Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis

Assalé Adjé[+], Stéphane Gaubert[*] and Eric Goubault[†]

[+] CEA, LIST and LIX, Ecole Polytechnique (MeASI),
F-91128 Palaiseau Cedex, France,
[*] INRIA Saclay and CMAP, Ecole Polytechnique,
F-91128 Palaiseau Cedex, France,
[†] CEA, LIST (MeASI),
F-91191 Gif-sur-Yvette Cedex, France,
Assale.Adje@cea.fr, Stephane.Gaubert@inria.fr, Eric.Goubault@cea.fr

Abstract. We introduce a new domain for finding precise numerical invariants of programs by abstract interpretation. This domain, which consists of level sets of non-linear functions, generalizes the domain of linear "templates" introduced by Manna, Sankaranarayanan, and Sipma. In the case of quadratic templates, we use Shor's semi-definite relaxation to derive computable yet precise abstractions of semantic functionals, and we show that the abstract fixpoint equation can be solved accurately by coupling policy iteration and semi-definite programming. We demonstrate the interest of our approach on a series of examples (filters, integration schemes) including a degenerate one (symplectic scheme).

1 Introduction

We introduce a complete lattice consisting of level sets of (possibly non-convex) functions, which we use as an abstract domain in the sense of abstract interpretation [CC77] for precisely over-approximating numerical program invariants. This abstract domain is parametrized by a basis of functions, akin to the approach set forward by Manna, Sankaranarayanan, and Sipma (the template abstract domain [SSM05,SCSM06]), except that the basis functions or "templates" which we use here need not be linear. The domains obtained in this way encompass the classical abstract domains of intervals, octagons and (linear) templates.

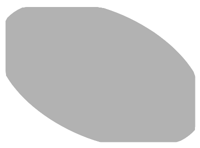
To illustrate the interest of this generalization, let us consider an harmonic oscillator: $\ddot{x} + c\dot{x} + x = 0$. By taking an explicit Euler scheme, and for $c = 1$ we get the program shown at the left of Figure 1.

The invariant found with our method is shown right of Figure 1. For this, we have considered the "template" based on functions $\{x, -x, v, -v, 2x^2 + 3v^2 + 2xv\}$, i.e. we consider a domain where we are looking for upper bounds of

The first author has been supported by a fellowship from the Région Île-de-France. This work has been partially supported by the ANR project ASOPT.

Fig. 1. An harmonic oscillator, its Euler integration scheme and the loop invariant found at control point 2

```
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] }
```



$\{-1.8708 \leq x \leq 1.8708,\ -1.5275 \leq v \leq 1.5275,\ 2x^2+3v^2+2xv \leq 7\}$

these quantities. This means that we consider the linear templates based on $\{x, -x, v, -v\}$, i.e. intervals for each variable of the program, together with the *non-linear* template $2x^2 + 3v^2 + 2xv$. The last template comes from the Lyapunov function that the designer of the *algorithm* may have considered to prove the stability of his scheme, *before it has been implemented.* In view of *proving the implementation correct*, one is naturally led to considering such templates[1]. Last but not least, it is to be noted that the loop invariant using intervals, zones, octagons or even polyhedra (hence with any linear template) is the very disappointing invariant $h = 0.01$ (the variables v and x cannot be bounded.) However, the main interest of the present method is to carry over to the non-linear setting. For instance, we include in our benchmarks a computation of invariants (of the same quality) for an implementation of the Arrow-Hurwicz algorithm, which is essentially an harmonic oscillator limited by a non-linear saturation term (a projection on the positive cone), or a highly degenerate example (a symplectic integration scheme, for which alternative methods fail due to the absence of stability margin).

Contributions of the paper We describe the lattice theoretical operations in terms of Galois connections and generalized convexity in Section 2. We also show that in the case of a basis of quadratic functions, good over-approximations $F^{\mathcal{R}}$ of abstractions $F^\sharp$ of semantic functionals can be computed in polynomial time (Section 3). Such over-approximations are obtained using Shor's relaxation, which is based on semi-definite programming. Moreover, we show in Subsection 4.3 that the multipliers produced by this relaxation are naturally "policies", in a policy iteration technique for finding the fixpoints of $F^{\mathcal{R}}$, precisely over-approximating the fixpoints of $F^\sharp$. Finally, we illustrate on examples (linear recursive filters, numerical integration schemes) that policy iteration on such quadratic templates is extremely efficient and precise in practice, compared with Kleene iteration with widenings/narrowings. The fact that quadratic templates are efficient on such algorithms is generally due to the existence of (quadratic) Lyapunov functions that prove their stability. The method has been implemented as a set of `Matlab` programs.

[1] Of course, as for the templates of [SSM05,SCSM06], we can be interested in automatically finding or refining the set of templates considered to achieve a good precision of the abstract analysis, but this is outside the scope of this article.

Related work This work is to be considered as a generalization of [SSM05], [SCSM06] because it extends the notion of template to non-linear functions, and of [CGG$^+$05], [GGTZ07], [AGG08], [GS07a] and [GS07b] since it also generalizes the use of policy iteration for better and faster resolution of abstract semantic equations. Polynomial inequalities (of bounded degree) were used in [BRCZ05] in the abstract interpretation framework but the method relies on a reduction to linear inequalities (the polyhedra domain), hence is more abstract than our domain. Particular quadratic inequalities (involving two variables - i.e. ellipsoidal methods) were used for order 2 linear recursive filters invariant generation in [Fer05][2]. Polynomial *equalities* (and not general functional inequalities as we consider here) were considered in [MOS04,RCK07]. The use of optimization techniques and relaxation for program analysis has also been proposed in [Cou05], mostly for synthetizing variants for proving termination, but invariant synthesis was also briefly sketched, with different methods than ours (concerning the abstract semantics and the fixpoint algorithm). Finally, the interest of using quadratic invariants and in particular Lyapunov functions for proving control programs correct (mostly in the context of Hoare-like program proofs) has also been advocated very recently by E. Féron et al. in [FF08,FA08].

2 Lattices of level sets and abstract support functions

We introduce a new abstract domain, parametrized by a basis of functions (P below). The idea is that an abstract value will be a vector of bounds for each of these functions, hence the name of "level sets", with some abstract convexity condition, Definition 3.

2.1 $\boldsymbol{P}$-level sets, and their Galois connection with $\boldsymbol{\mathcal{P}}(\mathbb{R}^{d})$

Let P denote a set of functions from $\mathbb{R}^d$ to $\mathbb{R}$, which is going to be the basis of our templates. We denote $\mathcal{F}(P,\overline{\mathbb{R}})$ the set of functions v from P to $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$. We define a Galois connection (Proposition 1) between $\mathcal{F}(P,\overline{\mathbb{R}})$ and the set of subsets of $\mathbb{R}^d$ (made of a concretization operator $v \mapsto v^\star$, Definition 1 and an abstraction operator $C \mapsto C^\dagger$, Definition 2). This will give the formal background for constructing abstract semantics using P-level sets using abstract interpretation [CC77], in Section 3.

Definition 1 (P-level sets). *To a function $v \in \mathcal{F}(P,\overline{\mathbb{R}})$, we associate the P-level set denoted by $v^\star$ and defined as:*

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

When P is a set of convex functions, the P-level sets are the intersection of classical level sets known in convex analysis. In our case, P can contain non-convex functions so P-level sets are not necessarily convex in the usual sense.

[2] A generalization to order n linear recursive filters is also sketched in this article.

Example 1. We come back to the first example and we are focusing on its representation in term of P-level set. Let us write, for $(x, v) \in \mathbb{R}^2$, $p_1 : (x, v) \mapsto x$, $p_2 : (x, v) \mapsto v$ and $p_3 : (x, v) \mapsto 2x^2 + 3v^2 + 2xv$. Let us take $P = \{p_1, -p_1, p_2, -p_2, p_3\}$, $v(p_1) = 1.8708$, $v(-p_1) = 1.8708$, $v(p_2) = 1.5275$, $v(-p_2) = 1.5275$, and $v(p_3) = 7$. The set $v^\star$ is precisely the one shown right of Figure 1.

