A fast method to compute disjunctive quadratic invariants of numerical programs

Xavier Allamigeon
INRIA and CMAP, École polytechnique, CNRS
France
xavier.allamigeon@inria.fr

Stéphane Gaubert
INRIA and CMAP, École polytechnique, CNRS
France
stephane.gaubert@inria.fr

Eric Goubault
LIX, École polytechnique, CNRS
France
eric.goubault@lix.polytechnique.fr

Sylvie Putot
LIX, École polytechnique, CNRS
France
sylvie.putot@lix.polytechnique.fr

Nikolas Stott
INRIA and CMAP, École polytechnique, CNRS
France
nikolas.stott@inria.fr

ABSTRACT

We introduce a new method to compute non-convex invariants of numerical programs, which includes the class of switched affine systems with affine guards. We obtain disjunctive and non-convex invariants by associating different partial execution traces with different ellipsoids. A key ingredient is the solution of non-monotone fixed points problems over the space of ellipsoids with a reduction to small size linear matrix inequalities. This allows us to analyze instances that are inaccessible in terms of expressivity or scale by earlier methods based on semi-definite programming.

CCS CONCEPTS

• **Software and its engineering** → **Formal methods**; • **Theory of computation** → *Invariants*;

KEYWORDS

Static analysis, abstract interpretation, invariant generation, stability, hybrid and switched linear systems

ACM Reference format:
Xavier Allamigeon, Stéphane Gaubert, Eric Goubault, Sylvie Putot, and Nikolas Stott. 2017. A fast method to compute disjunctive quadratic invariants of numerical programs. In *Proceedings of Embedded Systems Week, Seoul, South Korea, October 2017 (EMSOFT'17)*, **??** pages.
DOI:

1 INTRODUCTION

Motivation. A basic problem in formal verification consists in computing invariants of numerical programs. The latter are omnipresent in embedded systems and control-command software. For the analysis of such programs, the existing methods range from polyhedral domains [**?**], including explicit or tractable subclasses with a constrained geometry [**? ?**], to domains of quadratic, or piecewise-quadratic invariants, involving ellipsoids [**? ? ? ? ? ? ? ?**], or even higher degree polynomial invariants [**? ? ?**].

The analysis of numerical programs is often challenging. Even for a simple program which iteratively switches between two linear dynamics, reachability is already undecidable [**?**]. In the case of switched linear systems, however, ellipsoids are a natural choice. Indeed, they are known to provide optimal (Lyapunov-type) stability certificates for unswitched linear systems. Moreover, efficient algorithms based on semi-definite programming allow one to propagate ellipsoids through the analysis. Unfortunately, in the switched case, a single ellipsoid leads to extremely conservative invariants.

Contribution. In this paper, we obtain disjunctive and non-convex invariants for numerical programs, by exploiting the good geometric and algorithmic properties of the space of ellipsoids. To do so, we develop a new method. It relies on associating an ellipsoid with each partial execution trace. This leads to a disjunctive, possibly non-convex, invariant of the program as a union of ellipsoids.

While disjunctive analyses generally suffer from a computational blow up, our method preserves scalability by expressing the invariant as the union of a prescribed number of ellipsoids. These ellipsoids are obtained efficiently by solving a non-linear and non-monotone fixed point problem over the space of ellipsoids. Indeed, every fixed point iteration is implemented by solving linear matrix inequalities (LMI) of relatively small size, depending on the number of program variables but not the number of terms in the disjunction.

Our method is flexible enough to handle programs that *switch* between several *affine assignments* of the form $x \leftarrow A_i x + B_i u + c_i$, where x is the state vector and u represents a control or uncertainty vector. The switching process can be *arbitrary* (or non-deterministic) or *state-dependent* via *affine guards*.

We report experiments showing that our method can handle instances of switched systems out of reach of earlier methods [? ? ? ?], demonstrating the benefit brought by the present "small-LMI" and "no-LMI" approaches in terms of scalability, expressivity, and accuracy compared to earlier methods based on ellipsoids.

Related work. In static analysis by abstract interpretation, ellipsoidal methods have been proposed by a number of authors: for accurately analyzing linear filters [?], for control programs and switched systems [?], for linear time invariant systems [?], using policy iteration [? ?], with clever widening techniques [?], or even using piecewise quadratic invariants [?]. Our work is also inspired by control theory [?], where intersections of ellipsoids are used to approximate the joint spectral radius of linear switched systems without guards. Intersection of ellipsoids and piecewise quadratic invariants (like multiple Lyapunov functions [?]) provide good invariants for systems that have a single equilibrium, i.e. switching between assignments of the form $x \leftarrow A_i x + B_i u + c_i$ with $c_i = 0$. When $c_i \neq 0$, those assignments do not have the same equilibrium point in general, and there is no guarantee that a "small" convex invariant exists (since affine systems behave asymptotically like linear systems, there must be a convex invariant which can be very large). Therefore, our method deals with *non-convex* invariants.

Moreover, our work falls in the category of disjunctive analyses [?] constructed on a given abstract domain (the ellipsoidal domain of [?]). Although very appealing in theory (see [?]), disjunctive analyses may lead to very expensive analyses in practice. Several proposals have been made to control the potential explosion of the number of disjunctions, manipulated during an analysis, see e.g. [? ? ? ?]. In order to manage a tractable set of disjunctions of abstract elements, several authors have proposed methods involving a transformation of the original program. This includes semantic techniques such as initial unfolding of loops (i.e. the copy of n times the loop body, before the loop itself), as used in e.g. [?] and loop unfolding (i.e. n duplications of the loop body, within the loop). This has been applied for instance to synchronous languages in [?] and in [?] for improving the precision of numerical analyses of imperative programs. Some authors have also proposed to duplicate the abstract elements attached to single statement according to the values of binary variables (that typically control conditionals, or modes of the program), as in BDD APRON [?].

In this paper, we make a similar elaboration by choosing an abstraction of the traces, on which we perform trace partitioning [?] through an automaton. Contrarily to the existing elaborations, ours can be seen as identifying traces with a common suffix.

This has a number of advantages. First, our method is separating out invariants because of the recent "history": for stable iterative numerical systems, this is much better in the sense that the older the values that are computed in traces of executions, the fewer effect it has on the outcome of the computation. Moreover, our method has only to solve a number of *small* LMI problems, instead of one big LMI system as in most classical methods, making it much more tractable in practice. We compare our method with existing methods [? ?] based on ellipsoidal invariants at the end of Section 5.2.

Program 1: Switched affine program with guards (each $\diamond$ belongs to $\{\leqslant, <\}$)

```
$y \leftarrow \mathcal{E}_I$;
while true do
    $x := y$;
    $u \leftarrow \mathcal{E}_U$;
    if $(f_{1,1}^T x \diamond g_{1,1}) \wedge \cdots \wedge (f_{1,P}^T x \diamond g_{1,P})$ then
        $y := A_1 \times x + B_1 \times u + c_1$;
    end
    ⋮
    if $(f_{N,1}^T x \diamond g_{N,1}) \wedge \cdots \wedge (f_{N,P}^T x \diamond g_{N,P})$ then
        $y := A_N \times x + B_N \times u + c_N$;
    end
end
```

An essential originality of our method is the solution of a non-monotone fixed point problem involving Löwner ellipsoids. This differs from classical static analysis by abstract interpretation which lead to monotone fixed point problems. In the special linear case, the convergence proof relies on properties of metric geometry. We use a canonical metric on the cone of positive matrices, Riemann's metric, and prove that the iterative scheme solving the non-monotone fixed point problem is a contraction.

Comparison with earlier methods. The present method computes a program invariant as the union of a finite number of ellipsoids. Several works have proposed to compute quadratic invariants of numerical programs as a single ellipsoid, obtained by iterative schemes [?], by semi-definite programming [?] or policy iteration [?]. These methods usually produce invariants that are less accurate than ours given the disjunctive nature of our approach.

Other methods compute invariants as the intersection of ellipsoids [?] or as piecewise quadratic functions [?], i.e. quadratic functions defined on polyhedral cells. These methods are based on partitioning into finite execution traces. Additionally, they are restricted to the case of linear assignments ($c_1 = c_2 = 0$). As a consequence, when compared to our method, they are less expressive, and more expensive when the size of the automaton is large, since they are expressed as solution of large LMIs.

2 PROGRAMS OF INTEREST

We are interested in programs of the form of Program 1, involving several affine switching conditions within a loop. This class of programs includes programs that simulate switched affine systems with affine guards.

For simplicity, we assume that all variables have global scope. We denote by x (resp. y) the vector containing the *state variables* $x_1, \ldots, x_n$ (resp. $y_1, \ldots, y_n$) and by $\mathcal{E}_I$ an ellipsoid that over-approxi-mates of the set of initial values for the state variables in y. We refer the reader to the next section for background on ellipsoids. We also denote by u the vector of *input variables* $u_1, \dots, u_m$, that may represent values measured from a sensor. The set of possible values of the vector u is over-approximated by an ellipsoid $\mathcal{E}_\mathcal{U}$. The operations $+$ and $\times$ are vector or matrix operations. A vector assignment is denoted $(x_1, \dots, x_n) := (y_1, \dots, y_n)$, or $x := y$ for short. The non-deterministic choice of a value for the vector x inside a set $\mathcal{X}$ is denoted $x \leftarrow \mathcal{X}$. In Program 1, $f_{i,j}$ is a vector of dimension n and $g_{i,j}$ is a real number; so $\wedge_j(f_{i,j}^T x \diamond g_{i,j})$, where $\cdot^T$ denotes the transposition and each occurrence of $\diamond$ can be replaced either by $\leqslant$ or $<$, is a conjunction of affine guard conditions. In each assignment, A_i is an $n \times n$ matrix, B_i is an $n \times m$ matrix and c_i is a vector of dimension n.

