_{\mathcal{F}} \Phi_w^{\pi(w)}$ *;*

3. $\Phi_w^{\pi(w)}(w) = F^{\mathcal{R}}(w)$ *;*

4. *Suppose that the least fixpoint of* $\Phi_w^{\pi(w)}$ *is* $L \in \mathcal{F}(\mathbb{P}, \mathbb{R})$*. Then* L *can be computed as the unique optimal solution of the linear program:*

$$\inf \left\{ \sum_{p' \in \mathbb{P}} v(p') \mid \forall (i,p) \in \mathcal{I} \times \mathbb{P},\ \varphi_{i,w,p}^{\pi_\lambda(w)(i,p)}(v) \leq v(p),\ \forall q \in \mathbb{P}, X^{\text{in}\mathcal{R}}(q) \leq v(q) \right\}. \tag{20}$$

LP problem (20) corresponds exactly to the linear program presented in the case of quadratic templates [AGG12, Eq. 4.4].

Proof. Let $\pi \in \Pi$ and $w \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$.

1. The map $\Phi_w^{\pi(w)}$ is monotone as the map $\varphi_{w,i,p}^{\pi_\lambda(w)(i,p)}$ is monotone for all $i \in \mathcal{I}$ and for all $p \in \mathbb{P}$, and the the fact that the point-wise supremum of monotone maps is also monotone.

2. Let $v \in \mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and let $p \in \mathbb{P}$. Recall that:

$$\left(F^{\mathcal{R}}(v)\right)(p) = \sup \left\{ \sup_{i \in \mathcal{I}} \left(F_i^{\mathcal{R}}(v)\right)(p), X^{\text{in}\mathcal{R}}(p) \right\},$$

and from the third assertion of Prop. 5.1, we have for all $i \in \mathcal{I}$, $F_i^{\mathcal{R}}(v)(p) \leq \varphi_{w,i,p}^{\pi_\lambda(w)(i,p)}$, by taking the supremum over $\mathcal{I}$ and then the supremum with $X^{\text{in}\mathcal{R}}(p)$, we obtain that $F^{\mathcal{R}}(v)(p) \leq \Phi_w^{\pi(w)}(v)(p)$, yielding the desired result.

3. This result follows readily from the fourth assertion of Prop. 5.1 and the definition of $\Phi_w^{\pi(w)}$ (Eq. (18)).

4. By Tarski's theorem and as $\Phi_w^{\pi(w)}$ is monotone, $\Phi_w^{\pi(w)}$ has a least fixpoint in $\mathcal{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$. Let L be this least fixpoint supposed to be finite valued. Now, from Tarski's theorem and the definition of $\Phi_w^{\pi(w)}$, we have:

$$\begin{aligned} L &= \inf\{v \mid \Phi_w^{\pi(w)}(v) \leq_{\mathcal{F}} v\} \\ &= \inf\left\{v \mid \forall (i,p) \in \mathcal{I} \times \mathbb{P},\ \varphi_{i,w,p}^{\pi_\lambda(w)(i,p)}(v) \leq v(p),\ \forall q \in \mathbb{P}, X^{\text{in}\mathcal{R}}(q) \leq v(q)\right\}. \end{aligned}$$

Let us suppose that there exists a feasible solution $\bar{v}$ such that $\sum_{q \in \mathbb{P}} \bar{v}(q) < \sum_{q \in \mathbb{P}} L(q)$. Note that since $X^{\text{in}\mathcal{R}} \leq_{\mathcal{F}} \bar{v}$, $\sum_{q \in \mathbb{P}} \bar{v}$ is finite. Then we have $\inf\{\bar{v}, L\} \leq L$ and $\inf\{\bar{v}, L\} \neq L$. As $\Phi_w^{\pi(w)}$ is monotone and as $\bar{v}$ and L are feasible, we have $\Phi_w^{\pi(w)}(\inf\{\bar{v}, L\}) \leq \inf\{\bar{v}, L\}$. This contradicts the minimality of L. We conclude that L is the optimal solution of Linear Program (20). □

Remark 1. *We recall that the linear constraints in Problem* (20) *come from the use of the function defined at Equation* (17) *which is affine on the variable* v*. The linear forms are defined from the vector of Lagrange multipliers* λ *found when we solve the minimization problem involved in Equation* (12)*. If we had allowed a vector of SOS polynomials* λ *as vector of Lagrange multipliers,*

we would obtain a set of polynomial inequalities that we would solve using SOS programming. The resulted problem would not have a feasible solution.

For example, let us consider an SOS polynomial template p, an SOS (non scalar) polynomial λ and a scalar c. Then, in this case, an analog of Problem (12) *would be:*

$$\min\{v(p) \in \mathbb{R} \mid \lambda(x)v(p) + c \leq v(p), \forall\, x \in \mathbb{R},\ v(p) \geq X^{\text{in}^\dagger}(p)\}$$

We assumed that p is a SOS polynomial template, implying that $X^{\text{in}^\dagger}(p)$ is strictly positive. Since $\lambda(x)$ is a non scalar SOS polynomial and $v(p) > 0$, then $v(p)(1 - \lambda(x)) - c$ is negative for some sufficiently large x. This proves the infeasibility of the problem.

