Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 317 (2015) 3–18

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Policy Iteration in Finite Templates Domain

Assalé Adjé [1,2]

DTIM
Onera Toulouse
31055 Toulouse Cedex 4 - France.

Abstract

We prove in this paper that policy iteration can be generally defined in finite domain of templates using Lagrange duality without any assumption on the templates. Such policy iteration algorithm always converges to a fixed point under a very simple technical condition. This fixed point furnishes a safe over-approximation of the set of reachable values taken by the variables of a program. The paper also discusses the choice of good templates and links these good templates to invariant algebraic relations. When templates are well chosen, the policy iteration algorithm developed in this paper can be easily initialised for one single loop programs.

Keywords: Abstract interpretation, policy iteration, convex optimisation.

1 Introduction

In [1,2], we introduced a complete lattice consisting of sub-level sets of (possibly non-convex) functions, which we use as an abstract domain in the sense of abstract interpretation [5] for computing numerical program invariants. This abstract domain is parameterised by a basis of functions, akin to the approach put forward by Manna, Sankaranarayanan, and Sipma (the linear template abstract domain [17]), except that the basis functions or templates which we used need not be linear. In [1,2], we also developed a policy iteration scheme using Shor relaxation and semi-definite programming in the case of affine arithmetics and quadratic constraints. We only proved that this latter policy iteration converged to a postfixpoint of the relaxed semantics. Moreover, in [1,2], the quadratic templates were provided by the user or more precisely an automatician. Indeed, the set of quadratic templates used in examples of [1,2] contains a Lyapunov function of the affine systems induced by the (affine) arithmetics programs. The usage of Lyapunov functions as quadratic templates was fully automatised by Roux et al in [13].

[1] Email: assale.adje@onera.fr

[2] The author is supported by the RTRA / STAE Project BRIEFCASE.

http://dx.doi.org/10.1016/j.entcs.2015.10.002

Contribution of the paper. In this paper, we present a policy iteration algorithm in finite templates domain using Lagrange duality without any restriction on the arithmetics of the program or the algebraic structure of templates. We also prove that such policy iteration algorithm converges to a fixed point of the relaxed semantics functional.

The paper also deals with the generation of good templates in the simple case of one-loop programs and we link the notion of good templates with Lyapunov functions. Indeed, templates can be thought as invariant algebraic relations which help to prove correctness of the programs. We show that a good choice of templates leads a natural initial policy.

Organisation of the paper. The paper is organised as follows. Section 2 recalls the abstract domain based on non-linear templates and abstract semantics functional. Section 3 details the general construction of the relaxed semantics function using Lagrange duality. Section 4 recalls both Kleene iteration and policy iteration in templates domain and gives convergence proofs. Section 5 proposes a discussion on what the good templates are and how we can initialise policy iteration easily in case on one-loop programs. Section 6 consists in applications. Finally Section 7 concludes.

2 Recalling the generalised templates

In [1,2], we introduced the concept of generalised templates which are just functions from $\mathbb{R}^d$ to $\mathbb{R}$. We can think of hidden algebraic relations to prove certain properties on the analysed program. For the moment, we suppose that these functions are given by some oracles. Suppose that the subset of relations between variables is fixed, we denote by $\mathbb{P}$ this set and $\mathbb{P} \subseteq \mathbf{F}\left(\mathbb{R}^d, \mathbb{R}\right)$. First, we recall the basic definitions (abstraction and concretisation maps) and prove that this pair of maps forms a Galois connection. Then we describe the lattice structures of abstract and concrete domains.

2.1 Basic notions

We are interested in replacing the classical concrete semantics by meaning of sub-level sets i.e. we have a functional representation of numerical invariants through the functions of $\mathbb{P}$. An invariant will be determined as the intersection of sub-level sets. The problem is thus reduced to find the optimal levels on each templates p. We introduce a set of functions from $\mathbb{P}$ to $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$ denoted by $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$. For an element $v \in \mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, we associate the intersection of sub-level sets defined by $v(p)$ where p belongs to $\mathbb{P}$.

Definition 2.1 ($\mathbb{P}$-sub-level sets) *To a function $v \in \mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, we associate the $\mathbb{P}$-sub-level set denoted by $v^\star$ and defined as:*

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

When $\mathbb{P}$ is a set of convex functions, the $\mathbb{P}$-sub-level sets corresponds to the intersection of classical sub-level sets from convex analysis. In our case, $\mathbb{P}$ can contain non-convex functions so $\mathbb{P}$-sub-level sets are not necessarily convex in the usual sense.

We also want a functional representation of a set. In convex analysis, it is well-known that a closed convex set can be represented by its support function i.e. the supremum of linear forms on the set (e.g. [14, § 13]). Here, we use the same notion but we replace the linear forms by the functions $p \in \mathbb{P}$ which are not necessarily linear. This generalisation is not new and was introduced by Moreau [11]. The reader can be also consult [15,16] for more details about those concepts.

Definition 2.2 ($\mathbb{P}$-support functions) *To $X \subseteq \mathbb{R}^d$, we associate the abstract support function denoted by $X^\dagger$ and defined as:*

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

We equip the $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ with the classical partial order for the functions i.e $v \leq w \iff v(p) \leq w(p)$ for all $p \in P$. We order the set of the subsets of $\mathbb{R}^d$ by the inclusion. By taking these orders, we get the following proposition.

Proposition 2.1 *The pair of maps $v \mapsto v^\star$ and $X \mapsto X^\dagger$ defines a Galois connection between $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and the set of subsets of $\mathbb{R}^d$.*

In the terminology of abstract interpretation, $(.)^\dagger$ is the abstraction function, and $(.)^\star$ is the concretisation function. The Galois connection result will provide the correctness of the semantics.

2.2 The lattices of $\mathbb{P}$-convex sets and $\mathbb{P}$-convex functions

Now, we are interested in closed elements (in term of Galois connection) that we call here $\mathbb{P}$-convex elements.