In most applications we have in mind, the switching conditions are mutually exclusive, meaning exactly one switching condition is valid for any value of the variable vector x. We point out that this assumption can be made without loss of generality, up to adding more conditional statements within the loop and adjusting the assignments accordingly.

We shall also consider the somehow simpler variant of this program in which every guard condition $(f_{i,j}^T x \diamond g_{i,j})$ is replaced by the test of a random boolean, and refer to it as a *non-deterministic switched system.*

An invariant for Program 1 is defined as a set $\mathcal{I}$ that satisfies $\mathcal{E}_I \subseteq \mathcal{I}$, and, for all i, j, $x \in \mathcal{I}$ and $u \in \mathcal{U}$,

$$A_i x + B_i u + c_i \in \mathcal{I} \quad \text{whenever } f_{i,j}^T x \diamond g_{i,j}\,.$$

(In the non-deterministic case, the condition $f_{i,j}^T x \diamond g_{i,j}$ is dropped.)

3 LÖWNER ELLIPSOID APPLIED TO ABSTRACT INTERPRETATION

In this section, we introduce the domain of ellipsoids and several operations on ellipsoids that are needed in our analysis.

We begin by recalling some notation. The set of real $m \times n$ matrices is denoted by $\mathcal{M}_{m,n}$, and the set of real $n \times n$ matrices is abbreviated as $\mathcal{M}_n$. The $n \times n$ identity matrix is denoted by I_n.

A matrix M is *symmetric* if $M = M^T$. The set of $n \times n$ symmetric matrices is denoted $\mathcal{S}_n$. A symmetric matrix $M = (M_{ij})$ is called *positive semi-definite* if $x^T M x = \sum_{i,j} M_{ij} x_i x_j \geqslant 0$ for all real vectors $x = (x_i)$. This is equivalent to the existence of a matrix $L \in \mathcal{M}_n$ such that $M = LL^T$. When the matrix M is positive semi-definite, we write $M \succcurlyeq 0$, and we denote by $\mathcal{S}_n^+$ the set of positive semi-definite matrices. When $x^T M x > 0$ holds for all non-zero vectors $x \in \mathbb{R}^n$, we say that the matrix M is *positive definite*. We denote the set of positive definite matrices by $\mathcal{S}_n^{++}$. We extend $\preccurlyeq$ into an order relation over $\mathcal{S}_n$ as follows: given $A, B \in \mathcal{S}_n$, we say that $A \preccurlyeq B$ when the matrix $B - A$ is positive semi-definite. This order relation is called the *Löwner order*. Finally, a set $\mathcal{X} \subset \mathbb{R}^n$ is said to be full-dimensional if it is not contained in an affine subspace of codimension 1, or, equivalently, its affine hull coincides with $\mathbb{R}^n$.

A linear matrix inequality (LMI for short) refers to a constraint of the form

$$A_0 + \sum_{k=1}^{d} x_k A_k \succcurlyeq 0\,, \tag{1}$$

where $x \in \mathbb{R}^d$ is the variable and $(A_k)_{1 \leqslant k \leqslant d}$ are given symmetric $n \times n$ matrices. In other words, given a symmetric matrix $A(x_1, .., x_d)$ whose entries depend in an affine way on $x \in \mathbb{R}^d$, the constraint $A(x_1, \dots, x_d) \succcurlyeq 0$ is an LMI. Several LMIs can be combined into a single LMI, since $\begin{pmatrix} A(x) & 0 \\ 0 & B(x) \end{pmatrix} \succcurlyeq 0$ is equivalent to $A(x) \succcurlyeq 0$ and $B(x) \succcurlyeq 0$. The problem of minimizing a convex function in the variable x that satisfies the LMI in Equation (1) is called a semi-definite program (SDP). We refer to [?] for introductive background on these programs.

Semi-definite programs can be solved in "polynomial time" in the following approximate sense (semi-definite feasibility is not known to be polynomial time in the Turing model of computation). Given an accuracy parameter $\varepsilon > 0$, one can obtain, in particular by interior point methods (the most efficient in practice), a ε-approximate solution of a SDP in a number of arithmetic operations which is polynomial in n, d, $\log \varepsilon$, and $\log(R/r)$, assuming that the set $\mathcal{F}$ of vectors which satisfy (1) is such that $B(a, r) \subset \mathcal{F} \subset B(a, R)$ for some point $a \in \mathbb{R}^n$, where $B(a, r)$ denotes the Euclidean ball of center a and radius r, see [?]. We warn the reader, however, that the exponent of the polynomial is relatively high (see Section 5.1 for details). Hence, it is essential for scalability purposes to limit as far as possible the growth of the dimension n and of the number of variables d, which is one of our main goals in what follows.

We can now introduce ellipsoids. Let $\mathcal{B}_n = \{x \in \mathbb{R}^n : x^T x \leqslant 1\}$ denote the unit ball of $\mathbb{R}^n$. An *ellipsoid* is defined as the image of the unit ball $\mathcal{B}_n$ under an affine map $x \mapsto Lx + q$, where $L \in \mathcal{M}_{p,n}$ and $q \in \mathbb{R}^p$. When the matrix L is invertible, the matrix $Q = LL^T$ is positive definite and the ellipsoid $\mathcal{E}(Q, q)$ is given by

$$\mathcal{E}(Q, q) \coloneqq \{y \in \mathbb{R}^p \colon (y - q)^T Q^{-1} (y - q) \leqslant 1\}\,.$$

When the matrix L is not invertible, the matrix Q is only positive semi-definite, so we have $\mathcal{E}(Q, q) = \{y \in \mathbb{R}^p \colon (y - q)(y - q)^T \preccurlyeq Q\}$ as shown in [?]. When the matrix Q is positive definite, the ellipsoid $\mathcal{E}(Q, q)$ is full-dimensional, whereas the ellipsoid $\mathcal{E}(Q, q)$ is "flat" when Q is only positive semi-definite. We say that the ellipsoid $\mathcal{E}(Q, q)$ is *centered* if $q = 0$. The volume of a full-dimensional ellipsoid is proportional to $\det L = (\det Q)^{1/2}$ (the proportionality constant only depends on the dimension). We denote the set of ellipsoids in $\mathbb{R}^n$ by $\mathfrak{E}_n$.

We recall a famous result by Löwner [?].

Theorem 3.1 (Löwner [?]). *Given a compact and full-dimen-sional set $\mathcal{X} \subset \mathbb{R}^n$, there is a unique ellipsoid $\mathcal{E}(Q, q)$ that contains $\mathcal{X}$ and that has minimum volume.*

This ellipsoid is called the *Löwner ellipsoid* of the set $\mathcal{X}$, and we denote it by $\text{Löw}(\mathcal{X})$.

In our analysis, we need to be able to manipulate the union of ellipsoids, the Minkowski sum of ellipsoids and the intersection of an ellipsoid with a half-space, none of which are ellipsoids in general. In the spirit of abstract interpretation, we shall replace these operations by abstract operators that return ellipsoidal values. These abstract operators need to be sound, meaning that they must over-approximate their concrete counterpart. In order to evaluate most of these operators, reductions by semi-definite programming, which rely on the S-Lemma and classical tools in linear algebra such as Schur complement, have been proposed in [? ?]. We next recall these reductions for the sake of completeness.

Inclusion. We can check if an ellipsoid $\mathcal{E}(Q_1, q_1)$ is included in the ellipsoid $\mathcal{E}(Q_2, q_2)$. When the matrix Q_2 is positive definite, checking the inclusion $\mathcal{E}(Q_1, q_1) \subseteq \mathcal{E}(Q_2, q_2)$ is equivalent to checking if the following LMI has a solution as proved in [?]:

$$\exists \lambda \in \mathbb{R} \colon \begin{pmatrix} Q_2 & q_1 - q_2 & L_1 \\ (q_1 - q_2)^T & 1 - \lambda & 0_{1,n} \\ L_1^T & 0_{n,1} & \lambda I_n \end{pmatrix} \succcurlyeq 0 \,,$$

where L_1 satisfies $Q_1 = L_1 L_1^T$. In the special case where $q_1 = q_2$, this amounts to checking if $Q_1 \preccurlyeq Q_2$. We shall see later on that the case where the ellipsoid $\mathcal{E}(Q_2, q_2)$ is not full dimensional does not arise in our analysis.

Image by an affine map. Let $f \colon x \mapsto Ax + b$ denote an affine map, with $A \in \mathcal{M}_{p,n}$ and $b \in \mathbb{R}^p$. The image of the ellipsoid $\mathcal{E}(Q, q) \in \mathfrak{E}_n$ by the affine map f is again an ellipsoid, given by $f\big(\mathcal{E}(Q, q)\big) \coloneqq \mathcal{E}(AQA^T, Aq + b) \in \mathfrak{E}_p$.