Recall that a function $g : \mathbb{R}^d \mapsto \mathbb{R}$ is upper-semicontinuous at x iff for all $(x_n)_{n\in\mathbb{N}}$ converging to x, then $\limsup_{n\to+\infty} g(x_n) \leq g(x)$.

Proposition 5.3. *Let $p \in \mathbb{P}$. Then $w \mapsto F^{\mathcal{R}}(w)(p)$ is upper-semicontinuous on $\mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right) \cap \mathcal{F}(\mathbb{P}, \mathbb{R})$.*

Proof. Let $\pi \in \Pi$, $w \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right) \cap \mathcal{F}(\mathbb{P}, \mathbb{R})$ and $p \in \mathbb{P}$. Let $i \in \mathcal{I}$. Let $(w_n)_{n\in\mathbb{N}}$ be a sequence of elements of $\mathcal{F}(\mathbb{P}, \mathbb{R})$ converging to w. Let $\lambda = \pi_\lambda(w)(i, p)$. Since $\varphi^{\lambda}_{i,w,p}$ is affine on $\mathcal{F}(\mathbb{P}, \mathbb{R})$, then $\varphi^{\lambda}_{i,w,p}$ is continuous on $\mathcal{F}(\mathbb{P}, \mathbb{R})$ and finally $v \mapsto \Phi^{\pi(w)}_w(v)(p)$ is continuous on $\mathcal{F}(\mathbb{P}, \mathbb{R})$ as a finite supremum of continuous functions on $\mathcal{F}(\mathbb{P}, \mathbb{R})$. Then from the second point of Prop. 5.2, for all $n \in \mathbb{N}$, $F^{\mathcal{R}}(w_n)(p) \leq \Phi^{\pi(w)}_w(w_n)(p)$. By taking the lim sup, we obtain: $\limsup_{n\to+\infty} F^{\mathcal{R}}(w_n)(p) \leq \limsup_{n\to+\infty} \Phi^{\pi(w)}_w(w_n)(p) = \Phi^{\pi(w)}_w(w)(p) = F^{\mathcal{R}}(v)(p)$. □

5.2 Policy Iteration

Next, we describe the policy iteration algorithm. We suppose that we have a post-fixpoint w^0 of $F^{\mathcal{R}}$ in $\mathcal{F}(\mathbb{P}, \mathbb{R})$.

We detail step by step the algorithm presented in Figure 4. At Line 1, the algorithm is initialized and thus $k = 0$. At Line 4, we compute $F^{\mathcal{R}}(w^k)$ using Eq. (13) and solve the SOS problem involved in Eq. (12). At Line 6, if for all $i \in \mathcal{I}$ and for $p \in \mathbb{P}$, the SOS problem involved in Eq. (12) has an optimal solution, then a policy π is available and we can choose any optimal solution of SOS problem involved in Eq. (12) as policy. If an optimal solution does not exist then the algorithm stops and return w^k. Now, if a policy π has been defined, the algorithm goes to Line 12 and we can define $\Phi^{\pi(w^k)}_{w^k}$ following Eq. (18). Then, we solve LP problem (20) and define the new bound on templates w^{k+1} as the smallest fixpoint of $\Phi^{\pi(w^k)}_{w^k}$. Finally, at Line 13, k is incremented.

If for some $k \in \mathbb{N}$, $w^k \notin \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and $w^{k-1} \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ then the algorithm stops and returns w^k. Hence, we set for all $l \geq k$, $w^l = w^k$.

Theorem 5.1 (Convergence result of the algorithm presented in Figure 4)**.** *The following statements hold:*

1. *For all $k \in \mathbb{N}$, $w^k \in \mathcal{F}(\mathbb{P}, \mathbb{R})$ and $F^{\mathcal{R}}(w^k) \leq w^k$;*
2. *The sequence $(w^k)_{k\geq 0}$ generated by Algorithm 4 is decreasing and converges;*

input : $w^0 \in \mathcal{F}(\mathbb{P}, \mathbb{R})$, a post-fixpoint of $F^{\mathcal{R}}$
output: a fixpoint $w = F^{\mathcal{R}}(w)$ if $\forall\, k \in \mathbb{N}$, $w^k \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ or a post-fixpoint otherwise

```
k=0;
while fixpoint not reached do
    begin compute the next policy π for the current iterate w^k
        Compute F^R(w^k) using Eq. (13) and Eq. (12);
        if w^k ∈ FS(P, R̄) then
            Define π(w^k) ;
        else
            return w^k;
        end
    end
    begin compute the next iterate w^{k+1}
        Define Φ^{π(w^k)}_{w^k} and compute the least fixpoint w^{k+1} of Φ^{π(w^k)}_{w^k} from Problem (20);
        k=k+1;
    end
end
```

Figure 4: SOS-based policy iteration algorithm for PPS programs.

3. *Let* $w^\infty = \lim_{k\to+\infty} w^k$, *then* $F^{\mathcal{R}}(w^\infty) \leq w^\infty$. *Furthermore, if for all* $k \in \mathbb{N}$, $w^k \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ *and if* $w^\infty \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ *then* $F^{\mathcal{R}}(w^\infty) = w^\infty$.