Example 2. We next show some P-level sets which are not convex in the usual sense. Let us write, for $(x, y) \in \mathbb{R}^2$, $p_1 : (x, y) \mapsto -y^2 - (x + 2)^2$, $p_2 : (x, y) \mapsto -y^2 - (x - 2)^2$ and $p_3 : (x, y) \mapsto -(y - 2)^2 - x^2$, $p_4 : (x, y) \mapsto -(y + 2)^2 - x^2$. Let us take $P = \{p_1, p_2, p_3, p_4\}$ and $v(p_1) = v(p_2) = v(p_3) = v(p_4) = -2$. The set $v^\star$ is shown Figure 2.

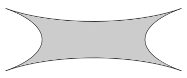


Fig. 2. A P-level set arising from non-convex quadratic functions.

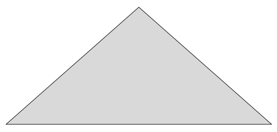

$\{y - x \leq 3,\ y + x \leq 3,\ -y \leq 0\}$

Fig. 3. A P-level set arising from linear forms.

In our case, P is a set of functions from $\mathbb{R}^d$ to $\mathbb{R}$ not necessarily linear, so we generalize the concept of support functions (e.g see Section 13 of [Roc96]).

Definition 2 (Abstract support functions). *To $X \subset \mathbb{R}^d$, we associate the abstract support function denoted by $X^\dagger$ and defined as:*

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

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

In the terminology of abstract interpretation, $(.)^\dagger$ is the abstraction function, and $(.)^\star$ is the concretization function.

2.2 The lattices of *P*-convex sets and *P*-convex functions

The sets of points in $\mathbb{R}^d$ which are exactly represented by their corresponding P-level sets are called P-convex sets, as in the definition below. These can be identified to the set of *abstract elements* we are considering[3]. We show in Theorem 1 that they constitute a complete lattice.

Definition 3 (P-convexity). *Let $v \in \mathcal{F}(P, \overline{\mathbb{R}})$, we say that v is a P-convex function if $v = (v^\star)^\dagger$. A set $X \subset \mathbb{R}^d$ is a P-convex set if $X = (X^\dagger)^\star$.*

[3] Formally, this is the upper-closure in $\mathcal{P}(\mathbb{R}^d)$ of the set of abstract elements.

Example 3. Let us consider a triangle, depicted in Figure 3. If P is the set of linear forms defined by the faces of this triangle i.e P consists of the maps $(x, y) :\mapsto y - x$, $(x, y) :\mapsto y + x$ and $(x, y) :\mapsto -y$, then it is an abstract convex set. But if P is, for example, linear forms defined by orthogonal vectors to the faces of the triangle, the previous triangle is no longer an abstract convex set.

Definition 4. *We respectively denote by* $\mathrm{Vex}_{\mathrm{P}}(P \mapsto \overline{\mathbb{R}})$ *and* $\mathrm{Vex}_{\mathrm{P}}(\mathbb{R}^d)$ *the set of P-convex function of $\mathcal{F}(P, \overline{\mathbb{R}})$ and the set of P-convex sets of $\mathbb{R}^d$.*

Definition 5 (The meet and join). *Let v and w be in $\mathcal{F}(P, \overline{\mathbb{R}})$. We denote by* $\inf(v, w)$ *and* $\sup(v, w)$ *the functions defined respectively by, $p \mapsto \inf(v(p), w(p))$ and $p \mapsto \sup(v(p), w(p))$. We equip* $\mathrm{Vex}_{\mathrm{P}}(P \mapsto \overline{\mathbb{R}})$ *with the meet (respectively join) operator:*

$$v \vee w = \sup(v, w) \tag{1}$$

$$v \wedge w = (\inf(v, w)^\star)^\dagger \tag{2}$$

Similarly, we equip $\mathrm{Vex}_{\mathrm{P}}(\mathbb{R}^d)$ *with the two following operators: $X \sqcup Y = ((X \cup Y)^\dagger)^\star$, $X \sqcap Y = X \cap Y$.*

The family of functions $\mathrm{Vex}_{\mathrm{P}}(P \mapsto \overline{\mathbb{R}})$ is ordered by the partial order of real-valued functions i.e $v \le w \iff v(p) \le w(p)\ \forall p \in P$. The family of set $\mathrm{Vex}_{\mathrm{P}}(\mathbb{R}^d)$ is ordered by the inclusion order denoted by $\subseteq$.

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

Definition 6. *For $v \in \mathcal{F}(P, \overline{\mathbb{R}})$, we denote by* $\mathrm{vex}_{\mathrm{P}}(v)$ *the P-convex hull of v which is the greatest P-convex function smaller than v.*

Similarly, we denote by the set $\mathrm{vex}_{\mathrm{P}}(X)$ *the P-convex hull of a subset X which is the smallest P-convex set greater than X.*

Example 4. Let us come back to the Example 3. Let us take $P = \{(x, y) :\mapsto y + x,\ (x, y) :\mapsto x - y,\ (x, y) :\mapsto -x\}$. Its P-convex hull is the one depicted Figure 4. If we take instead $P = \{(x, y) :\mapsto y^2 - x^2,\ (x, y) :\mapsto x,\ (x, y) :\mapsto -x\}$, its P-convex hull is shown in Figure 5.

Proposition 2 (P-convex hull characterization). *Let v be in $\mathcal{F}(P, \overline{\mathbb{R}})$ and X be a subset of $\mathbb{R}^d$.*

1. *For $p \in P$,* $(\mathrm{vex}_{\mathrm{P}}(v))(p) = \sup\{p(x) \mid x \in \mathbb{R}^d,\ q(x) \le v(q),\ \forall q \in P\}$
2. $\mathrm{vex}_{\mathrm{P}}(X) = \bigcap\{Y \mid Y \in \mathrm{Vex}_{\mathrm{P}}(\mathbb{R}^d),\ X \subseteq Y\}$

2.3 Intervals, Zones, octagons and Manna et al's Templates

The interval domain is naturally subsumed by our abstract convex sets: take as basis $P = \{x_1, -x_1, \ldots, x_n, -x_n\}$ where x_i $(i = 1, \ldots, n)$ are the program variables. And abstract value v in our domain thus naturally corresponds to the supremum of the interval for x_i: $v(x_i)$ and its infimum: $-v(-x_i)$.

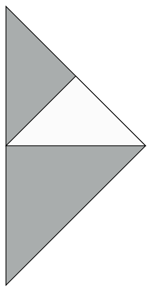


Fig. 4. A convex hull of the triangle.

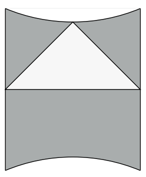


Fig. 5. Another convex hull of the same triangle, with a different set of templates.

Zones and octagons are treated in a similar manner. For instance, for zones, take $P = \{x_i - x_j \mid i, j = 0, \ldots, n, i \neq j\}$, adding a slack variable x_0 (always equal to 0), as customary, to subsume intervals. Of course, (linear) templates as defined in [SSM05] are particular abstract convex sets, for which P is given by a set of linear functionals.

We remark that in the case of zones, $v(x_i - x_j)$ is exactly the entry i, j of the DBM (Difference Bound Matrix) representing the corresponding zone. Also, elements of $\mathrm{Vex}_{\mathrm{P}}(P \mapsto \overline{\mathbb{R}})$ corresponding naturally to *closed* DBMs, that is, canonical forms of DBMs. As well known [Min04b], the union of two zones preserves closure whereas the intersection does not necessarily preserve closure. This is reflected in our domain by (1) and (2).