Definition 2.3 ($\mathbb{P}$-convexity) *Let $v \in \mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$, we say that v is a $\mathbb{P}$-convex function if $v = (v^\star)^\dagger$. A set $X \subset \mathbb{R}^d$ is a $\mathbb{P}$-convex set if $X = (X^\dagger)^\star$.*

Definition 2.4 *We respectively denote by $\mathrm{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$ and $\mathrm{Vex}_{\mathbb{P}}(\mathbb{R}^d)$ the set of $\mathbb{P}$-convex function of $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ and the set of $\mathbb{P}$-convex sets of $\mathbb{R}^d$.*

The family of functions $\mathrm{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$ is ordered by the partial order of real-valued functions i.e $v \leq w \iff v(p) \leq w(p)\ \forall p \in \mathbb{P}$. The family of set $\mathrm{Vex}_{\mathbb{P}}(\mathbb{R}^d)$ is ordered by the inclusion order denoted by $\subseteq$. Galois connection permits to construct lattice operations on $\mathbb{P}$-convex elements. Note that as classical convex notion, infima of $\mathbb{P}$-convex functions and joins of $\mathbb{P}$-convex sets are not $\mathbb{P}$-convex and should be convexified. Moreover, as classical convexity, $\mathbb{P}$-convexity is preserved by functional suprema and intersections (meets of $\mathbb{P}$-convex sets).

Definition 2.5 (The meet and join) *Let v and w be in $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$. 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 $\mathrm{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})$ with the join operator $v \vee w =$*

$\sup(v,w)$ *and the meet operator* $v \wedge w = (\inf(v,w)^\star)^\dagger$. *Similarly, we equip* $\mathrm{Vex}_{\mathbb{P}}(\mathbb{R}^d)$ *with the join operator* $X \sqcup Y = ((X \cup Y)^\dagger)^\star$ *and the meet operator* $X \sqcap Y = X \cap Y$.

It is well-known that with the previous lattice operations, the lattice sets of $\mathbb{P}$-convex elements are isomorphic complete lattices. The next theorem follows readily from the fact that the pair of functions $v \mapsto v^\star$ and $C \mapsto C^\dagger$ defines a Galois connection, see e.g. [7, § 7.27].

Theorem 2.2 $(\mathrm{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}}), \wedge, \vee)$ *and* $(\mathrm{Vex}_{\mathbb{P}}(\mathbb{R}^d), \sqcap, \sqcup)$ *are isomorphic complete lattices.*

2.3 Abstract semantics

Suppose now we are given a program with d variables $(x_1, \ldots, x_d)$ and n control points numbered from 1 to n. We suppose this program is written in a simple toy version of a C-like imperative language, comprising global variables, no procedures, assignments of variables using only *parallel assignments* $(x_1, \ldots, x_d) = T(x_1, \ldots, x_d)$, tests of the form $r(x_1, \ldots, x_d) \leq 0$, where $r : \mathbb{R}^d \mapsto \mathbb{R}^m$ (m denotes the number of conjunctions of real tests), and while loops with similar entry tests. We do not recapitulate the standard collecting semantics that associates to this program a monotone map $F : \left(\wp(\mathbb{R}^d)\right)^n \to \left(\wp(\mathbb{R}^d)\right)^n$ whose least fixed points $\mathrm{lfp}(F)$ has as ith component $(i = 1, \ldots, n)$ the subset of $\mathbb{R}^d$ of values that the d variables $x_1, \ldots, x_d$ can take at control point i. The aim of this section is to compute, inductively on the syntax, the abstraction (or a good over-approximation of it) $F^\sharp$ of F from $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$ to itself defined as usual as using Proposition 2.1:

$$\begin{array}{rcll} F^\sharp : \left(\mathrm{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})\right)^n & \to & \left(\mathrm{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})\right)^n & \\ v & \mapsto & (F(v^\star))^\dagger & := p \mapsto \sup\limits_{y \in F(v^\star))} p(y) \end{array}$$

The notation $v^\star$ is in fact the vector of sets $(v_1^\star, \cdots, v_n^\star)$, $(F(v^\star)^\dagger)$ is also interpreted component-wise. We recall that standard collecting semantics F can only take three forms at some control point ℓ. For variable assignments with a map $T : \mathbb{R}^d \to \mathbb{R}^d$ which acts on set of control point ℓ': $F_\ell(X) = T(X_{\ell'})$. We will denote by $\mathbb{A}$ the set of control points representing assignments. For assignments under tests (for both branches of conditional branchments) with a map $T : \mathbb{R}^d \to \mathbb{R}^d$ (acting on set of control point ℓ') and a test map $r_\ell : \mathbb{R}^d \to \mathbb{R}^m$: $F_\ell(X) = T(X_{\ell'} \cap r_\ell^{-1}(\mathbb{R}^m_-))$. We will denote by $\mathbb{I}$ the set of control points representing assignments under tests. For unions (for while loops and join of both branches of condition branchments): $F_\ell(X) = X_{\ell_1} \cup X_{\ell_2}$. We will denote by $\mathbb{U}$ the set of control points representing unions. Finally, the abstract functional $F^\sharp$ takes the following form in case of

assignments under tests and assignments (taking $r_\ell \equiv -1$ for instance):

$$\left(F^\sharp_\ell(v)\right)(p) = \begin{cases} \sup\limits_{y \in T\left(v^\star_{\ell'} \cap r_\ell^{-1}(\mathbb{R}^m_-)\right)} p(y) = \sup\limits_{\substack{x \in v^\star_{\ell'} \\ r_\ell(x) \leqslant 0}} p(T(x)) & \text{if } \ell \in \mathbb{A} \cup \mathbb{I} \\ \sup\limits_{y \in v^\star_{\ell_1} \cup v^\star_{\ell_2}} p(y) = \left(v^\star_{\ell_1} \cup v^\star_{\ell_2}\right)^\dagger(p) & \text{if } \ell \in \mathbb{U} \end{cases} \tag{1}$$

In case of assignments, the abstract functional is the value functional of a constrained optimisation problem and the new least fixed point equation to solve becomes:

$$\inf\{v \in \left(\mathrm{Vex}_{\mathbb{P}}(\mathbb{P} \mapsto \overline{\mathbb{R}})\right)^n \mid F^\sharp(v) \leqslant v\} \ . \tag{2}$$

3 Relaxed semantics using Lagrange duality

In this section, we construct the relaxed semantics from Lagrange duality in general setting without any consideration (except finiteness) on the set of the templates. This construction is an asbtract version of the work of Gaubert et al [8] where the authors only deal with linear templates and affine program arithmetics. Here, the templates and program arithmetics can be non-linear. Nevertheless, non-linearity implies that abstract semantics and relaxed semantics are not equal in general.