Proof. 1. We reason by induction. We have $F^{\mathcal{R}}(w^0) \leq w^0$ and $w^0 \in \mathcal{F}(\mathbb{P}, \mathbb{R})$ by assumption. Now suppose that for some $k \in \mathbb{N}$, $F^{\mathcal{R}}(w^k) \leq w^k$ and $w^k \in \mathcal{F}(\mathbb{P}, \mathbb{R})$. If $w^k \notin \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ then $w^l = w^k$ for all $l \geq k$ and then we have proved the result. Now suppose that $w^k \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and let us take $\pi \in \Pi$ such that $\Phi^{\pi(w^k)}_{w^k}(w^k) = F^{\mathcal{R}}(w^k)$. From induction property $\Phi^{\pi(w^k)}_{w^k}(w^k) \leq w^k$ and thus w^k is a post-fixpoint of $\Phi^{\pi(w^k)}_{w^k}$ belonging to $\mathcal{F}(\mathbb{P}, \mathbb{R})$. Since every post-fixpoint of $\Phi^{\pi(w^k)}_{w^k}$ is greater than $X^{\mathrm{in}_{\mathcal{R}}}$ then least fixpoint of $\Phi^{\pi(w^k)}_{w^k}$ is finite valued and thus it is the optimal solution w^{k+1} of Problem (20). Moreover from the second point of Prop. 5.2, $F^{\mathcal{R}}(w^{k+1}) \leq_{\mathcal{F}} \Phi^{\pi(w^k)}_{w^k}(w^{k+1})$ and since w^{k+1} is the least fixpoint of $\Phi^{\pi(w^k)}_{w^k}(w^{k+1})$ then $F^{\mathcal{R}}(w^{k+1}) \leq w^{k+1}$. This completes the proof and for all $k \in \mathbb{N}$, $w^k \in \mathcal{F}(\mathbb{P}, \mathbb{R})$ and $F^{\mathcal{R}}(w^k) \leq w^k$.

2. Let $k \in \mathbb{N}$. If $w^k \notin \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ then $w^{k+1} = w^k \leq w^k$. Now suppose that $w^k \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and let $\pi \in \Pi$ such that $\Phi^{\pi(w^k)}_{w^k}(w^k) = F^{\mathcal{R}}(w^k)$, then from the third point of Prop. 5.2, $\Phi^{\pi(w^k)}_{w^k}(w^k) = F^{\mathcal{R}}(w^k) \leq w^k$; the inequality results from the first assertion. Then w^k is a post-fixpoint of $\Phi^{\pi(w^k)}_{w^k}$. Since w^{k+1} is the least fixpoint of $\Phi^{\pi(w^k)}_{w^k}$ and $\Phi^{\pi(w^k)}_{w^k}$ is monotone then from Tarski's theorem $w^{k+1} \leq w^k$. From the first point, for all $k \in \mathbb{N}$, $w^k \in \mathcal{F}(\mathbb{P}, \mathbb{R})$. Moreover by definition of $F^{\mathcal{R}}$, $X^{\mathrm{in}_{\mathcal{R}}} \leq F^{\mathcal{R}}(w^k)$ for all $k \in \mathbb{N}$, then from the first point $(w^k)_{k\geq 0}$ is lower bounded then it converges to some w^∞.

3. If for some k, $w^k \notin \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and $w^{k-1} \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, then $w^\infty = w^k$ and we have $F^{\mathcal{R}}(w^\infty) \leq w^\infty$ from the first point. Now suppose that for all $k \in \mathbb{N}$, $w^k \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$. Since

$F^{\mathcal{R}}$ is monotone then for all $k \in \mathbb{N}$, $F^{\mathcal{R}}(w^\infty) \leq F^{\mathcal{R}}(w^k) \leq w^k$ from the first point. Now taking the limit of the right-hand side, we get $F(w^\infty) \leq w^\infty$. Now, let $k \in \mathbb{N}$ and let $\pi \in \Pi$ such that $\Phi_{w^k}^{\pi(w^k)}(w^k) = F^{\mathcal{R}}(w^k)$. From the second point, $w^{k+1} \leq w^k$ and from the monotonicity of $\Phi_{w^k}^{\pi(w^k)}$, we have $w^{k+1} = \Phi_{w^k}^{\pi(w^k)}(w^{k+1}) \leq \Phi_{w^k}^{\pi(w^k)}(w^k) = F^{\mathcal{R}}(w^k)$. By taking the lim sup on k, we get $w^\infty \leq \limsup_{k\to+\infty} F^{\mathcal{R}}(w^k)$. As $F^{\mathcal{R}}$ is upper-semicontinuous on $\mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right) \cap \mathcal{F}\left(\mathbb{P}, \mathbb{R}\right)$ then, if $w^\infty \in \mathcal{FS}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, $w^\infty \leq \limsup_{k\to+\infty} F^{\mathcal{R}}(w^k) \leq F^{\mathcal{R}}(w^\infty)$ and so $w^\infty = F^{\mathcal{R}}(w^\infty)$. □