3 Quadratic Zones

In this section, we instantiate the set P to linear and quadratic functions. This allows us to give a systematic way to derive the abstract semantics of a program. The main result is that the abstract semantics for an assignment for instance, can be approximated precisely in polynomial time by Shor's relaxation scheme, Fact 1.

Definition 7. *We say that P is a quadratic zone if every element (template) $p \in P$ can be written as:*

$$x \mapsto p(x) = x^T A_p x + b_p^T x + c_p,$$

where A_p is a $d \times d$ symmetric matrix (so A can be a zero matrix), x^T denotes the transpose of a vector x, b_p is a $\mathbb{R}^d$ vector and c_p a scalar.

Now, we suppose that P is finite and we suppose that for all q, $v(q) > -\infty$. We denote by $\mathcal{F}(P, \mathbb{R} \cup \{+\infty\})$ the set of functions from P to $\mathbb{R} \cup \{+\infty\}$ and $\mathcal{F}(P, \mathbb{R}_+)$, the set of functions from P to $\mathbb{R}_+$.

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 affine assignments*[4] $(x_1, \ldots, x_d) = T(x_1, \ldots, x_d)$ (i.e. T is an affine map), tests of the form $(x_1, \ldots, x_d) \in C$, where C is some shape in $\mathcal{P}(\mathbb{R}^d)$, and while loops with similar entry tests. We do not recap the standard collecting semantics that associates to this program a monotone map $F : (\mathcal{P}(\mathbb{R}^d))^n \to (\mathcal{P}(\mathbb{R}^d))^n$ whose least fixed points $lfp\ F$ 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 $\mathcal{F}(P, \mathbb{R} \cup \{+\infty\})^n$ to itself defined as usual as:

$$F^\sharp(v) = (F(v^\star)^\dagger) \tag{3}$$

The notation $v^\star$ is in fact the vector of sets $(v_1^\star, \cdots, v_n^\star)$ and $(F(v^\star)^\dagger)$ is also interpreted component-wise. The notation $\mathrm{vex}_{\mathrm{P}}(v)$ will be also understood component-wise.

3.1 Shor's semi-definite relaxation scheme

Shor's relaxation scheme (see Section 4.3.1 of [TN01] or Shor's original article [Sho87] for details) consists of over-approximating the value of a general quadratic optimization problem by the optimal value of a semi-definite programming (SDP) problem. We know, if a dual feasibility condition holds, that the SDP problems are solvable in polynomial time by an interior point method, see e.g [Ali95].

Let p, $\{q_i\}_{i=1,\ldots,m}$ be quadratic functions on $\mathbb{R}^d$. Let us consider the following constrained maximization problem:

$$\sup\{p(x) \mid q_i(x) \leq 0,\ \forall i = 1, \ldots, m\} \tag{4}$$

The first step is to relax the latter optimization problem by Lagrange duality techniques see e.g Section 5.3 of [AT03]. The relaxed problem is:

$$\inf_{\lambda \in \mathbb{R}_+^m} \sup_{x \in \mathbb{R}^d} p(x) + \sum_{i=1}^{m} \lambda_i q_i(x)$$

where $\lambda \in \mathbb{R}_+^m$ are called Lagrange multipliers. Its optimal value is always greater or equal than the optimal value of the problem (4) and is, even, in well-known cases equal (this will be used Proposition 3).

[4] The abstraction of non-linear assignments is outside the scope of this article. Easy ways to deal with them from the material given in this paper include: getting back, locally, to an interval semantics in case of non-linear assignments, or using the linearization methods of [Min04a].

Then, we introduce the matrix $M(p)$, for a quadratic function as in Definition 7 p and the matrix $N(y)$ for a real y defined as:

$$M(p) = \begin{pmatrix} c_p & \frac{1}{2}b_p^T \\ \frac{1}{2}b_p & A_p \end{pmatrix}, \text{ and } N_{1,1}(y) = y,\ N_{i,j}(y) = 0 \text{ if } (i,j) \neq (1,1) \qquad (5)$$

Let $\preceq$ denotes the Loewner ordering of symmetric matrices, so that $A \preceq B$ if all eigenvalues of $B - A$ are non-negative.

Shor's relaxation scheme consists of solving the following SDP problem:

$$\inf_{\substack{\lambda \in \mathbb{R}_+^m \\ \eta \in \mathbb{R}}} \{\eta \text{ s.t } M(p) + \eta N(-1) - \sum_{i=1}^{m} \lambda_i M(q)] \preceq 0\}$$

which is the optimal value of the relaxed problem, hence an over-approximation of the optimal value of the problem (4).

3.2 Abstraction of assignments

In this subsection, we focus on assignments $(x_1, \dots, x_d) = T(x_1, \dots, x_d)$ at control point i. Equation 3 translates in that case to (given that v_{i-1} defines the abstract value at control point $i-1$, i.e. immediately before the assignment):

$$(F_i^\sharp(v))(p) = \sup\{p \circ T(x) \mid q(x) \leq v_{i-1}(q), \forall q \in P\} \qquad (6)$$

We recognize the constrained optimization problem 4 and we use Lagrange duality as in the first step of Subsection 3.1. In our case, the Lagrange multipliers are some non-negative functions λ from P to $\mathbb{R}$. We thus consider the transformed optimization problem:

$$\inf_{\lambda \in \mathcal{F}(P, \mathbb{R}_+)} \sup_{x \in \mathbb{R}^d} p \circ T(x) + \sum_{q \in P} \lambda(q)[v_{i-1}(q) - q(x)] \qquad (7)$$

We write $F_i^{\mathcal{R}}(v)(p)$ for the value of Equation 7. It is called the relaxed function of $F_i^\sharp$. In general, $F_i^{\mathcal{R}}$ is more abstract than $F_i^\sharp$, in other words:

Theorem 2. *For all $v \in \mathcal{F}(P, \mathbb{R} \cup \{+\infty\})^n$, for all $p \in P$,*

$$(F_i^\sharp(v))(p) \leq (F_i^{\mathcal{R}}(v))(p)$$

Moreover, if a constraint qualification, called Slater condition, is satisfied, there exists some λ which achieves the minimum in (7); and the over-approximation we make is not in general that big; in some cases even, the inequality above is an equality:

Proposition 3 (Selection Property). *If the set $\{x \in \mathbb{R}^d \mid q(x) - v_{i-1}(q) < 0,\ \forall\, q \in P\}$ is nonempty, there exists $\lambda^* \in \mathcal{F}(P, \mathbb{R}_+)$ such that:*

$$F_i^{\mathcal{R}}(v)(p) = \sup_{x \in \mathbb{R}^d} p \circ T(x) + \sum_{q \in P} \lambda^*(q)[v_{i-1}(q) - q(x)]$$

Furthermore, if p is a concave quadratic form and if $v_{i-1}(q) < \infty$ only when q is a convex quadratic form, then, we get: $(F_i^{\mathcal{R}}(v))(p) = (F_i^{\sharp}(v))(p)$.

The second part of Proposition 3 follows from the strong duality theorem for convex optimization problems, see e.g. Proposition 5.3.2 of [AT03].

From now on, we write, for a map w from P to $\mathbb{R} \cup \{+\infty\}$,

$$w^{\circ} = \{x \in \mathbb{R}^d \mid q(x) - w(q) < 0, \, \forall q \in P\}.$$

Let us suppose that $v_{i-1}^{\circ} \neq \emptyset$, let us fix $\lambda \in \mathcal{F}(P, \mathbb{R}_+)$, and observe that the sum $\sum_{q \in P} \lambda(q) v_{i-1}(q)$ does not depend on the variable x in (7). We now define $F_i^{\lambda}(v)$ by :

$$(F_i^{\lambda}(v))(p) = \sum_{q \in P} \lambda(q) v_{i-1}(q) + V_i^{\lambda}(p) \tag{8}$$

$$\text{where } V_i^{\lambda}(p) = \sup_{x \in \mathbb{R}^d} p \circ T(x) - \sum_{q \in P} \lambda(q) q(x) \tag{9}$$

So that, $(F_i^{\mathcal{R}}(v))(p) = \inf_{\lambda \in \mathcal{F}(P, \mathbb{R}_+)} (F^{\lambda}(v))(p)$.