Union of ellipsoids. A celebrated theorem of Kadison [?] implies that the space of centered ellipsoids, equipped with the inclusion order, is as far as possible from a lattice: it is an *anti-lattice*, meaning that two centered ellipsoids have a least upper bound if and only if they are comparable. Similarly, the set of (not necessarily centered) ellipsoids equipped with the inclusion order does not constitute a lattice. This means that, given ellipsoids $\mathcal{E}_1, \dots, \mathcal{E}_p$, there is generally not a single ellipsoid $\mathcal{E}$ that is the smallest among ellipsoids that contain $\cup_k \mathcal{E}_k$. For this reason, we over-approximate the union of a finite number of ellipsoids by the Löwner ellipsoid $\text{Löw}\big(\cup_k \mathcal{E}_k\big)$. The latter can be computed as $\mathcal{E}(Y^{-2}, Y^{-1}y)$, where (Y, y) is the optimal solution of the following semi-definite program [?]:

$$\begin{aligned} \operatorname*{argmin}_{Y,y} \quad & -\log\det Y \\ \text{subject to} \quad & \begin{pmatrix} I_n & (Yq_k - y) & YL_k \\ (Yq_k - y)^T & 1 - \lambda_k & 0_{1,n} \\ L_k^T Y & 0_{n,1} & \lambda_k I_n \end{pmatrix} \succcurlyeq 0 \,,\ \forall k \\ & Y \succcurlyeq 0 \end{aligned} \tag{2}$$

and $\mathcal{E}_k = \mathcal{E}(L_k L_k^T, q_k)$. This is only true when the convex hull of $\cup_k \mathcal{E}_k$ is full-dimensional (the opposite case will not arise in our analysis). For the sake of brevity, we denote this Löwner ellipsoid by

$$\sqcup_k \mathcal{E}_k \coloneqq \text{Löw}\big(\cup_k \mathcal{E}_k\big) \,. \tag{3}$$

It will be convenient to write in infix form, $\mathcal{E}_1 \sqcup \dots \sqcup E_p$ instead of $\sqcup_k \mathcal{E}_k$, noting that this is an abuse of notation, since the operation $\sqcup$ is not associative.

Minkowski sum of ellipsoids. Recall that the Minkowski sum of two sets X, Y is the set $X + Y \coloneqq \{x + y \colon x \in X, y \in Y\}$. Like the case of the union of ellipsoids, we over-approximate the Minkowski sum of two ellipsoids $\mathcal{E}_0, \mathcal{E}_1$ by its Löwner ellipsoid, and denote it by

$$\mathcal{E}_0 \boxplus \mathcal{E}_1 \coloneqq \text{Löw}\big(\mathcal{E}_0 + \mathcal{E}_1\big) \,.$$

It has been shown in [?] that, given two full-dimensional ellipsoids $\mathcal{E}(Q_1, q_1)$ and $\mathcal{E}(Q_2, q_2)$, the Löwner ellipsoid of $\mathcal{E}(Q_1, q_1) + \mathcal{E}(Q_2, q_2)$ is then equal to $\mathcal{E}(Z^{-1}, q_1 + q_2)$, where Z is the solution of the semi-definite program

$$\begin{aligned} \operatorname*{argmin}_{Z,\lambda} \quad & \log\det Z^{-1} \\ \text{subject to} \quad & \begin{pmatrix} \lambda Q_1^{-1} & 0_n \\ 0_n & (1 - \lambda) Q_2^{-1} \end{pmatrix} \succcurlyeq \begin{pmatrix} Z & Z \\ Z & Z \end{pmatrix} \\ & Z \succcurlyeq 0,\ 0 \leqslant \lambda \leqslant 1 \,. \end{aligned} \tag{4}$$

Intersection between ellipsoid and half-space. A *half-space* is defined as the set $\mathcal{H}(f, g) = \{x \in \mathbb{R}^n \colon f^T x \leqslant g\}$, where $f \in \mathbb{R}^n$ is a non-zero vector and g is a real number. Given an ellipsoid $\mathcal{E}$, we over-approximate its intersection with the half-space $\mathcal{H}$ by its Löwner ellipsoid, and we denote it by

$$\mathcal{E} \sqcap \mathcal{H} \coloneqq \text{Löw}\big(\mathcal{E} \cap \mathcal{H}\big) \,.$$

When $\mathcal{E}$ is full-dimensional (we shall see later that it is always the case in our computations), the set $\mathcal{E} \sqcap \mathcal{H}$ can be computed analytically following [?]. We give the formula below for the sake of completeness. Given an ellipsoid $\mathcal{E}(Q, q)$ and a half-space $\mathcal{H}(f, g)$, let α denote the quantity $\alpha \coloneqq (f^T Q f)^{-1}(g - f^T q)$. If $\alpha \geqslant 1/n$, we have $\mathcal{E} \sqcap \mathcal{H} = \mathcal{E}$. If $\alpha < -1$, then $\mathcal{E} \sqcap \mathcal{H} = \emptyset$. If $-1 \leqslant \alpha \leqslant 1/n$, we have $\mathcal{E} \sqcap \mathcal{H} = \mathcal{E}(Q^+, q^+)$, with $q^+ = q - (1 + n)^{-1}(1 - n\alpha)Qf$ and

$$Q^+ = \frac{n^2(1 - \alpha^2)}{n^2 - 1} \Big(Q - 2 \frac{1 - n\alpha}{(1 + n)(1 - \alpha)} (Qf)(Qf)^T \Big) \,.$$

The intersection with several half-spaces is handled in a sequential way, meaning that we evaluate $\text{Löw}\big(\mathcal{E} \cap (\mathcal{H} \cap \mathcal{H}')\big)$ as $(\mathcal{E} \sqcap \mathcal{H}) \sqcap \mathcal{H}'$. It will again be convenient to do an abuse of notation, denoting the latter operation by $\mathcal{E} \sqcap (\mathcal{H} \cap \mathcal{H}')$. Although this evaluation remains sound, it may yield a very coarse over-approximation, since the maps $\mathcal{E} \mapsto E \sqcap \mathcal{H}$ and $\mathcal{E} \mapsto \mathcal{E} \sqcap \mathcal{H}'$ do not commute in general. When several half-spaces are involved, finding a better over-approximation is a difficult and intractable problem, see [? , Section 3.7].

The Löwner ellipsoid has the following invariance property:

Proposition 3.2. *The Löwner ellipsoid commutes with invertible affine transformations: given a compact full-dimensional* *set $\mathcal{X} \subset \mathbb{R}^n$ and an invertible affine map $f\colon \mathbb{R}^n \to \mathbb{R}^n$, we have*

$$f\big(\text{Löw}(\mathcal{X})\big) = \text{Löw}\left(f(\mathcal{X})\right) .$$

We give a proof in Appendix **??**.

The Minkowski sum, union and intersection are operators that commute with the action of invertible affine maps. By Proposition 3.2, this is also the case for the Löwner ellipsoid. As a consequence, the operators $\sqcup$, $\sqcap$ and $\boxplus$ that we have defined do also commute with invertible affine maps. Since those operators are the only ones that are used in the subsequent analysis, it follows that the invariants deduced for two systems that only differ by an affine change of variable will also only differ by the same affine transformation. This implies that the static analysis method which we next develop commutes with an affine rewriting of the program variables, which is a desirable robustness property.

4 A TRACE-DEPENDENT FIXED POINT SCHEME AVOIDING THE RECOURSE TO LARGE LMI

In this section, we present our approach to compute an invariant for Program 1 as the union of finitely many ellipsoids. It is obtained as the fixed point of a non-monotone map that is based on an automaton, whose states represent finite execution traces (in the case of De Bruijn automata, see Section 5.2, these distinguish between different suffixes of traces of the same length). As a consequence, we expect that the more states this automaton has (i.e. the more execution traces are taken into account during the computation), the more accurate the invariant to be. Moreover, since an ellipsoid is associated with each state of the automaton, the number of disjunctions in the invariant is constant during the computation.

We label the branches of the loop of Program 1 by integers from 1 to N, and denote by $\Sigma \coloneqq \{1, \dots, N\}$. We can thus identify the set of finite traces of the program with the set Σ^* of finite words built on the alphabet Σ. For all $a \in \Sigma$, we introduce the abstract operator guard_a over the set of ellipsoids $\mathfrak{E}_n$, given by:

$$\text{guard}_a(\mathcal{E}) \coloneqq \mathcal{E} \sqcap \left(\cap_j \mathcal{H}_{a,j} \right) ,$$

where $\mathcal{H}_{a,j}$ denotes the half-space $f_{a,j}^T x \leqslant g_{a,j}$. We also introduce an action of Σ^* on the set of ellipsoids $\mathfrak{E}_n$, denoted by $\cdot$ and defined for $a \in \Sigma$ and $\mathcal{E} \in \mathfrak{E}_n$ by:

$$a \cdot \mathcal{E} \coloneqq \left(f_a \circ \text{guard}_a(\mathcal{E})\right) \boxplus \left(B_a \mathcal{E}_{\mathcal{U}}\right) ,$$

where f_a denotes the affine map $x \mapsto A_a x + c_a$. In this way, the map $\mathcal{E} \mapsto a \cdot \mathcal{E}$ represents the abstract operator associated with the branch a of the program.