5.3 Initialization and templates choice

In Section 3, we have made the assumption that the template basis was given by an oracle. Moreover, in Algorithm 4, we suppose that we have a post-fixpoint $w^0 \in \mathcal{F}\left(\mathbb{P}, \mathbb{R}\right)$ of $F^{\mathcal{R}}$. Now, we give details about the templates basis choice and the computation of a post-fixpoint $w^0 \geq w^\infty$. The templates basis choice relies on the computation of a template basis composed of one element. This single template is constructed by the method developed in [AGM15] and is then completed using the strategy proposed in [AGM15, Ex. 9]. The single template computation also permits us to compute w^0. Actually, the method developed in [AGM15] is constructed by using the definition of being a post-fixpoint of $F^{\mathcal{R}}$. Indeed, suppose that the templates basis is constituted of one template p then w^0 is a post-fixpoint $F^{\mathcal{R}}$ if and only if $F^{\mathcal{R}}(w^0)(p) \leq w^0(p)$. This is equivalent to:

$$X^{\mathrm{in}\mathcal{R}} = \inf\{\eta \mid \eta - p + \sum_{j=1}^{n_{\mathrm{in}}} \nu_j^{\mathrm{in}} r_j^{\mathrm{in}} \in \Sigma[x],\ \nu^{\mathrm{in}} \in \Sigma[x]^{n_{\mathrm{in}}}\} \leq w^0\,,$$

and for all $i \in \mathcal{I}$:

$$(F_i^{\mathcal{R}}(w^0))(p) = \inf_{\lambda,\mu,\gamma,\eta} \eta \quad \text{s.t.} \begin{cases} \eta - p \circ T^i - \lambda^i(w^0 - p) + \langle \mu, r^i \rangle + \langle \gamma, r^0 \rangle \in \Sigma[x] \\ \lambda^i \geq 0,\ \mu \in \Sigma[x]^{n_i},\ \gamma \in \Sigma[x]^{n_0},\ \eta \in \mathbb{R} \end{cases} \leq w^0\,.$$

By definition of the infimum, it is equivalent to the existence of $\nu^{\mathrm{in}} \in \Sigma[x]$ and for all $i \in \mathcal{I}$ of $\lambda^i \geq 0$, $\mu^i \in \Sigma[x]^{n_i}$, $\gamma^i \in \Sigma[x]^{n_0}$ such that:

$$\begin{array}{c} w^0 - p + \sum_{j=1}^{n_{\mathrm{in}}} \nu_j^{\mathrm{in}} r_j^{\mathrm{in}} \in \Sigma[x] \\ w^0 - p \circ T^i - \lambda^i(w^0 - p) + \langle \mu, r^i \rangle + \langle \gamma, r^0 \rangle \in \Sigma[x]\,. \end{array} \tag{21}$$

Now to find a template, it suffices to find p such that Eq. (21) holds. However, the following two issues remain.

First, without an objective function, $p = 0$ is a solution of Eq. (21). A workaround to avoid this trivial solution consists of optimizing a certain objective function under the constraints given in Eq. (21). In [AGM15], a similar optimization procedure (Problem (13) of [AGM15]) is used to prove a property of the form $\mathfrak{R} \subseteq \{x \in \mathbb{R}^d \mid \kappa(x) \leq \alpha\}$, for a given real-valued function κ. Here, we are interested in proving the boundedness of the reachable value set, which corresponds to minimize α with $\kappa = \|\cdot\|_2^2$.

Second, finding λ^i and p satisfying Eq. (21) boils down to solving a bilinear SOS problem, which is not easy to handle in practice. Thus, we fix $\lambda^i = 1$ as in Lyapunov equations. We also take

$w^0 = 0$ since p has a constant part. Finally, to obtain a template p, we solve the following SOS problem:

$$
\begin{array}{rl}
\inf\limits_{p\in\mathbb{R}[x]_{2m},w\in\mathbb{R}} & w \ , \\
\text{s.t.} & -p = \sigma_0 - \sum\limits_{j=1}^{n_{\text{in}}} \sigma_j r_j^{\text{in}} \ , \\
& \forall\, i \in \mathcal{I},\ p - p\circ T^i = \sigma^i - \sum\limits_{j=1}^{n_i} \mu_j^i r_j^i - \sum\limits_{j=1}^{n_0} \gamma_j^i r_j^0 \ , \\
& w + p - \|\cdot\|_2^2 = \psi \ , \\
& \forall\, j = 1,\dots,n_{\text{in}} \ ,\ \sigma_j \in \Sigma[x] \ ,\ \deg(\sigma_j r_j^{\text{in}}) \le 2m \ , \\
& \sigma_0 \in \Sigma[x] \ ,\ \deg(\sigma_0) \le 2m \ , \\
& \forall\, i \in \mathcal{I} \ ,\ \sigma^i \in \Sigma[x] \ ,\ \deg(\sigma^i) \le 2m \deg T^i \ , \\
& \forall\, i \in \mathcal{I} \ ,\ \forall\, j = 1,\dots,n_i \ ,\ \mu_j^i \in \Sigma[x] \ ,\ \deg(\mu_j^i r_j^i) \le 2m \deg T^i \ , \\
& \forall\, i \in \mathcal{I} \ ,\ \forall\, j = 1,\dots,n_0 \ ,\ \gamma^i \in \Sigma[x] \ ,\ \deg(\gamma_j^i r_j^0) \le 2m \deg T^i \ , \\
& \psi \in \Sigma[x] \ ,\ \deg(\psi) \le 2m \ .
\end{array}
\tag{22}
$$

Let (p, w) be a solution of Problem (22). In [AGM15, Prop. 1], we proved that the set $\{x \in \mathbb{R}^d \mid p(x) \le 0\}$ defines an inductive invariant. To complete the template basis, we use the strategy proposed in [AGM15, Ex. 9], that is, we work with the templates basis $\{x \mapsto x_i^2, i \in [d]\} \cup \{p\}$. We thus use the inductive invariant set $\{p(x) \le 0, x_i^2 \le w\}$ as initialization i.e. the initial bound is $w^0(q) = w$ if $q \neq p$ and $w^{(0)}(q) = 0$ if $q = p$. As opposed to the approach of [AGM15], we avoid increasing the degree of polynomial p to obtain better bounds on the reachable values set.