By applying the so-called "Simple Lemma" of Section 4.3.1 of [TN01], we can write (9) as the following SDP problem:

$$V_i^{\lambda}(p) = \inf\{\eta \in \mathbb{R} \mid M(p \circ T) + \eta N(-1) - \sum_{q \in P} \lambda(q) M(q) \preceq 0\}$$

where $M(p \circ T)$, $N(-1)$ and $M(q)$ are the matrices defined in (5).

So, by applying Shor's relaxation scheme of Subsection 3.1, we get:

$$(F_i^{\mathcal{R}}(v))(p) = \inf_{\substack{\lambda \in \mathcal{F}(P, \mathbb{R}_+) \\ \eta \in \mathbb{R}}} \eta \text{ s.t } M(p \circ T) + \eta N(-1) + \sum_{q \in P} \lambda(q)[N(v_{i-1}(q)) - M(q)] \preceq 0 \tag{10}$$

which can be solved by a SDP solver.

To get a safe optimal value of (10), we can use a verified SDP solver as VSDP [JCK07].

We remark that we can apply the Shor's relaxation scheme for over-approximating the P-convex hull of a given function $w \in \mathcal{F}(P, \overline{\mathbb{R}})$, which will be useful in the next section.

Corollary 1. *Let w be in $\mathcal{F}(P, \overline{\mathbb{R}})$ and p in P we have:*

$$(\mathrm{vex_P}(w))(p) \leq \inf_{\substack{\lambda \in \mathcal{F}(P, \mathbb{R}_+) \\ \eta \in \mathbb{R}}} \eta \;\; s.t \;\; M(p) + \eta N(-1) + \sum_{q \in P} \lambda(q)[N(w(q)) - M(q)] \preceq 0$$

Finally, we conclude that we can compute over-approximations of (6) as well as over-approximations of the P-convex hull of an element of $\mathcal{F}(P, \overline{\mathbb{R}})$ by solving a SDP problem, which can be done in polynomial time [TN01]. We sum up what we achieved in the following fact:

Fact 1. *In the case of quadratic templates, the relaxed functional $F^{\mathcal{R}}$ and a sound over-approximation of the P-convex hull operation can be evaluated using Shor's semi-definite relaxation.*

Example 5. We analyze the following parallel affine assignment T that implements a rotation of angle ϕ on the unit sphere S^1 of $\mathbb{R}^2$:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

where $x^2 + y^2 = 1$.

Let us take $P = \{p_1(x,y) \mapsto x^2 + y^2,\ p_2(x,y) \mapsto -(x^2+y^2)\}$ and we set $v_1(p_1) = 1$ and $v_1(p_2) = -1$. Equation (10) translates into:

$$\begin{aligned} v_2(p_1) &= T(v_1^\star)^\dagger(p_1) = \sup\{p_1(T(x,y)) \mid p_1(x,y) \leq 1,\ p_2(x,y) \leq -1\} \\ v_2(p_2) &= T(v_1^\star)^\dagger(p_2) = \sup\{p_2(T(x,y)) \mid p_1(x,y) \leq 1,\ p_2(x,y) \leq -1\} \end{aligned}$$

$v_2(p_1) =$

$$\inf_{\substack{\lambda(p_1)\geq 0 \\ \lambda(p_2)\geq 0 \\ \eta\in\mathbb{R}}} \eta \text{ s.t} \begin{pmatrix} -\eta + \lambda(p_1) - \lambda(p_2) & 0 & 0 \\ 0 & 1 - \lambda(p_1) + \lambda(p_2) & 0 \\ 0 & 0 & 1 - \lambda(p_1) + \lambda(p_2) \end{pmatrix} \preceq 0$$

and
$v_2(p_2) =$

$$\inf_{\substack{\lambda(p_1)\geq 0 \\ \lambda(p_2)\geq 0 \\ \eta\in\mathbb{R}}} \eta \text{ s.t} \begin{pmatrix} -\eta + \lambda(p_1) - \lambda(p_2) & 0 & 0 \\ 0 & -1 - \lambda(p_1) + \lambda(p_2) & 0 \\ 0 & 0 & -1 - \lambda(p_1) + \lambda(p_2) \end{pmatrix} \preceq 0$$

To solve these optimization problems, we could call an SDP solver, but in this case, it suffices to solve a system of inequalities: all the diagonal elements must be non-positive, for example, for the first problem, this implies that $\eta \geq 1$ and since we minimize η we get $\eta = 1$.

Hence, we find $v_2(p_1) = 1$ and $v_2(p_2) = -1$. This simple analysis finds automatically that the circle is invariant by a rotation.

3.3 Abstraction of simple tests

We assume here that a test $(x_1, \ldots, x_d) \in C$ is translated on three control points $j-2$, $j-1$, j and $j+1$ as follows: $F_{j-1}(X) = C$, $F_j(X) = X_{j-2} \cap X_{j-1}$, for the "`then`" branch. For the "`else`" branch, beginning at control point k, we have similarly $F_k(X) = X_{j-2} \cap \neg X_{j-1}$. As we deal with arbitrary $C \in \mathcal{P}(\mathbb{R}^d)$, it is sufficient to show here how to deal with the equations on control points $j-2$, $j-1$ and j.

By using Equation (3), we get, for $v \in \mathcal{F}(P, \mathbb{R} \cup \{+\infty\})^n$, and $p \in P$, $(F_j^\sharp(v))(p) = ((v_{j-2}^\star \sqcap v_{j-1}^\star)^\dagger(p)$ then, by a simple calculus,

$$(F_j^\sharp(v))(p) = (v_{j-2} \wedge v_{j-1})(p).$$

As for the abstraction of assignments, we calculate $F_j^{\mathcal{R}}$ instead of $F_j^\sharp$. We can compute $F_j^{\mathcal{R}}$ in two ways. The first one consists in using the fact that $(v_{j-2} \wedge v_{j-1})(p) = \mathrm{vex}_{\mathrm{P}}(\inf(v_{j-2}, v_{j-1}))(p)$. Hence we can apply Proposition 1 to $\inf(v_{j-2}$-$, v_{j-1})$ as a practical means to compute $F_j^{\mathcal{R}}$, using a SDP solver. This method can be used during Kleene iteration, since at any iteration, we know the values taken by v_{j-2} and v_{j-1}. Unfortunately, this method cannot be used in policy iteration, hence we use the following method in that case.

The second method consists in noticing that $x \in v_{j-2}^\star \sqcap v_{j-1}^\star \Rightarrow \forall q \in P,\ q(x) \leq v_{j-2}(q)$ and $q(x) \leq v_{j-1}(q)$ so:

$$(F_j^\sharp(v))(p) = \sup\{p(x) \mid q(x) \leq v_{j-2}(q),\ q(x) \leq v_{j-1}(q)\ \forall q \in P\}$$

Then, supposing the Slater condition is satisfied, it suffices to apply the same techniques as for the abstraction of assignments. The only difference is that we have now a couple (λ, μ) of $\mathcal{F}(P, \mathbb{R}_+)$ as Lagrange multipliers, the first one is associated to v_{j-1} and the second one to v_{j-2}. The function (9) becomes a function which depends on the two parameters (λ, μ), this new function is written $V_j^{(\lambda,\mu)}$, its evaluation is reduced once again to a SDP problem.