Let $\mathcal{W}$ denote a finite subset of Σ^* and τ denote a map from $\mathcal{W} \times \Sigma$ to $\mathcal{W}$. The triple $\mathcal{A} \coloneqq (\Sigma, \mathcal{W}, \tau)$ defines a deterministic finite automaton, whose alphabet is Σ, whose states are elements of $\mathcal{W}$ and whose transition function is τ. Every state in this automaton is both an initial and final state. The function τ being totally defined over $\mathcal{W} \times \Sigma$, the automaton $\mathcal{A}$ accepts every word of Σ^*.

We shall consider functions $\underline{\mathcal{E}}$ from the finite set of traces $\mathcal{W}$ to the space of ellipsoid $\mathfrak{E}_n$. We denote by $\mathcal{E}_w$ the ellipsoid associated with $w \in \mathcal{W}$ by this function. It will be convenient to identify $\underline{\mathcal{E}}$ to the vector $(\mathcal{E}_w)_{w \in \mathcal{W}}$ in $\mathfrak{E}_n^{\mathcal{W}}$ indexed by elements of w.

Given an automaton $\mathcal{A} = (\Sigma, \mathcal{W}, \tau)$, let T denote a map from $\mathfrak{E}_n^{\mathcal{W}}$ to itself, whose w-th coordinate is defined by

$$T_w(\underline{\mathcal{E}}) \coloneqq \mathcal{E}_I \sqcup \bigsqcup_{\tau(v,a)=w} a \cdot \mathcal{E}_v . \tag{5}$$

The fact that semi-definite programs can only be solved up to a prescribed accuracy is a well known source of difficulties in numerical program verification. If the approximate invariant which is found is mapped to its interior, meaning that some strictly feasible solution is returned by the SDP solver, then, an exact invariant can be obtained a posteriori by some rounding procedure, see the discussion in [?]. In order to make such methods applicable in the present setting, we introduce a small margin $\varepsilon > 0$ which will absorb numerical imprecisions as suggested by the authors in [?]. Hence, we define the perturbed map T^ε from $\mathfrak{E}_n^{\mathcal{W}}$ to itself, whose w-th coordinate is obtained by adding a "padding" εI_n to each ellipsoid:

$$T^\varepsilon_w(\underline{\mathcal{E}}) \coloneqq \mathcal{E}\big(Q_w + \varepsilon I_n, q_w\big) \quad \text{where} \quad \mathcal{E}(Q_w, q_w) = T_w(\underline{\mathcal{E}}) .$$

Introducing the parameter ε induces a trade-off between speed (ε large) and precision (ε small). The speed-up effect is shown in Equation (7), resulting from the complexity analysis in Appendix **??**. The loss of precision is due to the fact that the map T^ε is a "deformation" of the true map T. In the experiments, we have chosen $0.01 \leqslant \varepsilon \leqslant 0.2$.

These operators enable the computation of invariants as unions of ellipsoids:

Theorem 4.1. *Let $\underline{\mathcal{E}} = (\mathcal{E}_w)_{w \in \mathcal{W}}$ denote a fixed point of the map T^ε. Then the set $\cup_{w \in \mathcal{W}} \mathcal{E}_w$ is an invariant for Program 1.*

The same is true if $\underline{\mathcal{E}}$ is only a post-fixed point of the map T^ε, i.e. if for all $w \in \mathcal{W}$, we have $T^\varepsilon_w(\underline{\mathcal{E}}) \subseteq \mathcal{E}_w$. We give a proof of this theorem in Appendix **??**.

The map T^ε is the analogue of the fixed point functional in abstract interpretation. Classical abstract interpretation requires the fixed point functional to be a monotone map defined on a complete lattice [?]. Then, a program invariant can be obtained as the least fixed point of this functional, which can be computed by a standard fixed point scheme, *Kleene iteration*. The present setting is more complex, for the space $\mathfrak{E}_n^{\mathcal{W}}$ is not a lattice, and the operators $\sqcup$, $\sqcap$ and $\boxplus$ are not monotone (this can be quickly verified, even for ellipsoids of dimension 2). This entails that the map T^ε is not monotone. However, we can still formulate an iteration scheme *a la* Kleene in the present setting, defining

$$\begin{aligned} \underline{\mathcal{E}}^0 &= (\mathcal{E}_I, \dots, \mathcal{E}_I) \\ \underline{\mathcal{E}}^{k+1} &= T^\varepsilon(\underline{\mathcal{E}}^k) . \end{aligned} \tag{6}$$

We assume in the sequel that the set of initial states $\mathcal{E}_I$ and the set of controls $\mathcal{E}_{\mathcal{U}}$ are full-dimensional. If this is not the case, we may approximate these ellipsoids by "nearly flat" ellipsoids. Then, it is easily shown by induction that all occurrences of the operators $\sqcup$, $\sqcap$ and $\boxplus$ in the Kleene iteration can be computed with the means presented in Section 3:

Lemma 4.2. *Assume that the ellipsoids $\mathcal{E}_I$ and $\mathcal{E}_{\mathcal{U}}$ are full-dimen-sional. Then all the ellipsoids $\mathcal{E}_w^k$ computed at each step of the Kleene iteration in Equation* (6) *are full-dimensional.*

The lack of monotonicity of the operator T^ε, and the fact that the space of ellipsoids does not constitute a lattice, make more difficult the analysis of the Kleene iteration scheme than in the classical case of abstract analysis. In particular, we have to replace some order theoretical arguments by metric fixed point properties. We establish the convergence of the Kleene iteration in the *linear case* — i.e. when the assignments are linear ($B_a = 0$ and $c_a = 0$), the switching process is non-deterministic and the ellipsoids are centered — if the "stability margin" is sufficient.

Theorem 4.3. *Assume that the assignments are linear* ($B_a = 0$ *and* $c_a = 0$)*, the switching process is non-deterministic and that the ellipsoid $\mathcal{E}_I$ is centered and full-dimensional. Then there is a positive constant $\mu_{n,I}$ depending on the dimension n and the initial states $\mathcal{E}_I$ such that if the spectral norms of the matrices $(A_a)_{a\in\Sigma}$ are smaller than $\mu_{n,I}$, then the Kleene iteration $\underline{\mathcal{E}}^{k+1} = T^\varepsilon(\underline{\mathcal{E}}^k)$ converges.*

This theorem is proved in Appendix **??**. The bound $\mu_{n,I}$ that is given is very conservative. However, we shall see in Section 5.2 that our algorithm converges although the condition is not satisfied.

We also deduce that

$$O\Big(\frac{\log\varepsilon - \log\max_w \|Q_w^0 - Q_w^\infty\|_2}{3|\mathcal{W}|\log(\lambda_1/\lambda_0) + 2\log\max_a\|A_a\|_2}\,.\Big) \tag{7}$$

iterations suffice to compute an invariant. As shown in Appendix **??**, the values $\lambda_0 < \lambda_1$ depend only on the set of initial states and the matrices A_a.

Establishing the convergence of the Kleene iteration in Equation 6 in the general case is difficult problem and remains open.

Open Problem 1. *Does the iterative scheme in Equation 6 converge if the matrices A_a are suitably small, when either affine assignments* ($c_a \neq 0$)*, guards (switching is state-dependent) or non-constant controls are present* ($B_a \neq 0$) *?*

The "small-LMI" approach. When implementing the Kleene iteration scheme 6, a semi-definite program needs to be solved for each evaluation of the operator $\sqcup$ and $\boxplus$. The number of variables in each of these semi-definite programs only depends on the dimension n of the problem, not on the size $|\mathcal{W}|$ of the automaton, in contrast with alternative approaches detailed in Section 5.1, inspired by state of the art methods. For this reason, we call the method to compute invariants based on Theorem 4.1 and on Kleene iteration the "small-LMI" approach.

Note that Theorem 4.1 is also valid for the map T. As a consequence, we may be tempted to use this map rather than T^ε in the Kleene scheme. However, in practice, most operators are evaluated by solving semi-definite programs, which only return approximate optimal solutions. Introducing the parameter ε counters several hurdles that may be encountered and could endanger confidence in the final invariant. First, the parameter ε absorbs approximation errors that appear throughout the computation, and thus gives a margin of safety if computations are done using finite precision. Moreover, padding each inclusion constraint ensures that the set of feasible points has non-empty interior, so that the a posteriori numerical check presented in [**?**] can be used. Finally, if the parameter ε was not present, a fixed point would only be reached *ultimately*, i.e. after an infinite number of iterations. Now, a post-fixed point can be reached in a finite number of iterations.