Computational considerations The number of (a-priori unknown) coefficients of the polynomial p (of degree $2m$ and d variables) appearing in Problem 22 is $\binom{2m+d}{d}$. Similarly, the number of coefficients of each σ^i (resp. μ_j^i and γ_j^i) is $\binom{2m\deg T^i+d}{d}$. Thus, Problem 22 can be reformulated as an SDP program involving $\binom{2m+d}{d} + \sum_{i\in\mathcal{I}}[1 + n_i + n_0]\binom{2m\deg T^i+d}{d}$ SDP variables. Therefore, our framework is expected to be tractable when either d or m is small. As mentioned in [AGM15, Section 4], one could address bigger instances while exploiting sparsity properties of the initial system, as in [WKKM06].

6 Experiments

6.1 Details of the running Example.

Recall that our running example is given by the following PPS: $(X^{\text{in}}, X^0, \{X^1, X^2\}, \{T^1, T^2\})$, where:

$$
\begin{array}{l}
X^{\text{in}} = [-1,1]\times[-1,1] \\
X^0 = \mathbb{R}^2
\end{array}
\quad \text{and} \quad
\left\{
\begin{array}{l}
X^1 = \{x \in \mathbb{R}^2 \mid -x_1^2 + 1 \le 0\} \\
X^2 = \{x \in \mathbb{R}^2 \mid x_1^2 - 1 < 0\}
\end{array}
\right.
$$

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

$$T^1(x_1, x_2) = \begin{pmatrix} 0.687x_1 + 0.558x_2 - 0.0001x_1x_2 \\ -0.292x_1 + 0.773x_2 \end{pmatrix}$$

and

$$T^2(x_1, x_2) = \begin{pmatrix} 0.369x_1 + 0.532x_2 - 0.0001x_1^2 \\ -1.27x_1 + 0.12x_2 - 0.0001x_1x_2 \end{pmatrix} .$$

The first step consists in constructing the template basis and compute the template p and bound w on the reachable values as a solution of Problem (22). We fix the degree of p to 6. The template p generated from Matlab is of degree 6 and is equal to

$$\begin{array}{l} -1.931348006 + 3.5771x_1^2 + 2.0669x_2^2 + 0.7702x_1x_2 - (2.6284\text{e–}4)x_1^3 \\ -(5.5572\text{e–}4)x_1^2x_2 + (3.1872\text{e–}4)x_1x_2^2 + 0.0010x_2^3 - 2.4650x_1^4 - 0.5073x_1^3x_2 \\ -2.8032x_1^2x_2^2 - 0.5894x_1x_2^3 - 1.4968x_2^4 + (2.7178\text{e–}4)x_1^5 + (1.2726\text{e–}4)x_1^4x_2 \\ -(3.8372\text{e–}4)x_1^3x_2^2 + (6.5349\text{e–}5)x_1^2x_2^3 + (5.7948\text{e–}6)x_1x_2^4 - (6.2558\text{e–}4)x_2^5 \\ +0.5987x_1^6 - 0.0168x_1^5x_2 + 1.1066x_1^4x_2^2 + 0.3172x_1^3x_2^3 + 0.8380x_1^2x_2^4 + 0.0635x_1x_2^5 \\ +0.4719x_2^6 . \end{array}$$

The upper bound w is equal to 2.1343. As suggested in Section 5.3, we can take the template basis $\mathbb{P}_{\text{run}} = \{p, x \mapsto x_1^2, x \mapsto x_2^2\}$. We write q_1 for $x \mapsto x_1^2$ and q_2 for $x \mapsto x_2^2$. The basic semi-algebraic $\{x \in \mathbb{R}^2 \mid p(x) \leq 0,\ q_1(x) \leq 2.1343,\ q_2(x) \leq 2.1343\}$ is an inductive invariant and the corresponding bounds function is $w^0 = (w^0(q_1), w^0(q_2), w^0(p)) = (2.1343, 2.1343, 0)$.

As in Line 4 of Algorithm 4, we compute the image of w^0 by $F^{\mathcal{R}}$ using SOS (Eq. (12)). We found that

$$F^{\mathcal{R}}(w^0)(q_1) = 1.5503,\ F^{\mathcal{R}}(w^0)(q_2) = 1.9501 \text{ and } F^{\mathcal{R}}(w^0)(p) = 0\ .$$

Since $w^0 \in \mathcal{FS}\left(\mathbb{P}_{\text{run}}, \overline{\mathbb{R}}\right)$, Algorithm 4 goes to Line 6 and the computation of $F^{\mathcal{R}}(w^0)$ permits to determine a new policy $\pi(w^0)$. The important data is the vector λ. For example, for $i = 1$ and the template q_1, the vector λ is $(0, 0, 2.0331)$. It means that we associate for each template q a weight $\lambda(q)$. In the case of $\lambda = (0, 0, 2.0331)$, $\lambda(q_1) = 0$, $\lambda(q_2) = 0$ and $\lambda(p) = 2.0332$. For $i = 1$, the template q_1 and the bound vector w^0, the function $\varphi^{\lambda}_{w^0,1,q_1}(v) = 2.0331v(p) + 1.5503$.