Thus, as for (8), we have the following affine form $F_j^{(\lambda,\mu)}$ on $\mathcal{F}(P, \mathbb{R} \cup \{+\infty\})$:

$$(F_j^{(\lambda,\mu)}(v))(p) = \sum_{q \in P} \lambda(q) v_{j-1}(q) + \sum_{q \in P} \mu(q) v_{j-2}(q) + V_j^{(\lambda,\mu)}(p) \qquad (11)$$

The latter affine form is used for computing by linear programming the smallest fixpoint of a map associated to a policy (that, we will see in Section 4.3 corresponds to the Lagrange multipliers (λ, μ)).

Then, the relaxed function of $F_j^\sharp$ is evaluated by solving the same kind of SDP problem as in Equation (10).

3.4 Abstraction of loops

The only thing that we do not know yet how to interpret in the collecting semantics equations is the equation at control point i where we collect the values of the variables before the entry in the body of the loop, at control point $i-1$, with the values of the variables at the end of the body of the loop, at control point j: $F_i(X) = X_{i-1} \cup X_j$, since we know now how to deal with the interpretation of tests.

By using Equation (3), for $v \in \mathcal{F}(P, \mathbb{R} \cup \{+\infty\})^n$ and $p \in P$, $(F_i^\sharp(v))(p) = (v_{i-1}^\star \sqcup v_j^\star)^\dagger(p)$, by a simple calculus, the latter equality becomes:

$$(F_i^\sharp(v))(p) = \mathrm{vex}_{\mathrm{P}}(\sup(v_{i-1}, v_j))(p).$$

Hence, the calculus of the union can be reduced to a P-convex hull computation, see Proposition 1.

During a fixpoint iteration (as in Section 4), we only have to deal with "closed" abstract values, that is, elements v in $\mathrm{Vex}_\mathrm{P}(P \mapsto \overline{\mathbb{R}})^n$. As for zones, we notice that the union of two such "closed" abstract values v_{i-1} and v_j is directly given by taking their maximum on each element of the basis of quadratic functions P, without having to take its closure.

4 Solving the semantic equation

4.1 Fixpoint equations in quadratic zones

We recall that P is a finite set of quadratic templates and F is a monotone map which interprets a program with d variables and n labels in $(\mathcal{P}(\mathbb{R}^d))^n$. We want to find the smallest vector in $(\mathcal{P}(\mathbb{R}^d))^n$ such that $F(X) = X$. This fixpoint equation is generally unsolvable algorithmically. So as customary in abstract interpretation, we solve instead the abstract equation:

$$\inf\{v \in \mathrm{Vex}_\mathrm{P}(P \mapsto \overline{\mathbb{R}})^n \mid v = F^{\mathcal{R}}(v)\} \tag{12}$$

where v belongs to $\mathrm{Vex}_\mathrm{P}(P \mapsto \overline{\mathbb{R}})^n$.

We recall that $v^\star$ denotes the vector of sets $((v_1)^\star, \cdots, (v_n)^\star)$ and $F^\sharp(v) = (F(v^\star))^\dagger$ i.e $\forall\, i$, $F_i^\sharp(v) = (F_i(v^\star))^\dagger$ and $F^{\mathcal{R}}$ is the map, the components of which are the relaxed functions of $F^\sharp$.

We define and compare two ways of solving the fixpoint equation: Kleene iteration in Section 4.2, and policy iteration in Section 4.3.

4.2 Kleene iteration

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

1. $v^0 = \bot$
2. for $k \geq 0$, $v^{k+1} = \mathrm{vex}_\mathrm{P}(\sup(v^k, F^{\mathcal{R}}(v^k)))$

This sequence converges to the smallest fixpoint of $\mathrm{vex}_\mathrm{P}(F^{\mathcal{R}})$. But, the computation of it can be very slow or can never end so we use an acceleration technique to over-approximate it rapidly. After a certain number of iterations and during some iterations, we round bounds outwards with a decreasing precision (akin to the widening used in [GPBG08]). The closure we use, after each widening step during Kleene iteration, might end up not being a widening (as is the case in zones). So we extrapolate the result to $\top$ ($\top_i(p) = \infty$ for all $i = 1, \cdots, n$ and all $p \in P$) after a fixed number of steps.

4.3 Policy iteration algorithm

Selection property and policy iteration algorithm To define a policy iteration algorithm, we need policies. Here, our policies are given by the Lagrange multipliers introduced by the relaxation in the interpretation of assignments, Section 3.2, and in the interpretation of tests, Section 3.3. Hence the set of policies is the union of the sets of Lagrange multipliers for each assignment of the program and couple of Lagrange multipliers for each test of the program.

To define a policy iteration algorithm, we also need a selection property, as in e.g [GGTZ07]. We saw in Proposition 3 that the selection property is given by a constraint qualification argument. We thus introduce $\mathcal{FS}(P,\overline{\mathbb{R}})^n$, the set of elements of $\mathcal{F}(P,\overline{\mathbb{R}})$ which satisfy the Slater condition:

– when the component F_i of F corresponds to an assignment, the set of policies at i is the union of the sets of Lagrange multipliers,
– when the component F_j of F corresponds to a test, the set of policies at j is the union of the sets of couple of Lagrange multipliers.

We saw that for other coordinates, the set of policy is a singleton. We denote by $\mathit{\Pi}$ the set of all policies π and by π_i a policy at i.

Algorithm 1 Policy Iteration in Quadratic Templates

```
1 Choose $\pi^0 \in \mathit{\Pi}$ such that $V^{\pi^0} < +\infty$, $k = 0$.
2 Compute $V^{\pi_i^k} = \{V^{\pi_i^k}(q)\}_{q\in P}$.
3 Compute the smallest fixpoint $v^k$ in $\mathcal{F}(P,\overline{\mathbb{R}})^n$ of $F^{\pi^k}$.
4 Compute $w^k = \mathrm{vex}_{\mathrm{P}}(v^k)$.
5 If $w^k \in \mathcal{FS}(P,\overline{\mathbb{R}})^n$ continue otherwise return $w^k$.
6 Evaluate $F^{\mathcal{R}}(w^k)$, if $F^{\mathcal{R}}(w^k) = w^k$ return $w^k$ otherwise take $\pi^{k+1}$ s.t $F^{\mathcal{R}}(w^k) =$
  $F^{\pi^{k+1}}(w^k)$ and go to 2.
```

Remark 1. The initial policy is given after few Kleene iterations: this gives us a vector $v \in \mathrm{Vex}_{\mathrm{P}}(P \mapsto \overline{\mathbb{R}})^n$, then we compute, by solving Equation (10) and its equivalent for the abstraction of tests, a policy π^0.

For the third step of Algorithm 1, since P is finite and using (8) and (11), F^{π^l} is monotone and affine $\mathcal{F}(P,\overline{\mathbb{R}})^n$, we compute the smallest fixpoint of F^{π^l} by solving the following linear program see Section 4 of [GGTZ07]:

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

Remark 2. To ensure the feasibility of the solution of (13) computed by the LP solver, we replace, when possible, the constraint set by $F^{\pi^l}(v) + \epsilon \leq v$, where ϵ is a small constant (typically of the order of several $ulp(v)$).

To obtain safe bounds even though we run our algorithm on machine which uses finite-precision arithmetic, we should use a guaranteed LP solver (e.g LURUPA see [Kei05]) to check that the solution obtained verifies $F^{\pi^l}(v) \leq v$.

We can only prove that policy iteration on quadratic templates converges (maybe in infinite time) towards a post-fixed point of our abstract functional and that under some technical conditions, it converges towards a fixed point. One interest in policy iteration for static analysis is that we can always terminate the iteration after a finite time, and ends up with a post-fixed point.