First, we quickly recall the concept of Lagrange duality. Then, we construct the relaxed semantics on the assignments, tests and joins. Finally, we present the properties of the relaxed semantics that will be used to prove convergence of policy iteration algorithm to a fixed point of relaxed semantics.

3.1 Lagrange duality

Let f, $\{f_i\}_{i=1,\ldots,k}$ be functions on $\mathbb{R}^d$. Let us consider the following constrained maximisation problem:

$$\sup\{f(x) \mid f_i(x) \leq 0, \forall i = 1, \ldots, k\} \tag{3}$$

In constrained optimisation, it is classical to construct another constrained optimisation problem from the initial one in order to solve an easier problem. Lagrange duality (for details see e.g. [4, § 5.3]) consists adding linearly the constraints with a weight on each (*Lagrange multipliers*) to the objective function. and optimise both weights and problem variable. In our context, the optimal value of Problem (3) is given by the sup-inf (primal) value (4):

$$\sup_{x \in \mathbb{R}^d} \inf_{\lambda \in \mathbb{R}^k_+} f(x) - \sum_{i=1}^{k} \lambda_i f_i(x) \ . \tag{4}$$

The weak duality theorem (see e.g. [4, Prop. 5.6.1]) ensures that if we commute the inf and the sup in Formula (4), the result (*dual value*) is greater than the optimal

value of Problem (4).

$$\inf_{\lambda\in\mathbb{R}_+^k} \sup_{x\in\mathbb{R}^d} f(x) - \sum_{i=1}^{k} \lambda_i f_i(x) \ . \tag{5}$$

Note that the function $\lambda \mapsto \sup_{x\in\mathbb{R}^d} f(x) - \sum_{i=1}^{k} \lambda_i f_i(x)$ is always convex (the image of a segment is smaller than the segment of images) and lower semi-continuous (the sublevel sets are topologically closed), so it has good properties to minimise it.

Finally, the optimal values of Problem (4) and Problem (5) coincide (strong duality theorem see e.g. [4, Prop. 5.3.2]) when the functions $-f$, f_i are convex lower semi-continuous and if the Slater's condition (i.e. there exists $x \in \mathbb{R}^d$ such that $f_i(x) < 0$ for all $i = 1, \ldots, k$) holds. The Slater's condition will be discussed in Subsection 3.4.

3.2 Abstraction of assignments and test using Lagrange duality

Equation (1) becomes an optimisation problem of the form of Equation (3) and we can use Lagrange duality as in the first step of Subsection 3.1. In our case, Lagrange multipliers are some non-negative functions λ from $\mathbb{P}$ to $\mathbb{R}$. We thus consider the function which we will call the *relaxed* function:

$$\left(F_\ell^{\mathcal{R}}(v)\right)(p) = \inf_{\substack{\lambda\in\mathbf{F}(\mathbb{P},\mathbb{R}_+)\\ \mu\in\mathbb{R}_+^m}} \sup_{x\in\mathbb{R}^d} p(T(x)) + \sum_{q\in\mathbb{P}} \lambda(q)\left(v_{\ell'}(q) - q(x)\right) - \mu^\intercal r_\ell(x) \ . \tag{6}$$

3.3 Abstraction of loops

Note that the following double inequalities hold for all $v \in \mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$,

$$\sup(\mathrm{vex}_{\mathbb{P}}(v_{\ell_1}), \mathrm{vex}_{\mathbb{P}}(v_{\ell_2})) \leqslant \left(v_{\ell_1}^\star \cup v_{\ell_2}^\star\right)^\dagger \leqslant \sup(v_{\ell_1}, v_{\ell_2}) \tag{7}$$

This means that as for zones, the union of two such $\mathbb{P}$-convex functions v_{ℓ_1} and v_{ℓ_2} is directly given by taking their maximum on each element of the basis of functions $\mathbb{P}$. Nevertheless, during the fixed point iteration (as in Section 4) the functions v_{ℓ_1} and v_{ℓ_2} are not necessarily $\mathbb{P}$-convex. Moreover, if we take the abstract semantics $F_\ell^\sharp(v)$, we do not have an infimum of linear forms (or at least a maximum of linear forms) on the abstract values v_{ℓ_1} and v_{ℓ_2}, a formulation that we need. Finally, we relaxed the abstract semantics $F_\ell^\sharp(v)$ by the supremum itself and:

$$F_\ell^{\mathcal{R}}(v) = \sup(v_{\ell_1}, v_{\ell_2}) \ . \tag{8}$$

3.4 Properties of the relaxed semantics

The introduction of relaxed semantics aims to get better computational properties of the semantics. We describe in this subsection the properties of the relaxed semantics which justify the using of the new semantics.

First, we recall that relaxed semantics is a safe over-approximation of the abstract semantics (Theorem 3.5 in [2]).

Theorem 3.1 *Let i be in $\mathbb{A} \cup \mathbb{I} \cup \mathbb{U}$. For all $v \in \mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$, $F_i^\sharp(v) \leq F_i^{\mathcal{R}}(v)$.*

Furthermore, we recall that relaxed semantics is monotone (Prop 3.7 in [2]). This property is crucial to show that policy iteration provides more and more precise over-approximation of an invariant until a fixed point is reached.

Proposition 3.2 *For $i \in \mathbb{A} \cup \mathbb{I} \cup \mathbb{U}$, $v \mapsto F_i^{\mathcal{R}}(v)$ is monotone on the set $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$.*

Finally, we recall that the relaxed semantics functional acts as an affine maps on the abstract element (see Lemma 3.10 in [2]). This property is used to prove that Let v be in $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$ and let $i \in \mathbb{A} \cup \mathbb{I}$, we now define, for $p \in \mathbb{P}$, for $(\lambda, \mu) \in \mathbf{F}\left(\mathbb{P}, \mathbb{R}_+\right) \times \mathbb{R}_+^m$, $F_i^{\lambda,\mu}(v)$ by:

$$\left(F_i^{\lambda,\mu}(v)\right)(p) := \sum_{q \in \mathbb{P}} \lambda(q) v_{\ell'}(q) + V_i^{\lambda,\mu}(p) \tag{9}$$

$$\text{where } V_i^{\lambda,\mu}(p) := \sup_{x \in \mathbb{R}^d} p \circ T(x) - \sum_{q \in \mathbb{P}} \lambda(q) q(x) - \mu^\intercal r_\ell(x) \ . \tag{10}$$

The relaxed functional can now be readily rewritten as follows.