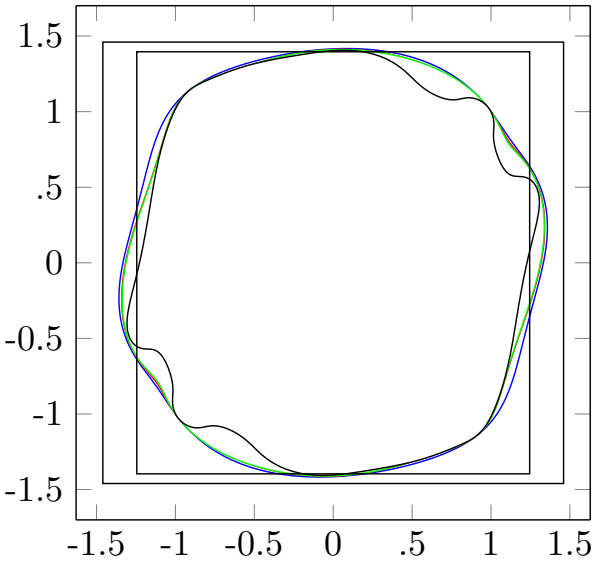
To get the new invariant, Algorithm 4 goes to Line 12 and we compute a bound vector w^1 solution of Linear Program (20). In this case, it corresponds to the following LP problem:

$$\begin{array}{c} \min v(q_1) + v(q_2) + v(p) \\ 2.0331v(p)+1.5503 \leq v(q_1),\ 1.0429v(p)+1.2235 \leq v(q_2),\ 0.9535v(p)-0.0248 \leq v(p), \qquad (i{=}2) \\ 0.4578v(p)+0.8843 \leq v(q_1),\ 0.2048v(p)+1.9501 \leq v(q_2),\ 0.9985v(p)-3.4691\text{e–}7 \leq v(p), \qquad (i{=}1) \\ 1 \leq v(q_1),\ 1 \leq v(q_2),\ 0 \leq v(p), \qquad \text{(init)} \end{array}$$

We obtain:

$$w^1(q_1) = 1.5503,\ w^1(q_2) = 1.9501 \text{ and } w^1(p) = 0\ .$$

Then, we come back to Line 4 of Algorithm 4 and we compute $F^{\mathcal{R}}(w^1)$ using the SOS program Eq. (12). The implemented stopping rule is $\|F^{\mathcal{R}}(w^k) - w^k\|_\infty \leq$ 1e–6 and since $\|F^{\mathcal{R}}(w^1) - w^1\|_\infty \leq$ 1e–6, Algorithm 4 terminates. The computed sets are presented in Figure 3, page 8. Figure 5 presents the semialgebraic sets obtained with higher dimensional templates, up to degree 10. Results are similar but could lead to different numbers of iterations depending on the degree. In case of multiple iterations, the final value is also reached at iterate 1 and is slightly modified by following iterations.



The semialgebraic sets denotes templates of degree 7 (blue), 8 (red), 9 (green) and 10 (black). The first box denotes the initial bounds as obtained in figure 3. The second one is the one obtained after 1 iterations. Except degree 7 that converged in 4 iterations, all others converged in 1. Degree 10 faced numerical issues and did not allow to refine the bounds without errors.

Figure 5: Different templates and associated bounds computed with Policy Iterations

6.2 Benchmarks.

The presented analysis has been applied to available examples of the control community literature: piecewise linear systems, polynomial systems, etc. We gathered the examples matching our criteria: discrete systems, possibly piecewise, at most polynomial. In all the considered cases, no common quadratic Lyapunov existed. In other words, not only the existing linear abstractions such as intervals or polyhedra would fail in computing a non trivial post-fixpoint, but also the existing analyses dedicated to digital filters such as [Fer04, GS07, AGG12, RJGF12].

The analysis has been implemented in Matlab and relies on the Mosek SDP solver [AA00], through the Yalmip [L̈04] SOS front-end. Without outstanding performances, all experiments are performed within a few seconds per iteration, which makes us believe that a more serious implementation would perform better. We recall that the analysis could be interrupted at any point, still providing a safe upper bound.