Theorem 3. *The sequence v^l computed by Algorithm 1 is non-increasing.*

Remark 3. In the case of intervals, zones and templates, at least for programs containing only linear or concave quadratic expressions in assignments, Proposition 3 implies that $F^\sharp = F^{\mathcal{R}}$. Therefore, we are giving a policy iteration algorithm in this paper, computing the same least fixpoints as the policy iteration algorithms described in papers [CGG+05,AGG08,GGTZ07].

4.4 A detailed calculation on the running example

Now we give details on the harmonic oscillator of Example 1. The program of this example implements an Euler explicit scheme with a small step h, that is, which simulates the linear system $(x, v)^T \leftarrow T(x, v)^T$ with $T = \begin{pmatrix} 1 & h \\ -h & 1-h \end{pmatrix}$.

The function $(x, v) :\mapsto (x, v)L(x, v)^T$ is a Lyapunov function of the new linear system with $L = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$.

We write $\underline{x} : (x, v) \mapsto x$, $\underline{v} : (x, v) \mapsto v$, $\underline{L} : (x, v) \mapsto (x, v)L(x, v)^T$ and finally $P = \{\underline{x}, -\underline{x}, \underline{v}, -\underline{v}, \underline{L}\}$. For $p = \underline{x},\ -\underline{x},\ \underline{v},\ -\underline{v},\ \underline{L}$ and $w \in \mathcal{F}(P, \overline{\mathbb{R}})$, we get the semantic equations described below the corresponding C code, at Figure 6, for all three control points.

Fig. 6. Implementation of the harmonic oscillator and its semantics in $\mathcal{F}(P, \overline{\mathbb{R}})^3$

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

$$F_1^\sharp(w)(p) = \{\underline{x}(x, v) \le 1,\ -\underline{x}(x, v) \le 0, \underline{v}(x, v) \le 1,\ -\underline{v}(x, v) \le 0, \underline{L}(x, v) \le 7\}$$
$$F_2^\sharp(w)(p) = \sup\{w_1(p), w_3(p)\}$$
$$F_3^\sharp(w)(p) = \sup_{(x,v)\in(w_2)^\star} (p \circ T)(x, v)$$

Now we are going to focus on the third coordinate of $(F^{\mathcal{R}}(v))(p)$. Let us consider, for example, $p = \underline{x}$, we get: $(F_3^{\mathcal{R}}(v))(\underline{x}) =$

$$\inf_{\lambda\in\mathcal{F}(P,\mathbb{R}_+)} \sum_{q\in P} \lambda(q)v_2(q) + \sup_{(x,v)} (x, v)(\lambda(\underline{L})(x, v)^T + \begin{pmatrix} 1 + \lambda(-\underline{x}) - \lambda(\underline{x}) \\ h + \lambda(-\underline{v}) - \lambda(\underline{v}) \end{pmatrix} (x, v)^T + 0. \tag{14}$$

By introducing the following symmetric matrices, we can rewrite (14) as (8): $M(\underline{x})(1,2) = M(\underline{x})(2,1) = \frac{1}{2}$ and 0 otherwise. $M(\underline{v})(1,3) = M(-\underline{v})(3,1) = \frac{1}{2}$ and 0 otherwise. $M(\underline{L})(2,2) = 2,\ M(\underline{L})(3,3) = 3,\ M(\underline{L})(2,3) = M(\underline{L})(3,2) = 1$ and 0 otherwise. Furthermore, $M(-\underline{x}) = M(\underline{x})$ and $M(-\underline{v}) = M(\underline{v})$.

$M(\underline{x} \circ T) = \begin{pmatrix} 0 & \frac{1}{2} & \frac{h}{2} \\ \frac{1}{2} & 0 & 0 \\ \frac{h}{2} & 0 & 0 \end{pmatrix}$

To initialize Algorithm 1, we choose a policy π^0. For the third coordinate of $F^{\mathcal{R}}$, we have to choose a policy π_3^0 such that $V_3^{\pi_3^0}(p)$ is finite. We can start, for example, by $\pi_3^0(p) = (0,0,0,0,1)$ for all $p \in P$. This consists, for $p = \underline{x}$, in taking $\lambda(\underline{x}) = \lambda(-\underline{x}) = \lambda(\underline{v}) = \lambda(-\underline{v}) = 0$ and $\lambda(\underline{L}) = 1$ in (14). By a `Matlab`[5] implementation, using `Yalmip` [L04] and `SeDuMi` [Stu99], we find:

$V_3^{\pi_3^0}(\underline{x} \circ F) = V_3^{\pi_3^0}(-\underline{x}) = 0.149$, $V_3^{\pi_3^0}(\underline{v}) = V_3^{\pi_3^0}(-\underline{v}) = 0.099$ and $V_3^{\pi_3^0}(\underline{L}) = 0$.

We solve the following linear program (see (13)):

$$\min \sum_{i=1}^{3} \sum_{p \in P} \beta_i(p)$$
$$\beta_2(\underline{L}) + V^{\pi_3^0}(p \circ F) \leq \beta_3(p)\ \ \forall p$$
$$\beta_3(p) \leq \beta_2(p),\ \forall p$$
$$1 \leq \beta_2(\underline{x}),\ 0 \leq \beta_2(-\underline{x}), 1 \leq \beta_2(\underline{v}),\ 0 \leq \beta_2(-\underline{v}),\ 7 \leq \beta_2(\underline{L})$$
$$1 \leq \beta_1(\underline{x}),\ 0 \leq \beta_1(-\underline{x}), 1 \leq \beta_1(\underline{v}),\ 0 \leq \beta_1(-\underline{v}),\ 7 \leq \beta_1(\underline{L})$$

Using solver `Linprog`, we find:

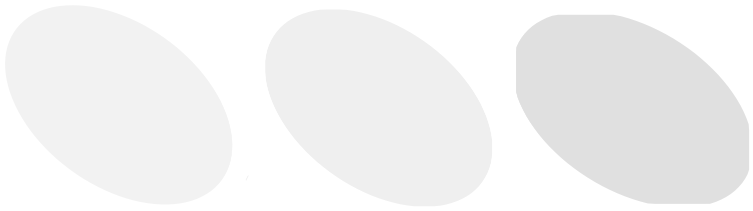
$$\begin{array}{lll}
w_1^0(\underline{x}) = \ \ 1.0000 & w_2^0(\underline{x}) = \ \ 7.1490 & w_3^0(\underline{x}) = \ \ 7.1490 \\
w_1^0(-\underline{x}) = 0 & w_2^0(-\underline{x}) = 7.1490 & w_3^0(-\underline{x}) = 7.1490 \\
w_1^0(\underline{v}) = \ \ 1.0000 & w_2^0(\underline{v}) = \ \ 7.0990 & w_3^0(\underline{v}) = \ \ 7.0990 \\
w_1^0(-\underline{v}) = 0 & w_2^0(-\underline{v}) = 7.0990 & w_3^0(-\underline{v}) = 7.0990 \\
w_1^0(\underline{L}) = \ \ 7.0000 & w_2^0(\underline{L}) = \ \ 7.0000 & w_3^0(\underline{L}) = \ \ 7.0000
\end{array}$$

The calculus of $u = \mathrm{vex}_{\mathrm{P}}(w^1)$ returns:

$$\begin{array}{lll}
u_1^0(\underline{x}) = \ \ 1.0000 & u_2^0(\underline{x}) = \ \ 2.0493 & u_3^0(\underline{x}) = \ \ 2.0493 \\
u_1^0(-\underline{x}) = 0 & u_2^0(-\underline{x}) = 2.0493 & u_3^0(-\underline{x}) = 2.0493 \\
u_1^0(\underline{v}) = \ \ 1.0000 & u_2^0(\underline{v}) = \ \ 1.6733 & u_3^0(\underline{v}) = \ \ 1.6733 \\
u_1^0(-\underline{v}) = 0 & u_2^0(-\underline{v}) = 1.6733 & u_3^0(-\underline{v}) = 1.6733 \\
u_1^0(\underline{L}) = \ \ 7.0000 & u_2^0(\underline{L}) = \ \ 7.0000 & u_3^0(\underline{L}) = \ \ 7.0000
\end{array}$$