Lemma 3.3 *For $i \in \mathbb{A} \cup \mathbb{I}$: $\left(F_j^{\mathcal{R}}(v)\right)(p) = \inf_{\substack{\lambda \in \mathbf{F}(\mathbb{P}, \mathbb{R}_+) \\ \mu \in \mathbb{R}_+^m}} \left(F_j^{\lambda,\mu}(v)\right)(p)$.*

Now, we discuss Slater's condition. First, we give the formal definition of it.

Definition 3.1 (Slater's condition) *Let $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^d \mapsto \mathbb{R}^k$. A constrained maximisation $\sup\{f(x) \mid g(x) \leq 0,\ x \in \mathbb{R}^d\}$ satisfies Slater's condition iff there exists $x_0 \in \mathbb{R}^d$ such that $g(x_0) < 0$ i.e. for all coordinates $i = 1, \ldots, k$, $g_i(x_0) < 0$.*

Slater's condition is linked to the non-emptiness of the interior of the set of constraints. Indeed if the interior of set of constraints is nonempty and constraints function g_i are convex and lower-semicontinuous, then $\text{int}(\{x \in \mathbb{R}^d \mid g(x) \leq 0\}) = \{x \in \mathbb{R}^d \mid g(x) < 0\}$, where int denotes the interior set and $g = (g_1, g_2, \ldots, g_k)$.

Depending on templates we choose, it is easy to check Slater's condition. For example, taking a set of templates $\mathbb{P}$ such that for some $x_0 \in \mathbb{R}^d$, $p(x_0) = 0$ for all $p \in \mathbb{P}$, for $i \in \mathbb{A} \cup \mathbb{I}$, if $v_{\ell(i)}(p) > 0$ and $r_\ell(x_0) < 0$, then Slater's condition holds for maximisation problem (1).

In term of static analysis, fixed point iteration appears when the analysed program contains loops. Slater's condition can fail when the set of possible values taken during the loop iteration is a singleton or templates p are equalities (splitted in p and $-p$) invariant by loop body.

Slater's condition is a sufficient condition to the existence of optimal solutions to the minimisation problem which appears in relaxed functional. Indeed Slater's condition implies the level boundiness of the dual functional. Optimal solutions will be used to compute a "pivoting" policy when a fixed point is not reached.

Proposition 3.4 (Selection property) *Let $i \in \mathbb{A} \cup \mathbb{I}$. Assume that the maximisation problem (1) satisfies the Slater's condition and for all $p \in \mathbb{P}$, there exists $(\lambda_p, \mu_p) \in \mathbf{F}(\mathbb{P}, \mathbb{R}_+) \times \mathbb{R}_+^m$ such that:*

$$\sup_{x \in \mathbb{R}^d} p(T(x)) - \sum_{q \in \mathbb{P}} \lambda_p(q) q(x) - \mu_p^{\intercal} r_\ell(x)$$

is finite. Then the minimisation problem (6) admits a solution i.e. for all $p \in \mathbb{P}$, there exists $(\lambda_p{}^*, \mu_p{}^*) \in \mathbf{F}(\mathbb{P}, \mathbb{R}_+) \times \mathbb{R}_+$ such that:

$$\big(F_i^{\mathcal{R}}(v)\big)(p) = p \circ T(x) + \sum_{q \in \mathbb{P}} \lambda_p{}^*(q)\big(v_{\ell'}(q) - q(x)\big) - \mu_p^{*\intercal} r_\ell(x)$$

The last result of this section discuss about continuity of the relaxed functional. Kleene iteration and policy iteration are iterative processes to compute fixed point. It is important to prove that the limits of the sequences produced by both iterations scheme are fixed points. To show it, we need continuity.

Proposition 3.5 (Continuity result on $F_i^{\mathcal{R}}$) *The following assertions holds:*

(i) *Let $i \in \mathbb{A} \cup \mathbb{I} \cup \mathbb{U}$. Let $p \in \mathbb{P}$. For all decreasing sequences $(v_n)_{n \geq 0} \in \mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$:*

$$\left(F_i^{\mathcal{R}}(\inf_{n \geq 0} v_n)\right)(p) = \left(\inf_{n \geq 0} F_i^{\mathcal{R}}(v_n)\right)(p) \ .$$

(ii) *Let $p \in \mathbb{P}$. Let $i \in \mathbb{A} \cup \mathbb{I}$. Assume, there exists a nonempty compact set $K_{i,p}$ such that $\big(F_i^{\mathcal{R}}(\cdot)\big)(p) = \inf_{(\lambda,\mu) \in K_{i,p}} F_i^{\lambda,\mu}(\cdot)(p)$; Then for all increasing sequences $(v_n)_{n \geq 0} \in \mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$:*

$$\big(\sup_{n \geq 0} F_i^{\mathcal{R}}(v_n)\big)(p) = \big(F_i^{\mathcal{R}}(\sup_{n \geq 0} v_n)\big)(p) \ .$$

4 Solving fixed point equations

4.1 Fixed point equations in templates domain

As usual in abstract interpretation, we are interested in solving the least fixed point Equation (2). Nevertheless, the function $F^\sharp$ is not easily computable (since the templates p are general). Hence, we solve instead the following fixed point equation in $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$:

$$\inf\{v \in \mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n \mid F^{\mathcal{R}}(v) \leq v\} \ . \tag{11}$$

We next describe and compare two ways of computing (or approximating) the smallest fixed point of the relaxed semantics equation: Kleene iteration in Section 4.2, and policy iteration in Section 4.3.