We next present the examples handled by our SOS policy iteration algorithm:

Example 6.1. *The following example corresponds to [Fen02, Ex. 2.1] and represents a piecewise linear system with 2 cases handling 3 variables. The initial set is:*

$$X^{\text{in}} = [-1, 1]^3 \ .$$

The set where the state-variable lies is:

$$X^0 = \mathbb{R}^3 \ .$$

The sets defining the partition of the state-space are:

$$X^1 = \{(x, y, z) \in \mathbb{R}^3 | x \leq 0\}, \quad X^2 = \{(x, y, z) \in \mathbb{R}^3 | x > 0\} \ .$$

Finally the dynamics associated to the partition are:

$$T^1(x, y, z) = \begin{pmatrix} x + 0.5y \\ -0.3x + 0.8y \\ 0.4z \end{pmatrix}, \quad T^2(x, y, z) = \begin{pmatrix} x + .4y + 0.01z \\ -0.1x + 0.8y \\ 0.5z \end{pmatrix} \ .$$

Example 6.2. *We consider the example [Fen02, Ex. 3.3] which describes a piecewise linear system with 4 cases handling 2 variables. The initial set is:*

$$X^{\text{in}} = [-1, 1]^2 \ .$$

The set where the state-variable lies is:

$$X^0 = \mathbb{R}^2 \ .$$

The sets defining the partition of the state-space are:

$$\begin{aligned} X^1 &= \{(x, y) \in \mathbb{R}^2 | x \leq -1\}, & X^2 &= \{(x, y) \in \mathbb{R}^2 | x \in]-1, 1] \wedge y > 0\}, \\ X^3 &= \{(x, y) \in \mathbb{R}^2 | x \in]-1, 1] \wedge y \leq 0\}, & X^4 &= \{(x, y) \in \mathbb{R}^2 | x > 1\} \ . \end{aligned}$$

Finally the dynamics associated to the partition are:

$$\begin{aligned} T^1(x, y) = \begin{pmatrix} 0.9x - 0.01y \\ 0.1x + y - 0.02 \end{pmatrix}, \quad T^4(x, y) = \begin{pmatrix} 0.9x - 0.01y \\ 0.1x + y + 0.02 \end{pmatrix}, \\ T^2(x, y) = T^3(x, y) = \begin{pmatrix} x - 0.02y \\ 0.02x + 0.9y \end{pmatrix} \ . \end{aligned}$$

Example 6.3. *The following example is the piecewise quadratic system with 2 cases handling 2 variables [AJ13, Ex. 3]. The initial set is:*

$$X^{\text{in}} = [-1, 1]^2 \ .$$

The set where the state-variable lies is:

$$X^0 = \mathbb{R}^2 \ .$$

The sets defining the partition of the state-space are:

$$X^1 = \{(x, y) \in \mathbb{R}^2 | -x^4 + x^2 - 1 \leq 0\}, \quad X^2 = \{(x, y) \in \mathbb{R}^2 | x^4 - x^2 + 1 < 0\} \ .$$

Finally the dynamics associated to the partition are:

$$T^1(x, y) = \begin{pmatrix} 0.687x + 0.558y - 0.0001xy \\ -0.292x + 0.773y \end{pmatrix}, \quad T^2(x, y) = \begin{pmatrix} 0.369x + 0.532y - 0.0001x^2 \\ -1.27x + 0.12y - 0.0001xy \end{pmatrix} \ .$$

Example 6.4. *The following example is the hand-crafted piecewise polynomial of degree 3 with 2 cases developed in [AGM15]. The initial set is:*

$$X^{\text{in}} = [0.9, 1.1] \times [0, 0.2] \ .$$

The set where the state-variable lies is:

$$X^0 = \mathbb{R}^2 \ .$$

The sets defining the partition of the state-space are:

$$X^1 = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1\}, \quad X^2 = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 > 1\} \ .$$

Finally the dynamics associated to the partition are:

$$T^1(x, y) = \begin{pmatrix} x^2 + y^3 \\ x^3 + y^2 \end{pmatrix}, \quad T^2(x, y) = \begin{pmatrix} 0.5x^3 + 0.4y^2 \\ -0.6x^2 + 0.3y^2 \end{pmatrix} \ .$$

The table 1 summarizes the examples considered, the bounds obtained, the degree of the polynomial templates and the number of iterations performed before reaching the fixpoint.

Examples	Bounds (ie. x_i^2)	Degree	# it.
Running example 2.1	No good invariant	4	–
	$[1.5503, 1.9501]$	6	1
	$[1.5503, 1.9502]$	8	7
	$[1.5500, 1.9436]$	10	1
	$[1.5503, 1.9383]$	12	2
Example 6.1	$[3.8260, 2.1632, 1.0000]$	4	1
	$[3.7482, 1.8503, 1.0000]$	6	1
	No good invariant	8,10,12	–
Example 6.2	$[1.8359, 1.3341]$	4	2
	$[1.5854, 1.2574]$	6	5
	$[1.5106, 1.2569]$	8	4
	$[1.4813, 1.2544]$	10	6
Example 6.3	$[1.5624, 1.2396]$	4	3
	$[1.5581, 1.1764]$	6	1
	$[1.5531, 1.1511]$	8	1
	No good invariant	10,12	–
Example 6.4	No good invariant	4,6,8,10	–
	$[1.2100, 0.9989]$	12	max (10)

"No good invariant" occurs when the template synthesis fails, i.e. does not provide a sound post-fixpoint or some numerical issues occurs during the policy iterations phase. It seems to be due to the large size of the SOS problems together with numerical issues related to the interior point methods implemented in the relying solvers.

Table 1: Experiments

7 Conclusion

We proposed an extension of policy iteration algorithms, using Sum-of-Squares programming. This extension allows to consider the wider class of disjunctive polynomial programs. In this new setting, we showed that we keep the advantage of policy iteration algorithms, while producing a sequence of increasingly safe over-approximations of the reachability set.