Using again `Yalmip` with the solver `SeDuMi`, the vector u is not a fixpoint of $F^{\mathcal{R}}$, so we get the new following new policy: $\pi_3^1(\underline{x}) = (0.9035, 0, 0, 0, 0.0134)$, $\pi_3^1(-\underline{x}) = (0, 0.9035, 0, 0, 0.0134)$, $\pi_3^1(\underline{v}) = (0, 0, 0.8830, 0, 0.0135)$, $\pi_3^1(-\underline{v}) = (0, 0, 0, 0.8830, 0.0135)$, $\pi_3^1(\underline{L}) = (0, 0, 0, 0, 0.9946)$. The invariant of the loop i.e. $w_2^\star$ at control point 2 is $\{-1.8708 \leq x \leq 1.8708,\ -1.5275 \leq v \leq 1.5275,\ 2x^2 + 3v^2 + 2xv \leq 7\}$ and is computed in 14 seconds. We draw $w_2^\star$ at each iteration of Algorithm 1 in Figure 7.

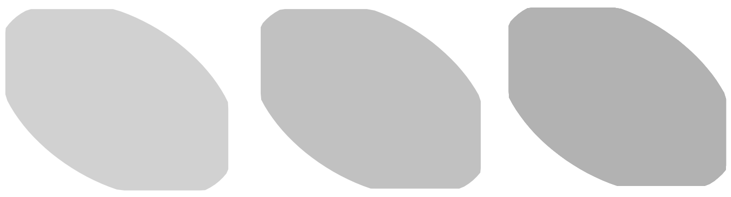
[5] Matlab is a registered trademark of the MathWorks,Inc.



$\{-2.0493 \leq x \leq 2.0493,\ -1.6733 \leq v \leq 1.6733,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

$\{-2.0462 \leq x \leq 2.0426,\ -1.665 \leq v \leq 1.665,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

$\{-1.9838 \leq x \leq 1.9838,\ -1.6097 \leq v \leq 1.6097,\ 2x^2 + 3v^2 + 2xv \leq 7\}$



$\{-1.8971 \leq x \leq 1.8971,\ -1.5435 \leq v \leq 1.5435,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

$\{-1.8718 \leq x \leq 1.8718,\ -1.5275 \leq v \leq 1.5275,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

$\{-1.8708 \leq x \leq 1.8708,\ -1.5275 \leq v \leq 1.5275,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

Fig. 7. Successive templates along policy iteration, at control point 2, for the harmonic oscillator.

This method is to be compared with the classical Kleene iteration with widening. On this example, we find without widening $x \in [-1.87078, 1.87083]$, $v \in [-1.52753, 1.52752]$ and $2x^2+3v^2+2xv \leq 7$ in 1360 iterations (for an overall time, under `Matlab` of 69 minutes).

4.5 Benchmarks

We implemented an analyzer for the quadratic template domain we presented, written in Matlab version 7.7(R2008b). This analyzer takes a text file in argument, this text file corresponds to the abstract equation $v = F^\sharp(v)$ where $F^\sharp$ is defined by Equation (2). The quadratic template can be loaded from a `dat` file by the analyzer. The affine maps are treated in the same manner.

In this analyzer, we can choose to use the Kleene iteration method or policy iteration. For the Kleene iteration method, the user gives as an argument a maximal number of iteration and the iteration number at which the acceleration method is applied. For the policy iteration method, the user gives the `dat` file defining the initial policy or chooses to make Kleene iterations before determining the initial policy.

Each ten steps during policy iteration, the user can decide to stop the analysis and so a postfixpoint is reached (as in policy iteration the least fixed point is always reached from above). Similarly, the Kleene iteration with acceleration provides a postfixpoint after acceleration and widening to top, if the iteration does not converge after a given number of iterations. The analyzer writes, in a text file, information about time, quality of the invariants found and number of iterations.

For the benchmarks, we used a PC equipped with a quad core AMD Phenom(tm) II X4 920 Processor at 2.8 Ghz and a memory of 4 Gb. We indicate in the Table 8, the name of the program analyzed, the method used (policy iteration or Kleene iteration) for solving the fixpoint equation, the cardinality of the basis of quadratic templates used, the number of lines of C code the program has, the number of variables it manipulates, the number of loops. Then we indicate the number of iterations made, whether it reaches a fixpoint or (strictly) a postfixpoint, and the time it took with our Matlab prototype.

Programs	Method	#P	#lines	#var	#loops	#Iter.	Inv. quality	Time
Rotation2	Policy	2	2	2	0	0	Fixpoint	0.72
Rotation2	Kleene	2	2	2	0	1	Fixpoint	1.07
Rotation10	Policy	2	2	10	0	0	Fixpoint	1.17
Rotation10	Kleene	2	2	10	0	1	Fixpoint	1.82
Filter	Policy	5	3	2	1	2	Fixpoint	9.35
Filter	Kleene	5	3	2	1	2	Fixpoint	19.7
Oscillator	Policy	5	3	2	1	5	Fixpoint	12
Oscillator	Kleene	5	3	2	1	15	Fixpoint	18.8
Symplectic	Policy	5	3	2	1	0	Fixpoint	3
Symplectic	Kleene	5	3	2	1	15	Fixpoint	18.3
SymplecticSeu	Policy	5	5	2	1	30	Postfixpoint	125.3
SymplecticSeu	Kleene	5	5	2	1	30	Postfixpoint	78.9
Arrow-Hurwicz	Policy	2	14	4	3	10	Postfixpoint	44.6
Arrow-Hurwicz	Kleene	2	14	4	3	26	Postfixpoint	81.7

Fig. 8. Benchmarks results

The file Rotation10 is the problem of Example 5 in dimension 10. By the fixpoint computation, we prove automatically that the unit sphere in dimension 10 is invariant by rotation. Both Kleene iteration and policy iteration find the unit sphere as invariant.

The program Oscillator is the problem 1. The invariant depicted Figure 1 in Section 1 is found by policy iteration whereas Kleene iteration after applying acceleration techniques from the iteration 5 to iteration 15 finds the less precise invariant $\{-2.44949 \leq x \leq 2.44949,\ -2 \leq v \leq 2,\ 2x^2 + 3v^2 + 2xv \leq 10\}$, in more time.

Symplectic is the implementation of a discretization of $\ddot{x} + c\dot{x} + x = 0$ with $c = 0$ by a symplectic method. In the case of $c = 0$, the dynamical system has imaginary eigenvalues (its orbits are circle), and the Euler scheme diverges, so we use a symplectic discretization scheme (preserving the symplectic form, see [HLW03]), which is an interesting highly degenerate numerical example from the point of view of static analysis (because there is no "stability margin", methods not exploiting the Lyapunov function are likely to produce trivial invariants when $c = 0$). As in Oscillator, we start from a position $x \in [0, 1]$ and a speed $v \in [0, 1]$. The discretization of $\ddot{x} + x = 0$ with the symplectic method and a step $\tau = 0.1$ gives us the matrix T such that $T_{1,1} = 1 - \frac{\tau}{2}$, $T_{1,2} = \tau - \frac{\tau^3}{4}$, $T_{2,1} = -\tau$ and $T_{2,2} = 1 - \frac{\tau}{2}$. We use the Lyapunov function L such that

$L(x,v) = (x,v)Q(x,v)^T$ with $Q = \begin{pmatrix} 1 & 0 \\ 0 & 1-\frac{\tau^2}{4} \end{pmatrix}$. The symplectic method ensures that $L(T(x,v)) = L(x,v)$. Our method takes advantage of this conservation law, since L is embedded as a template. The policy iteration returns: $\{-1.41333 \leq x \leq 1.41333,\ -1.4151 \leq v \leq 1.4151,\ x^2 + 0.9975v^2 \leq 1.9975\}$. The Kleene iteration returns: $\{-3.16624 \leq x \leq 3.16624,\ -3.16628 \leq v \leq 3.16628,\ x^2 + 0.9975v^2 \leq 10\}$, which is less precise. In particular, the Kleene algorithm misses the invariance of the Lyapunov function.