The "no-LMI" approach. When the assignments in each branch are linear ($B_a = 0$ and $c_a = 0$) and when the switching condition is non-deterministic (meaning that the test is replaced by a random boolean), it is possible to get rid of LMIs altogether, still building on the same principles. Indeed, it was shown in [**?**] that the Löwner ellipsoid of the union of *two* centered ellipsoids can be computed from a Cholesky decomposition, avoiding the use of LMI, resulting in an important speed-up. We will exploit here the latter result, by relaxing the computation of $\sqcup_k\mathcal{E}_k$ to a sequential computation $\big((\mathcal{E}_1 \sqcup \mathcal{E}_2)\cdots \sqcup \mathcal{E}_w\big)$. The variant of the map T^ε obtained in this way can now be computed without solving semi-definite programs. In the sequel, we will refer to this variant of the present "small-LMI" approach as the "no-LMI" approach.

5 ALTERNATIVE APPROACHES AND BENCHMARKS

5.1 The "big-LMI" and "big-BMI" approaches

For comparison, we next present two alternative approaches, derived from earlier works [**? ?**], leading to larger LMI or to non-convex programs.

The first approach has been studied in [**?**]. It is restricted to the special case of *linear assignments* under *non-deterministic switching*, where the initial state has been approximated by a *centered ellipsoid*. In other words, it requires that $B_1 = B_2 = 0$, $c_1 = c_2 = 0$, the guard condition $f^T x \leqslant g$ has been replaced by a non-deterministic switching mechanism and $\mathcal{E}_I = \mathcal{E}(Q_I, 0)$. Then, the post-fixed point problem can be rewritten as a single LMI involving the whole collection of design variables $(Q_w)_{w\in\mathcal{W}}$:

$$\begin{aligned} Q_I &\preccurlyeq Q_w,\ \forall w \in \mathcal{W}\,,\\ A_a Q_v A_a^T &\preccurlyeq Q_w,\ \forall w, v, a \text{ such that } \tau(v,a) = w\,. \end{aligned} \tag{8}$$

The variables in this LMI are highly coupled among themselves. Indeed, the variable Q_w appears N times on the right-hand-side of an inequality of the form above, but also N times on the left-hand-side. Thus, unless the transition map τ is constant ($\tau(v,a) = v$ for all a), it is not possible to solve Equation (8) for each word w separately. This case does not appear in practice: the transition map used in Section 5.2 induces a maximal coupling, where (almost) each word is related to N other words.

If $(Q_w)_{w\in\mathcal{W}}$ is a solution of this LMI, then the union of the ellipsoids $\cup_{w\in\mathcal{W}}\mathcal{E}(Q_w,0)$ is an invariant for the associated program. It is in fact a variation on the method in [?], because the LMI $Q_w \succ 0$ has been replaced by $Q_w \succcurlyeq Q_I$. This only requires that the solution provided in [?] be scaled to contain the set of initial states.

The latter optimization problem has $|\mathcal{W}|n(n+1)/2$ independent variables. We say that this optimization problem is a "big LMI" because the number of variables depends on the size of the automaton. Finding an approximate solution via semi-definite programming thus has an arithmetic complexity of $\mathcal{O}(n^{6.5}|\mathcal{W}|^5)$, according to the formulae in Section 4.6.3 of [?]. In comparison, the semi-definite programs used to compute the operators $\sqcup$ and $\boxplus$ both have an arithmetic complexity of $\mathcal{O}(n^{6.5})$ that does not depend on $|\mathcal{W}|$. Each iteration of the Kleene iteration only needs to compute $\mathcal{O}(|\mathcal{W}|)$ of these, for a total arithmetic complexity of $\mathcal{O}(n^{6.5}|\mathcal{W}|)$ per iteration. As a consequence, when the automaton used in the computation of the map T^ε has many states, the "big-LMI" approach becomes intractable, contrary to the "small-LMI" method.

The "big-LMI" method may be thought of as dual of a Lyapunov-type approach, also detailed in [?]: the latter is equivalent to representing the unit ball of the Barabanov norm by an intersection of ellipsoid. Instead, the "big-LMI" method computes an invariant set given by a union of ellipsoids. Both methods lead to semi-definite programs of a comparable nature and size.

In the case of a single centered ellipsoid, it is still possible to use LMIs if we assume that B_1 or B_2 is non-zero (but not both). We can use a method akin to the one used in [?], where bisection is used in order to successively compute a value for λ in Equation (4).

However, the more general case of affine assignments under non-deterministic switching cannot be dealt with LMIs, since the stability problem then involves *several bilinear inequalities* in terms of the design variables. This can dealt with by solving a *bilinear matrix inequality* (BMI for short), which has the form

$$A_0 + \sum_{i=1}^{d} x_i A_i + \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j A_{i,j} \succcurlyeq 0\,. \tag{9}$$

For instance, when $B_1 = B_2 = 0$ and the switching is nondetermi-nistic, the post-fixed point problem can be rewritten

Program 2: A simple switched affine program

```
$x \leftarrow \mathcal{E}_I$;
while $true$ do
    $u \leftarrow \mathcal{E}_\mathcal{U}$;
    if $f^T x \leqslant g$ then
        $x := A_1 x + B_1 u + c_1$;
    else
        $x := A_2 x + B_2 u + c_2$;
    end
end
```

as a BMI in the variables $(L_w)_{w\in\mathcal{W}}$, $(\lambda_w)_{w\in\mathcal{W}}$ and $(\mu_{v,a})_{(v,a)\in W\times\Sigma}$:

$$\forall w,\ \exists\lambda_w\in\mathbb{R}\colon \begin{pmatrix} L_w L_w^T & q_0 - q_w & L_0 \\ (q_0-q_w)^T & 1-\lambda_w & 0_{1,n} \\ L_0^T & 0_{n,1} & \lambda_w I_n \end{pmatrix} \succcurlyeq 0\,,$$

$\forall v, a,\ \exists \mu_{v,a}\in\mathbb{R}\colon$

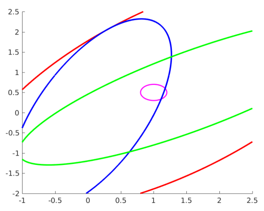
$$\begin{pmatrix} L_{\tau(v,a)} L_{\tau(v,a)}^T & A_a q_v + c_a - q_{\tau(v,a)} & L_v \\ (A_a q_v + c_a - q_{\tau(v,a)})^T & 1-\mu_{v,a} & 0_{1,n} \\ L_v^T & 0_{n,1} & \mu_{v,a} I_n \end{pmatrix} \succcurlyeq 0\,,$$

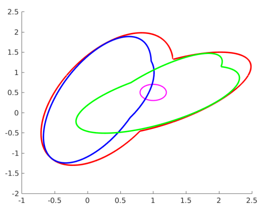
where $\mathcal{E}_I = \mathcal{E}(L_0 L_0^T, q_0)$. Any solution of this BMI yields as invariant the union of the ellipsoids $\mathcal{E}(L_w L_w^T, q_w)$. Unlike LMI, BMI have generally non-convex feasible sets, and therefore numerical solvers may return only locally optimal solutions. Despite the computational drawbacks of the "big-BMI" method, it is to our knowledge the only state of the art method that can deal with affine assignments with different equilibria.

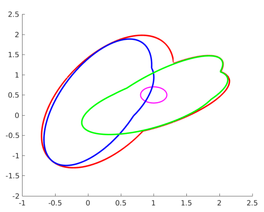
5.2 Benchmarks

We present in this section numerical benchmarks of our method. The experiments are implemented in Matlab, running on one core of an 2.2GHz Intel Core i7 with 8Gb RAM. We use the SDPT-3 solver [?], in conjunction with YALMIP [?] to solve LMIs, and the PENLAB solver [?] to solve BMIs. In all subsequent pictures, the initial state is shown in magenta and the disjunctive invariant $\mathcal{I} := \cup_{w\in\mathcal{W}}\mathcal{E}_w$ is shown in red. We show in blue (resp. green) the image of the invariant $\mathcal{I}$ by the abstract operators of the branch 1 (resp. 2), i.e. $\cup_{w\in\mathcal{W}} 1\cdot\mathcal{E}_w$ (resp. $\cup_{w\in\mathcal{W}} 2\cdot\mathcal{E}_w$), which prove an over-approximation of the reachable set in branch 1 (resp. 2). In all examples, it was sufficient to compute 30 iterations to obtain a post-fixed point, and thus an invariant.

Switched linear system with guards. We next show that automata $\mathcal{A}$ that "keep in memory" the m last switches that happened produce better invariants than other types of automata. Moreover, we demonstrate that the invariants that are produced are more accurate the more switches are "remembered". We instantiate the elements from Program 2 as follows: $\mathcal{E}_I = \mathcal{E}(0.04 I_2, \left(\begin{smallmatrix}1\\0.5\end{smallmatrix}\right))$, $\mathcal{U} = \mathcal{E}(0.1 I_2, 0)$, $A_1 = 0.5\left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)$, $B_1 = I_2$, $c_1 = 0$, $A_2 = 0.5\left(\begin{smallmatrix}1&0\\1&1\end{smallmatrix}\right)$, $B_2 = I_2$, $c_2 = 0$, $f = \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$, $g = 1$. Note that it involves an affine guard, so the analysis of


(a) De Bruijn automaton on Σ^0 (1 ellipsoid)


(b) De Bruijn automaton on Σ^1 (2 ellipsoids)


(c) De Bruijn automaton on Σ^2 (4 ellipsoids)

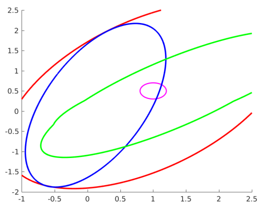
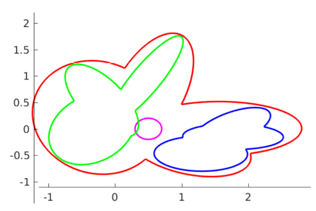

(d) Non-De Bruijn automaton with 3 states (3 ellipsoids)

Figure 1: Invariants (red) computed for a switched linear system w.r.t. the automaton, their image by the abstract operators (blue and green) and the initial state (magenta)

this example is out of reach of a "big-LMI", or "big-BMI",-type method.