As future work, we plan to generalize this algorithm to programs involving non-polynomial updates, including square roots, divisions as well as transcendental functions. The computational method developed in the paper could be also generalized to other classes of nonlinear switched systems involving either random or temporal switching.

Acknowledgments

The research leading to these results has partly received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement nr. 306595 "STATOR", the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program "Investissement d'avenir", the RTRA/STAE BRIEFCASE project grant, the ANR projects INS-2012-007 CAFEIN, and ASTRID VORACE.

References

[AA00] Erling D. Andersen and Knud D. Andersen. The mosek interior point optimizer for linear programming: An implementation of the homogeneous algorithm. In *High Performance Optimization*, volume 33 of *Applied Optimization*, pages 197–232. Springer, 2000.

[Adj14] A. Adjé. Policy iteration in finite templates domain. In *7th International Workshop on Numerical Software Verification (NSV'14)*, July 2014.

[AGG10] A. Adjé, S. Gaubert, and E. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In *ESOP*, volume 6012 of *LNCS*, pages 23–42. Springer, 2010.

[AGG12] 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), 2012.

[AGM15] Assalé Adjé, Pierre-Loïc Garoche, and Victor Magron. Property-based polynomial invariant generation using sums-of-squares optimization. In *Static Analysis - 22nd International Symposium, SAS 2015, Saint-Malo, France, September 9-11, 2015, Proceedings*, pages 235–251, 2015.

[AJ13] Amir Ali Ahmadi and Raphael M. Jungers. Switched stability of nonlinear systems via sos-convex lyapunov functions and semidefinite programming. In *CDC 2013*, pages 727–732, 2013.

[Bor99] Brian Borchers. Csdp, a c library for semidefinite programming. *Optimization Methods and Software*, 11(1-4):613–623, 1999.

[CGG+05] A. Costan, S. Gaubert, E. Goubault, M. Martel, and S. Putot. A policy iteration algorithm for computing fixed points in static analysis of programs. In *Computer aided verification*, pages 462–475. Springer, 2005.

[Fen02] Gang Feng. Stability analysis of piecewise discrete-time linear systems. *IEEE Trans. Automat. Contr.*, 47(7):1108–1112, 2002.

[Fer04] Jérôme Feret. Static analysis of digital filters. In *ESOP 2004*, volume 2986 of *LNCS*, pages 33–48. Springer, 2004.

[GGTZ07] Stephane Gaubert, Eric Goubault, Ankur Taly, and Sarah Zennou. Static analysis by policy iteration on relational domains. In *ESOP 2007*, volume 4421 of *LNCS*, pages 237–252. Springer, 2007.

[GS07] Thomas Gawlitza and Helmut Seidl. Precise fixpoint computation through strategy iteration. In *ESOP 2007*, volume 4421 of *LNCS*, pages 300–315. Springer, 2007.

[HK14] Didier Henrion and Milan Korda. Convex computation of the region of attraction of polynomial control systems. *Automatic Control, IEEE Transactions on*, 59(2):297–312, 2014.

[KHJ12] M. Korda, D. Henrion, and C. N. Jones. Inner approximations of the region of attraction for polynomial dynamical systems. *ArXiv e-prints*, October 2012.

[KHJ14] Milan Korda, Didier Henrion, and Colin N. Jones. Convex computation of the maximum controlled invariant set for polynomial control systems. *SIAM Journal on Control and Optimization*, 52(5):2944–2969, 2014.

[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.

[Las09] Jean-Bernard Lasserre. *Moments, positive polynomials and their applications*, volume 1. World Scientific, 2009.

[MVTT13] Anirudha Majumdar, Ram Vasudevan, Mark M Tobenkin, and Russ Tedrake. Convex optimization of nonlinear feedback controllers via occupation measures. *arXiv preprint arXiv:1305.7484*, 2013.

[Par03] Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003.

[PP09] A. Papachristodoulou and S. Prajna. Robust stability analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 54(5):1035–1041, May 2009.

[PTT16] M. Posa, M. Tobenkin, and R. Tedrake. Stability analysis and control of rigid-body systems with impacts and friction. *IEEE Transactions on Automatic Control*, 61(6):1423–1437, June 2016.

[Put93] M. Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

[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.

[SSM04] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Constraint-based linear-relations analysis. In *SAS 2004*, volume 3148 of *LNCS*, pages 53–68. Springer, 2004.

[SVBT14] V. Shia, R. Vasudevan, R. Bajcsy, and R. Tedrake. Convex computation of the reachable set for controlled polynomial hybrid systems. In *53rd IEEE Conference on Decision and Control*, pages 1499–1506, Dec 2014.

[VB94] Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1994.

[WKKM06] Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu. Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006.

[WLW13] T. C. Wang, S. Lall, and M. West. Polynomial Level-Set Method for Polynomial System Reachable Set Estimation. *IEEE Transactions on Automatic Control*, 58(10):2508–2521, Oct 2013.

[Yak77] V. A. Yakubovich. S-procedure in nonlinear control theory. *Vestnik Leningrad University: Mathematics*, 4:73–93, 1977.

[YFN+10] M. Yamashita, K. Fujisawa, K. Nakata, M. Nakata, M. Fukuda, K. Kobayashi, and K. Goto. A high-performance software package for semidefinite programs : Sdpa7. Technical report, Dept. of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 2010.