4.2 Kleene iteration

We denote by $\bot$ the smallest element of $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$ i.e. for all $i = 1, \cdots, n$ and for all $p \in \mathbb{P}$, $\bot_i(p) = -\infty$. The Kleene iteration sequence in $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$ is thus as follows:

$$v^0 = \bot, \text{ for } k \geq 0, \ v^{k+1} = F^{\mathcal{R}}(v^k) \ .$$

Now using continuity result of Proposition 3.5, we get the following theorem:

Theorem 4.1 *If for all $i \in \mathbb{A} \cup \mathbb{I}$, for all $p \in \mathbb{P}$, there exists a nonempty compact set $K_{i,p}$ such that $\left(F_i^{\mathcal{R}}(\cdot)\right)(p) = \inf_{(\lambda,\mu)\in K_{i,p}} F_i^{\lambda,\mu}(\cdot)(p)$; then Kleene iteration converges to the smallest fixed point of $F^{\mathcal{R}}$.*

Kleene iteration has the inconvenience that the values v^k which are obtained at a given iteration k (before convergence) do not provide a safe invariant. We shall see that policy iteration does not have this inconvenient: even if it is stopped at an intermediate step, it does provide a safe invariant. Moreover, the convergence of the Kleene iteration can be very slow, so it needs to be coupled with an acceleration technique which provides over-approximations. In [1,2], after a given number of iterations, and during a few iterations, we round bounds outwards with a decreasing precision (akin to the widening used in [9]).

4.3 Policy Iteration

We present now policy iteration algorithm. As usual, we present first the policies notion and then describe completely policy iteration at Algorithm 1.

4.3.1 Policy definition

A policy iteration algorithm can be used to solve a fixed point equation for a monotone function written as an infimum of a family of simpler monotone functions, obtained by selecting *policies*, see [6,8] for more background. The idea is to solve a sequence of fixed point problems involving simpler functions. In the present setting, we look for a representation of the relaxed function

$$F^{\mathcal{R}} = \inf_{\pi\in\Pi} F^{\pi} \tag{12}$$

where the infimum is taken over a set Π whose elements π are called *policies*, and where each function F^{π} is required to be monotone. The correctness of the algorithm relies on a selection property, meaning in the present setting that for each argument (i, v, p) of the function $F^{\mathcal{R}}$, there must exist a policy π such that $\left(F_i^{\mathcal{R}}(v)\right)(p) = \left(F_i^{\pi}(v)\right)(p)$. The idea of the algorithm is to start from a policy π, compute the smallest fixed point v of F^{π}, evaluate $F^{\mathcal{R}}$ at point v, and, if $v \neq F^{\mathcal{R}}(v)$, determine the new policy using the selection property (see Proposition 3.4) at point v.

Let us now identify the policies. Lemma 3.3 shows that for each template p, each coordinate $F_i^{\mathcal{R}}$ corresponding to an assignment $i \in \mathbb{A} \cup \mathbb{I}$ can be written as the infimum of a family of affine functions $v \mapsto F_i^{\lambda,\mu}(v)$, the infimum being taken over

the set of a couple of Lagrange multipliers (λ, μ). Choosing a policy π consists in selecting, for each $i \in \mathbb{A} \cup \mathbb{I}$ and $p \in \mathbb{P}$, a Lagrange multiplier a pair of Lagrange multipliers λ, μ (for $i \in \mathbb{A}$ a Lagrange multiplier has to be chosen, the added test is trivial ans thus μ has to be chosen equal to 0). We denote by $\pi_i(p)$ the value of (λ, μ) chosen by the policy π. Then, the map F^π in (12) is obtained by replacing $F_i^{\mathcal{R}}$ by the affine functions appearing in Lemma 3.3, for $i \in \mathbb{A} \cup \mathbb{I}$. For coordinates corresponding to loops, i.e., $i \in \mathbb{U}$, we take $F_i^\pi = F_i^{\mathcal{R}}$ (the choice of policy is trivial) since the infimum operation does not appear in the expression of $F^{\mathcal{R}}$ (see Equation (8)).

Proposition 3.4 shows that the selection property is valid under a Slater constraint qualification condition. We thus introduce $\mathcal{FS}(P, \overline{\mathbb{R}})^n$, the set of elements of $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)$ which satisfy the Slater condition when the component F_i of F corresponds to an assignment or a test. More concretely: $v \in \mathcal{FS}(P, \overline{\mathbb{R}})^n$, if, for all $i \in \mathbb{A} \cup \mathbb{I}$ the set: $\{x \in \mathbb{R}^d \mid q(x) < v_{\ell'}(q),\ \forall\, q \in \mathbb{P}\} \cap \{x \in \mathbb{R}^d \mid r_\ell(x) < 0\}$ is non-empty.

Note we can do restrictions on policies when degenerate cases appear. At some control point i and for corresponding label j, if there exists $p \in \mathbb{P}$ such that $v_j(p) = -\infty$ then we can choose any vector of non-negative λ such that $\lambda(p) \neq 0$. Note that in this case, $F_i^{\mathcal{R}}(v) \equiv -\infty$ and the smallest fixed point of $F^{\mathcal{R}}$ for the coordinate i must check $v_i \equiv -\infty$. At some control point i and for corresponding label j, if there exists $p \in \mathbb{P}$ such that $v_j(p) = +\infty$ then we can choose any vector of non-negative λ such that $\lambda(p) = 0$ for all $p \in \mathbb{P}$ such that $v_j(p) = +\infty$. These two restrictions let us work with finite values when we have to compute optimal policies.

4.4 Algorithm

Algorithm 1 Policy iteration in finite templates domain

1 Choose $\pi^0 \in \Pi$, $k = 0$.

2 Compute $V^{\pi^k} = \{V^{\pi^k}(q)\}_{q \in P}$ and define the associated function F^{π^k} by choosing λ and μ according to policy π^k using Equation (10).

3 Compute the smallest fixed point v^k in $\mathbf{F}\left(\mathbb{P}, \overline{\mathbb{R}}\right)^n$ of F^{π^k}.

4 If $w^k \in \mathcal{FS}(P, \overline{\mathbb{R}})^n$ continue otherwise return w^k.

5 Evaluate $F^{\mathcal{R}}(w^k)$, if $F^{\mathcal{R}}(w^k) = w^k$ return w^k otherwise take π^{k+1} s.t. $F^{\mathcal{R}}(w^k) = F^{\pi^{k+1}}(w^k)$. Increment k and go to 2.