We recall that the *De Bruijn automaton* on the set Σ^m is an automaton whose alphabet is Σ and whose states are precisely Σ^m. Its transition map τ deletes a word's first letter and appends the transition letter to its end. In other words, we have $\tau(v, a) = w$ if and only if $v = a_1a_2 \ldots a_m$ and $w = a_2 \ldots a_m a$, with $a_i \in \Sigma$. By construction, this automaton "remembers" the last m transitions. For this reason, we expect invariants computed using larger De Bruijn automata to be more precise. This has been verified experimentally and is shown in Figures **??**-**??**. We have also experimented with non-De Bruijn automata, as shown in Figure **??**. Notice that as the more switches are "remembered", the more concise the invariant becomes throughout Figure **??**. We also point out that the transition function for the automaton used in Figure **??** does not reflect a memory process. Although it performs slightly better that the De Bruijn automaton on Σ^0, using only one ellipsoid, the invariant computed with this automaton remains convex and thus less accurate than the previous ones.

Defocused switched affine systems. We demonstrate again the fact that the "small-LMI" method provides better invariants the more states the underlying automaton has. We consider a discretized version of Example 6.3 in [?], to which we have added a guard condition. Using a discretization step $\delta t = 0.5$, we instantiate Program 2 with $\mathcal{E}_I = \mathcal{E}(0.04I_2, \left(\begin{smallmatrix} 0.5 \\ 0 \end{smallmatrix}\right))$, $\mathcal{E}_{\mathcal{U}} = \mathcal{E}(0, 0)$, $A_1 = \left(\begin{smallmatrix} 0.68 & -0.75 \\ 0.19 & 0.68 \end{smallmatrix}\right)$, $B_1 = 0$, $c_1 = \left(\begin{smallmatrix} 0.5432 \\ -0.0724 \end{smallmatrix}\right)$, $A_2 = \left(\begin{smallmatrix} 0.72 & -0.39 \\ 0.39 & 0.72 \end{smallmatrix}\right)$, $B_2 = 0$, $c_2 = \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$, $f = \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right)$, $g = 0$. This


(a) De Bruijn automaton on Σ^2 (4 ellipsoids)

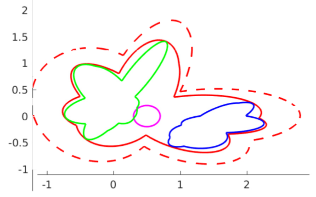

(b) De Bruijn automaton on Σ^4 (16 ellipsoids) – invariant with 4 ellipsoids in dashed

Figure 2: Invariants (red) computed for a defocused switched system w.r.t. the automaton, their image by the abstract operators (blue and green) and the initial state (magenta)

system has two distinct fixed points. Since it involves an affine guard, the analysis of this example is out of reach of a "big-LMI", or "big-BMI",-type method. We show in Figure **??** two invariants computed by using the De Bruijn automata on Σ^2 (4 states) and Σ^4 (16 states). The invariant computed with the latter automaton is strictly better than the one computed with the former.

Observer based controller for a coupled mass system [?] We show that our method can also be used to analyze systems with saturations. The addition of saturation simulates sensors that measure a physical quantity precisely within some range, but cannot measure values outside this range. We study the stability of an affine dynamical system subject to saturation conditions on the first coordinate of the state vector:

$$x_1^{k+1/2} = \begin{cases} \beta & \text{if } f^T x^k > \beta & (10a) \\ -\beta & \text{if } f^T x^k < -\beta & (10b) \\ x_1^k & \text{otherwise} & (10c) \end{cases}$$

$$x^{k+1} = A_i x^{k+1/2} + B_i u^{k+1/2} + c_i \ , \ i \in \mathcal{I} \,,$$

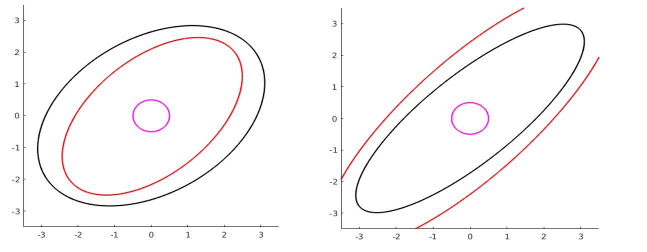
with a bounded control $u \in \mathcal{E}_{\mathcal{U}}$, first in a case without switching, and then in a case where switching occurs.

The semantics of a program implementing this system are the same of the program which resets the state vector to an initial value in the branches (**??**) and (**??**). In other words, we may choose $a \cdot \mathcal{E} := \mathcal{E}_I$ for all ellipsoid $\mathcal{E}$ (the same equation holds for b). Thus, the map T takes the usual form as in Equation (5) when the word w ends with c, and is written $T_w(\underline{\mathcal{E}}) = \mathcal{E}_I$ when w ends with a or b.

First, we demonstrate our method with $\mathcal{I} = \{1\}$, $c_1 = 0$, $f = \left(\begin{smallmatrix} 1 & 0 & 0 & 0 \end{smallmatrix}\right)^T$, $\beta = 0.5$,

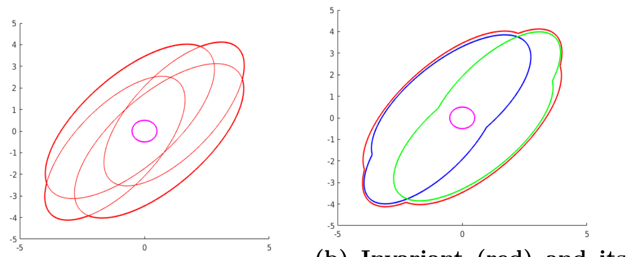
$$A_1 = \begin{pmatrix} 0.6227 & 0.3871 & -0.113 & 0.0102 \\ -0.3407 & 0.9103 & -0.3388 & 0.0649 \\ 0.0918 & -0.0265 & -0.7319 & 0.2669 \\ 0.2643 & -0.1298 & -0.9903 & 0.3331 \end{pmatrix} \text{ and } B_1 = \begin{pmatrix} 0.3064 & 0.1826 \\ -0.0054 & 0.6731 \\ 0.0494 & 1.6138 \\ 0.0531 & 0.4012 \end{pmatrix}.$$

We use the De Bruijn graph on Σ^2 (4 ellipsoids). Our algorithm converges towards some collection of matrices $\underline{\mathcal{E}}$, and the resulting ellipsoids satisfy $\mathcal{E} \subset \mathcal{E}_{aa}$ for all $\mathcal{E} \in \underline{\mathcal{E}}$, meaning



(a) Section in the (x_1, x_2) plane (b) Section in the (x_3, x_4) plane

Figure 3: Sections of the invariant ellipsoid (red) for the coupled mass system and the reference in [?] (black)



(a) The 4 ellipsoids representing the invariant (b) Invariant (red) and its image by the abstract operators (green and blue)

Figure 4: Section in the (x_1, x_2) plane of the invariant for a switching variant of the coupled mass system (4 ellipsoids)

that the invariant is a single ellipsoid. Sections of this ellipsoid, as well as sections of the ellipsoid obtained in [?] are depicted in Figure **??**.

We also demonstrate our algorithm on a variant of the system in [?], that combines a switching mechanism with a saturation constraint. More precisely, we shall use $\mathcal{I} = \{1, 2\}$, $A_1 = A_2$, $B_1 = B_2$, $c_1 = \begin{pmatrix} -1/2 & 0 & 0 & 0 \end{pmatrix}^T$ and $c_2 = \begin{pmatrix} 1/2 & 0 & 0 & 0 \end{pmatrix}^T$. The mode 1 is active if $x_1 > 0$ and the mode 2 is active otherwise. By design, this system has two fixed points (when $u = 0$). In this example, our method yields a non-convex invariant, whose section in the (x_1, x_2) plane is shown in Figure **??**.

Comparison in the centered case: big-LMI versus no-LMI. We compare the "big-LMI" and "no-LMI" methods numerically on systems switching between implementations of two damped harmonic oscillators $M_i\ddot{x} + C_i\dot{x} + K_i x = 0$, with a discretization time $\delta t = 0.1$. The matrices M_i, C_i, K_i are randomly generated positive definite matrices, for dimensions ranging from 2 to 30. In these examples, we have $B_1 = B_2 = 0$, $c_1 = c_2 = 0$ and the guard condition $f^T x \leqslant g$ has been replaced by a non-deterministic switching process. We use the De Bruijn automaton on Σ^3 in all computations, so the invariants that are computed are given as unions of 8 ellipsoids. The execution times for the "big-LMI" and "no-LMI" methods are shown in Table **??**. We point out that the "no-LMI" method outperforms the "big-LMI" method by several orders of magnitude. The fact that the time-complexity relative to the dimension of the matrices is much smaller for the "no-LMI" method is also apparent.