SymplecticSeu is a symplectic method with a threshold on $v = \dot{x}$. We iterate the Symplectic method while $v \geq \frac{1}{2}$, which gives the following code:

```
x = [0,1];
v = [0,1];
tau = 0.1 [1]
while [2] ((v>=1/2) [3]) { [4]
  x = (1-tau/2)*x+(tau-(tau^3)/4)*v;
  v = -tau*x+(1-tau/2)*v; [5]
};
```

Arrow-Hurwicz is an algorithm to compute both primal and dual solutions for convex constrained optimization problems. Arrow-Hurwicz ends when a fixpoint for the algorithm is reached, by our techniques, we prove that, if the last line of the program which implements the Arrow-Hurwicz method is attained, a fixpoint is reached. Our analysis also permits to find bounds at each control points. As pointed out in the introduction, the interest of the analysis resides in the appearance of saturations (non-linear projections) in the scheme. For both Kleene iteration and policy iteration, the invariant set of last line is $\{0 \leq \frac{11}{16}(u-x)^2 + (v-y)^2 \leq 1e-9\}$. The difference between the two final invariants comes from other lines where the invariant found by policy iteration is always smaller than the set found by Kleene, for example, when policy iteration returns, for example at line 11, $\{0 \leq \frac{11}{16}(u-x)^2 + (v-y)^2 \leq 3.18292\}$, Kleene returns $\{0 \leq \frac{11}{16}(u-x)^2 + (v-y)^2 \leq 10\}$.

The example files are available at:
`http://www.lix.polytechnique.fr/~adje/publi-presentations.html`.

5 Conclusion and future work

We have presented in this paper a generalization of the linear templates of Manna et al. [SSM05,SCSM06] that can also deal with non-linear templates. We showed that in the case of quadratic templates, we could efficiently abstract the semantic functionals using Shor's relaxation, and compute the resulting fixpoint using policy iteration. Future work include the use of more tight relaxations for quadratic problems. The use of SOS relaxation (see for instance [Las07] and [Par03]) for dealing with more general polynomial templates will be also considered. An other problem is to extend of the minimality result of [AGG08] which is currently only available for the interval domain, to our template domain. Finally, we wish to study more in-depth the complexity issues raised by our general policy iteration algorithm.

Acknowledgement. We thank Thomas Gawlitza and David Monniaux for their remarks on an earlier version of this paper.

References

[AGG08] A. Adje, S. Gaubert, and E. Goubault. Computing the smallest fixed point of nonexpansive mappings arising in game theory and static analysis of programs. Technical report, arXiv:0806.1160, Proceedings of MTNS'08, Blacksburg, Virginia, July 2008.

[Ali95] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51, 1995.

[AT03] A. Auslender and M. Teboulle. *Asymptotic Cones and Functions in Optimization and Variational Inequalities.* Springer, 2003.

[BRCZ05] R. Bagnara, E. Rodríguez-Carbonell, and E. Zaffanella. Generation of basic semi-algebraic invariants using convex polyhedra. In C. Hankin, editor, *Static Analysis: Proceedings of the 12th International Symposium*, volume 3672 of *LNCS*, pages 19–34. Springer, 2005.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of 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.

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

[Cou05] P. Cousot. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In *Sixth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *LNCS*, pages 1–24. Springer, 2005.

[FA08] E. Feron and F. Alegre. Control software analysis, part II: Closed-loop analysis. Technical report, arXiv:0812.1986, 2008.

[Fer05] J. Feret. Numerical abstract domains for digital filters. In *International workshop on Numerical and Symbolic Abstract Domains (NSAD 2005)*, 2005.

[FF08] E Feron and Alegre F. Control software analysis, part I: Open-loop properties. Technical report, arXiv:0809.4812, 2008.

[GGTZ07] 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.

[GPBG08] E. Goubault, S. Putot, P. Baufreton, and J. Gassino. Static analysis of the accuracy in control systems: Principles and experiments. In *Formal Methods for Industrial Critical System (FMICS 2007)*, volume 4916 of *LNCS*, pages 3–20, 2008.

[GS07a] T. Gawlitza and H. Seidl. Precise fixpoint computation through strategy iteration. In R. De Nicola, editor, *Programming Languages and Systems, 16th European Symposium on Programming, ESOP 2007*, volume 4421 of *LNCS*, pages 300–315. Springer, 2007.

[GS07b] T. Gawlitza and H. Seidl. Precise relational invariants through strategy iteration. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *LNCS*, pages 23–40. Springer, 2007.

[HLW03] E. Hairer, C. Lubich, and G. Wanner. Geometric numerical integration illustrated by the Störmer/Verlet method. *Acta Numerica*, 12:399–450, 2003.

[JCK07] C. Jansson, D. Chaykin, and C. Keil. Rigorous error bounds for the optimal value in semidefinite programming. *SIAM J. Numer. Anal.*, 46(1):180–200, 2007.

[Kei05] C. Keil. Lurupa - rigorous error bounds in linear programming. In *Algebraic and Numerical Algorithms and Computer-assisted Proofs*, 2005. `http://drops.dagstuhl.de/opus/volltexte/2006/445`.

[Las07] J.-B. Lasserre. A sum of squares approximations of nonnegative polynomials. *SIAM Review*, 49(4):651–669, 2007.

[L04] J. Lfberg. Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. `http://control.ee.ethz.ch/~joloef/yalmip.php`.

[Min04a] A. Miné. Relational abstract domains for the detection of floating-point run-time errors. In *Proc. of the European Symposium on Programming (ESOP'04)*, volume 2986 of *LNCS*, pages 3–17. Springer, 2004.

[Min04b] A. Miné. *Weakly Relational Numerical Abstract Domains.* PhD thesis, École Polytechnique, Palaiseau, France, December 2004. `http://www.di.ens.fr/~mine/these/these-color.pdf`.

[MOS04] M. Müller-Olm and H. Seidl. Computing polynomial program invariants. *Inf. Process. Lett.*, 91(5):233–244, 2004.

[Par03] P. Parillo. Semidefinite programming relaxations for semialgebraic problems. *Math. Prog.*, 96(2, series B):293–320, 2003.

[RCK07] E. Rodríguez-Carbonell and D. Kapur. Automatic generation of polynomial invariants of bounded degree using abstract interpretation. *Sci. Comput. Program.*, 64(1):54–75, 2007.

[Roc96] R.T. Rockafellar. *Convex Analysis.* Princeston University Press, 1996.

[SCSM06] S. Sankaranarayanan, M. Colon, H. Sipma, and Z. Manna. Efficient strongly relational polyhedral analysis. In E. Allen Emerson and Kedar S. Namjoshi, editors, *Verification, Model Checking, and Abstract Interpretation: 7^{th} International Conference, (VMCAI)*, volume 3855 of *LNCS*, pages 111–125, Charleston, SC, January 2006. Springer.

[Sho87] N. Shor. Quadratic optimization problems. *Soviet J. of Computer and Systems Science*, 25(6):1–11, 1987.

[SSM05] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *Sixth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *LNCS*, pages 25–41, January 2005.

[Stu99] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[TN01] A. Ben Tal and A. Nemirowski. *Lecture on Modern Convex Optimization: Analysis, Algorithm and Engineering Applications.* SIAM, 2001.