In [1,2], we have proved that policy iteration on quadratic templates converges to a postfixpoint of our relaxed functional (Theorem 4.2 here). Actually, the result holds independently of the restriction on quadratic templates. Combined with Theorem 3.1, this postfixpoint is also a postfixpoint of abstract semantics.

Theorem 4.2 *The following assertions hold: (1) $F^{\mathcal{R}}(v^l) \neq v^l \implies F^{\mathcal{R}}(v^l) < v^l$; (2) the sequence v^l computed by Algorithm 1 is strictly decreasing; (3) the limit v^∞ of the sequence v^l is a postfixpoint: $F^{\mathcal{R}}(v^\infty) \leq v^\infty$.*

Theorem 4.2 ensures that Algorithm 1 produces a sequence of safe over-approximations

of the numerical invariant we want. Now we complete Theorem 4.2 by showing that actually, Algorithm 1 converges to a fixed point.

Theorem 4.3 (Convergence of Algorithm 1) *If Slater condition is always satisfied then policy iteration converges to a fixed point.*

Proof. Third point of Theorem 4.2 is $F^{\mathcal{R}}(v^\infty) \leq v^\infty$. Now we have to prove that $v^\infty \leq F^{\mathcal{R}}(v^\infty)$. At third step of Algorithm 1, we compute the smallest fixed point of F^{π^k}. Since we have for all $k \geq 0$, $v^{k+1} \leq v^k$ and by the fact that $F^{\pi^{k+1}}$ is order-preserving we have: $v^{k+1} = F^{\pi^{k+1}}(v^{k+1}) \leq F^{\pi^{k+1}}(v^k) = F^{\mathcal{R}}(v^k)$. Now by taking the infimum on k, we get $v^\infty = \inf_k v^{k+1} = \inf_k v^k \leq \inf_k F^{\mathcal{R}}(v^k)$ and finally using the commutation of decreasing inf thanks to Proposition 3.5 then $\inf_k F^{\mathcal{R}}(v^k) = F^{\mathcal{R}}(\inf_k v^k) = F^{\mathcal{R}}(v^\infty)$ and we conclude that $v^\infty \leq F^{\mathcal{R}}(v^\infty)$. □

For the third step of Algorithm 1, since $\mathbb{P}$ is finite and using Lemma 3.3, F^{π^l} is monotone and affine $\mathbf{F}\,(\mathbb{P}, \mathbb{R} \cup \{+\infty\})^n$, we compute the smallest fixed point of F^{π^l} by solving the following linear program see [8, Section 4]:

$$\min \sum_{i=1}^{n} \sum_{q \in \mathbb{P}} v^i(q) \text{ s.t. } \left(F_k^{\pi_k^l}(v)\right)(q) \leq v_k(q), \ \forall k = 1, \cdots, n, \ \forall q \in \mathbb{P} \tag{13}$$

5 Templates design and initial policies

The choice of the initial policies is a crucial point for the quality of the fixed point found by policy iteration. For example, if we know that the values of the variables are bounded an unbounded first invariant can be a fixed point and policy iteration stops. The choice depends on the template design algorithm.

The set of reachable values taken by the variables of the analysed program is bounded (in the sense of a $\mathbb{R}^d$-norm) if there exists a function P such that P is level bounded ($\forall \alpha \in \mathbb{R}$, $\{x \in \mathbb{R}^d \mid P(x) \leq \alpha\}$ is bounded) and a sub-level of P is an invariant (i.e. contains all possible values taken by the variables of the analysed program). Nevertheless, finding both invariant function (relation) and invariant level seems to be difficult and in a first time, in template design step, we only focus on invariant relations. It means that we are looking for a function such that *all* sub-levels are invariant by program updates (assignments and guarded assignments). We can formulate the problem as follows.

Problem 5.1 *Find a function $P : \mathbb{R}^d \to \mathbb{R}$ such that:*

$$\text{For all } \alpha \in \mathbb{R} \text{ there exists } \beta \in \mathbb{R}_+ \text{ such that } P(x) \leq \alpha \implies \|x\|_2^2 \leq \beta; \tag{14a}$$

$$\text{For all } i \in \mathbb{A} \cup \mathbb{I}, \text{ for all } v_i \in \mathbb{R}, \ r_i(x) \leq 0 \wedge P(x) \leq v_i \implies P(T_i(x)) \leq v_i. \tag{14b}$$

We can formulate Problem 5.1 as a constrained maximisation problem and then using Lagrange duality in order to get a more restrictive but easier to solve problem. A solution can be given when affine or polynomial arithmetic are considered. We introduce Problem 5.2 which only deals with inequalities. The formulation in terms of

positivity implies that we could consider relaxations such as sum-of-squares [12,10] to compute polynomial invariants relations. The main issue to this generalisation is the selection of the right degree of the polynomial solution.

Problem 5.2 *Find* $P : \mathbb{R}^d \to \mathbb{R}, \{\gamma_i, i \in \mathbb{I}\}, \gamma_i \in \mathbb{R}^m_+)$ *such that:*

$$\forall\, x \in \mathbb{R}^d,\ P(x) - \|x\|_2^2 \geq 0; \tag{15a}$$

$$\forall\, i \in \mathbb{A} \cup \mathbb{I},\ \forall\, x \in \mathbb{R}^d,\ P(x) - P(T_i(x)) + \gamma_i^\intercal r_i(x) \geq 0; \tag{15b}$$

The next proposition states that a solution of Problem 5.2 gives a solution to Problem 5.1.

Proposition 5.3 (Problem 5.2 solves Problem 5.1) *We have:*

(i) *If* P *satisfies* (15a) *then* P *satisfies* (14a)*;*

(ii) *if* $(P, \{\gamma_i\}_{i\in\mathbb{A}\cup\mathbb{I}})$ *satisfies* (15b) *then* $(P, \{\gamma_i\}_{i\in\mathbb{A}\cup\mathbb{I}})$ *satisfies* (14b)*.*

(iii) *if* $(P, \{\gamma_i\}_{i\in\mathbb{A}\cup\mathbb{I}})$ *is a solution to Problem 5.2 then* $(P, \{\gamma_i\}_{i\in\mathbb{A}\cup\mathbb{I}})$ *is a solution to Problem 5.1.*

Note that Proposition 5.3 and Inequations (15b) can be used to compute unbounded algebraic invariant relations between variables. Then these relations can be used as templates and finite bounds on them (to compute the "diameter" of the numerical invariant) can be found by using policy iteration. If we are interested in proving that set of reachable values are bounded and we must consider the whole set of inequalities included Inequality (15a).