Dimension n	"big-LMI"	"no-LMI"	relative volume
5	0.4s	**0.2s**	1.24
10	0.8s	**0.3s**	1.12
15	5s	**0.4s**	1.15
20	22s	**0.7s**	-
25	2min	**1.0s**	-
30	6min	**1.4s**	-

Table 1: Invariant computation time of each method w.r.t. the dimension n of the matrices (8 ellipsoids)

Finally, we compare the relative accuracy of the "no-LMI" method with respect to the "big-LMI" approach by the *relative volume* of the computed invariants, defined by the n-th root of the ratio between the volume of the "no-LMI" invariant by the volume of the "big-LMI" invariant. We report a difference in relative volume no larger than 25% in Table **??**, where the volumes have been estimated by a Monte-Carlo approximation, up to dimension 15 (no results for higher dimensions, due to lack of precision of the Monte-Carlo approach).

Comparison in the uncentered case: big-BMI versus small-LMI. We compare the "big-BMI" and "small-LMI" methods numerically on systems switching between implementations of two damped harmonic oscillators with a non-deterministic control $M_i\ddot{x} + C_i\dot{x} + K_i x = u$, with a discretization time $\delta t = 0.1$ and the control u is bounded in a non-centered ellipsoid. The matrices M_i, C_i, K_i are randomly generated positive definite matrices, for dimensions ranging from 2 to 14. The switching process is non-deterministic. We have used the De Bruijn automaton on Σ^2 in all computations, so the invariants that are computed are given as unions of 4 ellipsoids. The execution times for the "big-BMI" and "small-LMI" methods are shown in Table **??**. Although the "big-BMI" is more efficient on lower dimensional examples, one can see that it is very time-costly for 14×14 matrices, as solving the BMI takes 2000 times longer than for 2×2 matrices. In contrast, the "small-LMI" method has a base time cost per iteration that only grows from 1s in dimension 2 to 4s in dimension 14. Finally, we compare the relative accuracy of the "small-LMI" method with respect to the "big-BMI" approach by the *relative volume* of the computed invariants. We report an difference in relative volume no larger than 8% in Table **??**, where the volumes have again been estimated by a Monte-Carlo approximation.

6 CONCLUDING REMARKS

We introduced a new method to compute invariants of numerical programs, taking switched affine affine systems with guards as a target application. This relies on two ingredients: developing abstract interpretation techniques in a domain

Dimension n	"big-BMI"	"small-LMI"	relative volume
2	**3.5s**	24s	1.08
4	**9.2s**	26s	1.04
6	**34s**	36.3s	1.03
8	2.8min	**42.5s**	1.03
10	9.5min	**1.2min**	1.03
12	20.5min	**1.5min**	1.02
14	2.1h	**2.1min**	1.05
16	> 3h	**3.8min**	1.04

Table 2: Invariant computation time of each method w.r.t. the dimension n of the matrices (4 ellipsoids)

(unions of ellipsoids) which is not a lattice, and allowing the invariant to depend on a finite execution trace, to improve the accuracy. It is the combination of these two ideas which leads to an accurate and scalable method. In particular, a key feature of the method is the replacement of the solution of large scale LMI which would be obtained by earlier methods by an iterative scheme, combine Kleene iteration and smaller LMI solving. This iterative scheme is based on successive approximations by Löwner ellipsoids. This has also an advantage in terms of the increase of expressivity: we can solve instances with affine guards, for which there is no natural global ("big-LMI") formulation. In the special case in which the big-LMI approach is feasible, the no-LMI variant of our approach is also feasible, and it yields a speed-up by an order of magnitude, at the price of a potential additional loss of accuracy ("relaxation gap") induced by the approximation by Löwner ellipsoids.

Let us finally mention directions for future research. First, we would like to quantify a priori the latter "relaxation gap". Next, it would be interesting to find more explicit quantitative conditions ensuring the convergence of the iterative scheme and estimates on the convergence rate, in the general case. We would like to experiment the known methods for generalizing our invariant computation to deal with general programs: as such, there is no difficulty in extending the work to deal with nested or sequentially composed loops; for polynomial assignments, we could also use linearization methods such as [?]. Finally, we would like to generalize this method to other abstract domains, such as polyhedral templates or intersections of ellipsoids with half-spaces.

7 ACKNOWLEDGEMENTS

We thank all the reviewers for their comments, which have led us to make a number of clarifications and changes.

A APPENDIX

A.1 Proof of Proposition 3.2

Let $\mathcal{E}$ denote an ellipsoid containing $\mathcal{X}$. We denote the invertible affine map f by $x \mapsto Ax+b$. Since the map f is invertible, we have $\mathcal{E} \supseteq f(\mathcal{X}) \iff f^{-1}(\mathcal{E}) \supseteq \mathcal{X}$. Moreover, the volume of the ellipsoid $\mathcal{E}$ is equal to the volume of the ellipsoid $f^{-1}(\mathcal{E})$ multiplied by $\det A$. Combined with the uniqueness of the Löwner ellipsoid, we deduce that $\mathcal{E} = \text{Löw}\big(f(\mathcal{X})\big)$ if and only if $f^{-1}(\mathcal{E}) = \text{Löw}(\mathcal{X})$.

A.2 Proof of Theorem 4.1

By definition, the set $\mathcal{I} := \cup_{w\in\mathcal{W}}\mathcal{E}_w$ is an invariant if and only if $\mathcal{E}_I \subset \mathcal{I}$ and $\bigcup_{a\in\Sigma}\Big[f_a\big(\mathcal{I}\cap\cap_j\mathcal{H}_{a,j}\big) + \big(B_a\mathcal{E}_{\mathcal{U}}\big)\Big] \subseteq \mathcal{I}$. Hence, we must show that

$$\bigcup_{a\in\Sigma}\bigcup_{v\in\mathcal{W}}\Big[f_a\big(\mathcal{E}_v\cap\cap_j\mathcal{H}_{a,j}\big) + \big(B_a\mathcal{E}_{\mathcal{U}}\big)\Big] \subseteq \bigcup_{w\in\mathcal{W}}\mathcal{E}_w\,.$$

Since the abstract operations $\sqcup$, $\sqcap$ and $\boxplus$ over-approximate their concrete counterpart $\cup$, $\cap$ and $+$, we have for all $a\in\Sigma$ and $v\in\mathcal{W}$

$$f_a\big(\mathcal{E}_v\cap\cap_j\mathcal{H}_{a,j}\big) + \big(B_a\mathcal{E}_{\mathcal{U}}\big) \subseteq f_a\circ\text{guard}_a(\mathcal{E}_v)\boxplus\big(B_a\mathcal{E}_{\mathcal{U}}\big)\,.$$

Let $w\in\mathcal{W}$. We have the inclusion

$$\bigcup_{\tau(v,a)=w}\Big[f_a\big(\mathcal{E}_v\cap\cap_j\mathcal{H}_{a,j}\big) + \big(B_a\mathcal{E}_{\mathcal{U}}\big)\Big] \subseteq \mathcal{E}_I \sqcup \bigsqcup_{\tau(v,a)=w}\Big[f_a\circ\text{guard}_a(\mathcal{E}_v)\boxplus\big(B_a\mathcal{E}_{\mathcal{U}}\big)\Big]\,,$$

where we have again used the same abuse of notation as in Equation (5). On the right-hand side, we recognize the w-th coordinate of $T(\underline{\mathcal{E}})$. Recall that $T^{\varepsilon}_w(\underline{\mathcal{E}})$ is defined as $\mathcal{E}(Q_w+\varepsilon I_n, q_w)$ when $T_w(\underline{\mathcal{E}}) = \mathcal{E}(Q_w, q_w)$. Since the ellipsoids $T^{\varepsilon}_w(\underline{\mathcal{E}})$ and $T_w(\underline{\mathcal{E}})$ have the same center, and $Q_w+\varepsilon I_n \succcurlyeq Q_w$, we deduce that $T_w(\underline{\mathcal{E}}) \subseteq T^{\varepsilon}_w(\underline{\mathcal{E}})$. Finally, since $\underline{\mathcal{E}}$ is a fixed point of the map T^{ε}, we have $T^{\varepsilon}_w(\underline{\mathcal{E}}) = \mathcal{E}_w$ and

$$\bigcup_{a\in\Sigma}\bigcup_{v\in\mathcal{W}}\Big[f_a\big(\mathcal{E}_v\cap\cap_j\mathcal{H}_{a,j}\big) + \big(B_a\mathcal{E}_{\mathcal{U}}\big)\Big] \subseteq \bigcup_{w\in\mathcal{W}}E_w\,.$$

Moreover, we have $\mathcal{E}_I \subseteq T_w(\underline{\mathcal{E}})$ for all $w\in\mathcal{W}$. We deduce from the same arguments as before that $\mathcal{E}_I \subseteq \cup_{w\in\mathcal{W}}\mathcal{E}_w$.