Now, we present the second main result of the paper. Let $(P, \{\gamma_i\}_{i\in\mathbb{A}\cup\mathbb{I}})$ be a solution of Problem 5.2. Using a set of templates $\mathbb{P} = \{P, x_i \mapsto x_i^2\}$. Then policy iteration can be easily initialised (independently of Kleene iteration and widening) for one simple loop program i.e. one loop with one update inside.

Theorem 5.4 (Policy iteration initialisation) *Let us consider the class of programs and the associated relaxed semantics of Figure 1.*

```
x = C;[1]
while [2](r(x) <= 0){
  x = T(x);[3]
}
```

$$\begin{cases} \big(F_1^{\mathcal{R}}(v)\big)(p) = C^\dagger(p) \\ \big(F_2^{\mathcal{R}}(v)\big)(p) = \sup\{v_1, v_3\}(p) \\ \big(F_3^{\mathcal{R}}(v)\big)(p) = \inf\limits_{\substack{\lambda\in\mathbf{F}(\mathbb{P},\mathbb{R}_+)\\ \mu\in\mathbb{R}^m_+}} \sup\limits_{x\in\mathbb{R}^d} p(T(x)) + \sum\limits_{q\in\mathbb{P}} \lambda(q)\,(v_2(q) - q(x)) - \mu^\intercal r(x) \end{cases}$$

Fig. 1. A class of one-loop programs at the top and its associated relaxed semantics functional at the bottom

The C *is the nonempty set where the variables are assumed to be in and* $C^\dagger$

denotes the abstraction of C. The map $T : \mathbb{R}^d \to \mathbb{R}^d$ represents the update of the variables. Let (P, γ) a solution to Problem 5.2 and let $\mathbb{P} = \{x \mapsto x_i^2, i = 1, \ldots, d\} \cup \{P\}$. For all $p' \in \mathbb{P}$, Policy iteration can be initialised with the policy:

$$\pi_3^0(p') = \left(p \mapsto \begin{cases} 0 & \text{if } p \neq P \\ 1 & \text{if } p = P \end{cases}, \gamma \right) \tag{16}$$

Moreover, the following polyhedron: $\{v \in \mathbf{F}(\mathbb{P}, \mathbb{R})^3 \mid F^{\pi^0}(v) \leq v\}$ is nonempty and bounded from below and $\mathrm{lfp}(F^{\pi^0})$ has finite coordinates.

Proof. Consider the initial policy π^0 such that π_3 is given by Equation (16), we have: $F_1^{\pi^0}(v) = C^\dagger$, $F_2^{\pi^0}(v) = \sup\{v_1, v_3\}$, $\left(F_3^{\pi^0}(v)\right)(p) = v_2(P) + V_3^{\pi^0}(p)$ with $V_3^{\pi^0}(p) = \sup_{x \in \mathbb{R}^d} p(T(x)) - P(x) - \gamma^\intercal r_3(x)$. From Inequality (15a), we have for all $p' \in \mathbb{P}$, $p' \neq P$ that $p'(x) \leq \|x\|_2^2 \leq P(x)$ for all $x \in \mathbb{R}^d$ and then $p'(T(x)) \leq \|T(x)\|_2^2 \leq P(T(x))$ for all $x \in \mathbb{R}^d$ also holds. From Inequality (15b), we have $P(T(x)) \leq P(x) + \gamma^\intercal r_3(x)$ for all $x \in \mathbb{R}^d$. Finally, $V_3^{\pi^0}(p') = \sup_{x \in \mathbb{R}^d} p'(T(x)) - P(x) - \gamma^\intercal r_3(x) \leq 0$ for all $p' \in \mathbb{P}$. We conclude that the polyhedron: $K^0 = \{v \in \mathbf{F}(\mathbb{P}, \mathbb{R})^3 \mid F^{\pi^0}(v) \leq v\}$ or more precisely $K^0 = \{v \in \mathbf{F}(\mathbb{P}, \mathbb{R})^3 \mid C^\dagger \leq v_1,\ v_1 \leq v_2,\ v_3 \leq v_2,\ v_2(P) + V_3^{\pi^0}(p') \leq v_3(p'), \forall p' \in \mathbb{P}\}$ is nonempty and bounded from below then the linear program: $\mathrm{Min}\{\sum_{i=1}^3 \sum_{p \in \mathbb{P}} v_i(p) \mid v \in K^0\}$ has a finite solution. □

6 Examples

6.1 With a Lyapunov function

Recall that, for a linear discrete-time dynamical system $x := Ax$, a quadratic function $x \mapsto x^\intercal L x$, where L is a $d \times d$ symmetric matrix, is called Lyapunov function iff: L is positive definite i.e. $x^\intercal L x > 0$ for all nonzero $x \in \mathbb{R}^d$ and $L - A^\intercal L A$ is positive definite. Note that L is positive definite is equivalent up to a multiplicative constant to $L - \mathrm{Id}$ is positive semi-definite ($x^\intercal(L - \mathrm{Id})x \geq 0$ for all $x \in \mathbb{R}^d$).

Suppose there exists only one (convergent) linear update in the analysed program without guards (test is of the form $-1 \leq 0$), then $(x \mapsto x^\intercal L x, 0)$ is a solution of Problem 5.2 for every Lyapunov function $x \mapsto x^\intercal L x$. An algorithm to compute automatically floating points certified Lyapunov functions for while infinite loops and one (guarded) affine update has been developed in [13].

As to illustrate the interest of the approach, let us consider a harmonic oscillator: $\ddot{x} + c\dot{x} + x = 0$. The program of this example which is given at Figure 2 implements an Euler explicit scheme with a small step $h = 0.01$ and $c = 1$, that is, which simulates the linear system $(x, v)^\intercal = T(x, v)^\intercal$ with $T = \begin{pmatrix} 1 & h \\ -h & 1 - h \end{pmatrix}$.

By semi-definite programming, we can compute a Lyapunov function for the lin-

```
x = [0,1];
v = [0,1];
h = 0.01;
while (true) { [2]
  w = v;
  v = v*(1-h)-h*x;
  x = x+h*w; [3] }
```

Fig. 2. Euler integration scheme of a harmonic oscillator

ear system $(x,v) :\mapsto (x,v)L(x,v)^\intercal$ defined as: $L = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$. Recall that Lyapunov functions for linear updates are solution of Problem 5.2. We also use the quadratic functions $\underline{x} : (x,v) \mapsto x^2$ and $\underline{v} : (x,v) \mapsto v^2$ which corresponds to interval constraints. We introduce the set of templates $\mathbb{P} = \{\underline{x}, \underline{v}, \underline{L}\}$. The set of templates $\mathbb{P}$ is thus good set of templates in the sense of Section 5 and we can use Theorem 5.4 to initialise Algorithm 1 and so we choose:

$$\pi_3^0(\underline{x}) = (0,0,1),\ \pi_3^0(\underline{v}) = (0,0,1),\ \pi_3^0(\underline{L}) = (0,0,1)\ .$$

In the case of quadratic templates $\mathbb{P}$, it is easy to evaluate functions V^π. By basic quadratic programming, we find: $V_3^{\pi_3^0}(\underline{x}) = V_3^{\pi_3^0}(\underline{v}) = V_3^{\pi_3^0}(\underline{L}) = 0$. To compute the least fixed point of F^{π^0}, we solve the linear program (see 13) and we obtain the following set at control point 2: $\{x^2 \leq 7,\ v^2 \leq 7,\ 2x^2 + 3v^2 + 2xv \leq 7\}$ which does not provide a fixed point of $F^{\mathcal{R}}$. After 5 iterations, policy iteration stops with a fixed point which provides the following numerical invariant at loop:

$$\{x^2 \leq 3.5000,\ v^2 \leq 2.3333,\ 2x^2 + 3v^2 + 2xv \leq 7\}\ .$$

6.2 Unbounded case

We consider a program which contains a loop while and non trivial test. This program is described at Figure 3.

```
i=0;
j=0;[1]
  while [2](i<=42){
      i=i+1;
      j=j+i;[3]
      }
```

Fig. 3. A simple program with a loop and a test

We want to prove that $j \leq \dfrac{i(i+1)}{2}$. We use policy iteration to prove it. The numerical invariant is unbounded and thus for using Proposition 5.3, we can only

check whether Inequality (15b) holds to initialise our policy iteration. We are looking for non-negative μ such that:

$$-\frac{(i+1)(i+2)}{2} + j + i + \frac{i(i+1)}{2} - j + \mu(42 - i) \leq 0 \; \forall (i,j) \in \mathbb{R}^2$$

A simple calculus permits to show that the inequality holds for $\mu = 0$. So we use the singleton set of templates $\mathbb{P} = \{(i,j) \mapsto -\frac{i(i+1)}{2} + j\}$. Then we can take as initial policy $\pi^0 = (1,0)$ and we get $V_3^{\pi^0} = 0$ and we have to solve the linear program: $\text{Min}\{v_1 + v_2 + v_3 \mid v_1 \geq 0,\ v_2 \geq v_1,\ v_2 \geq v_3,\ v_3 \geq v_2\}$. We get $v_1 = v_2 = v_3 = 0$ which is a fixed point of $F^{\mathcal{R}}$ and provides the wanted numerical invariants.

7 Conclusion and Future Works

We define policy iteration algorithm in a general setting using a finite domain of templates and prove that the algorithm converges to a fixed point of the relaxed semantics. This latter result allows us to use characterisation tools [3] to check whether the solution found is the smallest one. We also define the problem of computing good templates and prove that initialisation of policy iteration is provided from this choice of invariant relations. Future works should include an automatic method to compute the invariant algebraic relations and automatic way to initialise policy iteration from the relations generated.

References

[1] 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.

[2] 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.

[3] A. Adjé, S. Gaubert, and E. Goubault. Computing the smallest fixed point of order-preserving nonexpansive mappings arising in positive stochastic games and static analysis of programs. *Journal of Mathematical Analysis and Applications*, 410(1):227 – 240, 2014.

[4] A. Auslender and M. Teboulle. *Asymptotic Cones and Functions in Optimization and Variational Inequalities*. Springer, 2003.

[5] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. 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.

[6] 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 *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, pages 462–475. Springer, 2005.

[7] B. A. Davey and H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, New York, second edition, 2002.

[8] S. Gaubert, E. Goubault, A. Taly, and S. Zennou. Static analysis by policy iteration on relational domains. In *Proceedings of the Sixteenth European Symposium Of Programming (ESOP'07)*, volume 4421 of *LNCS*, pages 237–252. Springer, 2007.

[9] E. Goubault, S. Putot, P. Baufreton, and J. Gassino. Static analysis of the accuracy in control systems: Principles and experiments. In S. Leue and P. Merino, editors, *Formal Methods for Industrial Critical Systems*, volume 4916 of *Lecture Notes in Computer Science*, pages 3–20. Springer Berlin Heidelberg, 2008.

[10] J.B. Lasserre. *Moments, positive polynomials and their applications*. Imperial College Press optimization series. Imperial College Press, 2010.

[11] J. J. Moreau. Inf-convultion, sous-additivé, convexité des fonctions numériques. *Journal de Mathématiques Pures et Appliquées*, 49:109–154, 1970.

[12] P. Parillo. Semidefinite programming relaxations for semialgebraic problems. *Math. Prog.*, 96(2, series B):293–320, 2003.

[13] 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.

[14] R.T. Rockafellar. *Convex Analysis*. Princeston University Press, 1996.

[15] A. M. Rubinov. *Abstract Convexity and Global optimization*. Kluwer Academic Publishers, 2000.

[16] I. Singer. *Abstract Convex Analysis*. Wiley-Interscience Publication, 1997.

[17] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *The Sixth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *LNCS*, pages 25–41, January 2005. http://www.metapress.com/link.asp?id=X2G04CAME7MDKD72.