A.3 Proof of Theorem 4.3

Since all ellipsoids are centered, we may abuse the notation and write $Q_1\sqcup Q_2$ instead of $\mathcal{E}(Q_1,0)\sqcup\mathcal{E}(Q_2,0)$, and $T^{\varepsilon}_w(\underline{Q})$ instead of $T^{\varepsilon}_w(\underline{\mathcal{E}})$. We denote $\mathcal{E}_I = \mathcal{E}(Q_I, 0)$.

The map T^{ε} is written

$$T^{\varepsilon}_w(\underline{Q}) = Q_I \sqcup \bigsqcup_{\tau(v,a)=w}(A_aQ_vA_a^T + \varepsilon I_n)\,.$$

First, we show that there are positive reals $\lambda_0 < \lambda_1$ such that the set of X such that for all $\underline{Q}\in(\mathcal{S}_n^{++})^{\mathcal{W}}$,

$$\Big(\forall w.\ \lambda_0 I_n \preccurlyeq Q_w \preccurlyeq \lambda_1 I_n\Big) \implies \Big(\forall w.\ \lambda_0 I_n \preccurlyeq T^{\varepsilon}_w(\underline{Q}) \preccurlyeq \lambda_1 I_n\Big)\,.$$

The map T^{ε}_w is bounded below by Q_I, which is positive definite, so there is $\lambda_0 > 0$ such that $T^{\varepsilon}_w(\underline{Q}) \succcurlyeq \lambda_0 I_n$ for all X. Moreover, one can deduce from the explicit formulation of $\sqcup$ in [?] that $\sqcup_k Q_k \preccurlyeq \sum_k Q_k$ if $\sqcup$ is evaluated sequentially. Thus we have Assuming that the spectral norm (largest singular value denoted by $\|\cdot\|$) of the matrices A_i are strictly less than $N^{-1/2}$, then the desired property is satisfied for $\lambda_1 := \varepsilon + \|Q_I\|_2(1-\sum_a\|A_a\|)^{-1}$.

Moreover, there is a natural metric on the space of positive definite matrices given by the *Riemann metric* [?]:

$$d_R(X,Y) = \Big[\sum_k \big(\log \lambda_k\big)^2\Big]^{1/2},$$

where λ_k are the (real) eigenvalues of the matrix $X^{-1}Y$.

The set $\mathcal{K} \coloneqq \{X \in \mathcal{S}_n^{++} \colon \lambda_0 I_n \preccurlyeq X \preccurlyeq \lambda_1 I_n\}$ is bounded in Riemann's metric (its diameter is less than $n \log(\lambda_1 \lambda_0^{-1})$).

This metric is a *Finsler metric* [?], meaning that it behaves like a norm in a local setting: we have $d_R(X,Y) \sim \|Z^{-1}(X - Y)\|_2$ when $X, Y \to Z$, and $\|\cdot\|_2$ is the euclidean norm on $\mathcal{S}_n$. One can obtain the very coarse bound

$$(\lambda_1/\lambda_0^2)^{-1}\|X-Y\|_2 \leqslant d_R(X,Y) \leqslant (\lambda_1^2/\lambda_0)\|X-Y\|_2 \,.$$

We also deduce from its Finsler nature and the theory of Schur multipliers [?] that the $\sqcup$ operator is non-expansive with respect to the Riemann metric (the details of the proof will be given elsewhere). Hence the $\sqcup$ operator must be Lipschitz with respect to the euclidean norm on the set $\mathcal{K}$, with a Lipschitz constant no larger than $(\lambda_1/\lambda_0)^3$.

Finally, the Riemann metric has a remarkable property, which is convenient in the present analysis. It is invariant by a congruence by an invertible matrix P: $d_R(PXP^T, PYP^T) = d_R(X,Y)$. Combining these results, we deduce, for $\underline{Q}, \underline{Q}' \in (\mathcal{S}_n^{++})^{\mathcal{W}}$ and $\alpha \coloneqq \max_a \|A_a\|_2^2$,

$$\|T_w^\varepsilon(\underline{Q}) - T_w^\varepsilon(\underline{Q}')\|_2 \leqslant \alpha\big(\frac{\lambda_1}{\lambda_0}\big)^{3|\mathcal{W}|} \max_{\tau(v,a)=w} \|Q_v - Q'_v\|_2 \,.$$

Hence, the Kleene iteration converges if the spectral norms of the matrices A_a are small enough.

It is sufficient that $\|T_w^\varepsilon(\underline{Q}) - T_w^\varepsilon(\underline{Q}')\|_2 \leqslant \varepsilon$ to obtain an invariant, thus we deduce the number of iterations given in Equation (7).

REFERENCES

[] A. Adjé and P.-L. Garoche. 2015. Automatic Synthesis of Piecewise Linear Quadratic Invariants for Programs. In *Proceedings of VMCAI*. 99–116.

[] A. Adjé, S. Gaubert, and E. Goubault. 2010. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In *(ESOP 2010)*. Number 6012 in Lecture Notes in Computer Science. Springer, 23–42.

[] A. A. Ahmadi, R. M. Jungers, P. A. Parrilo, and M. Roozbehani. 2014. Joint Spectral Radius and Path-Complete Graph Lyapunov Functions. *SIAM J. Control and Optimization* 52, 1 (2014), 687–717.

[] X. Allamigeon, S. Gaubert, E. Goubault, S. Putot, and N. Stott. 2016. A Scalable Algebraic Method to Infer Quadratic Invariants of Switched Systems. *ACM Trans. Embedded Comput. Syst.* 15, 4 (2016), 69:1–69:20.

[] Koenraad M.R. Audenaert. 2013. Schur multiplier norms for Loewner matrices. *Linear Algebra Appl.* 439, 9 (2013), 2598 – 2608.

[] A. Ben-Tal and A. S. Nemirovskiaei. 2001. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[] Robert G. Bland, Donald Goldfarb, and Michael J. Todd. 1981. The Ellipsoid Method: A Survey. *Operations Research* 29, 6 (1981), 1039–1091.

[] A. D. Blondel and J. N. Tsitsiklis. 2000. A survey of computational complexity results in systems and control. *Automatica* 36 (2000), 1249–1274.

[] Silvére Bonnabel and Rodolphe Sepulchre. 2010. Riemannian Metric and Geometric Mean for Positive Semidefinite Matrices of Fixed Rank. *SIAM J. Matrix Anal. Appl.* 31, 3 (2010), 1055–1070.

[] F. Bourdoncle. 1992. Abstract Interpretation by Dynamic Partitioning. *J. Funct. Program.* 2, 4 (1992), 407–423.

[] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. 1994. *Linear Matrix Inequalities in System and Control Theory.* Studies in Applied Mathematics, Vol. 15. SIAM, Philadelphia, PA.

[] M. S. Branicky. 1998. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. Automat. Control* 43, 4 (Apr 1998), 475–482. DOI:https://doi.org/10.1109/9.664150

[] H. Busemann. 1950. The Foundations of Minkowskian Geometry. *Commentarii mathematici Helvetici* 24 (1950), 156–187. http://eudml.org/doc/139004

[] P. Cousot. 2005. Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming. In *VMCAI 2005, Paris, France, January 17-19, 2005, Proceedings.* 1–24.

[] P. Cousot and R. Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '77).*

[] P. Cousot and N. Halbwachs. 1978. Automatic Discovery of Linear Restraints Among Variables of a Program. In *Proceedings of POPL'78.* ACM, 84–96.

[] E. de Klerk and F. Vallentin. 2016. On the Turing model complexity of interior point methods for semidefinite programming. *SIAM J. Optim.* 26, 3 (2016), 1944–1961.

[] A. Deutsch. 1990. On Determining Lifetime and Aliasing of Dynamically Allocated Data in Higher-Order Functional Specifications. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California, USA, January 1990.* 157–168.

[] J. Feret. 2004. Static Analysis of Digital Filters. In *Proceedings of ESOP'04.* 33–48.

[] E. Feron and F. Alegre. 2008. Control software analysis, Part I Open-loop properties. *CoRR* abs/0809.4812 (2008).

[] T. Martin Gawlitza, H. Seidl, A. Adjé, S. Gaubert, and E. Goubault. 2012. Abstract interpretation meets convex optimization. *J. Symb. Comput.* 47, 12 (2012), 1416–1446.

[] R. Giacobazzi and F. Ranzato. 1998. Optimal domains for disjunctive abstract interpretation. *Science of Computer Programming* 32, 1-3 (1998), 177 – 210. 6th European Symposium on Programming.

[] E. Goubault and S. Putot. 2009. A zonotopic framework for functional abstractions. *CoRR* abs/0910.1763 (2009).

[] M. Stingl J. Fiala, M. Kočvara. 2013. PENLAB: A MATLAB solver for nonlinear semidefinite optimization. (2013).

[] B. Jeannet, N. Halbwachs, and P. Raymond. 1999. Dynamic Partitioning in Analyses of Numerical Properties. In *Static Analysis, 6th International Symposium, SAS '99, Venice, Italy, September 22-24, 1999, Proceedings.* 39–50.

[] R. V. Kadison. 1951. Order Properties of Bounded Self-Adjoint Operators. *Proc. Amer. Math. Soc.* 2, 3 (1951), 505–510. http://www.jstor.org/stable/2031784

[] J. Löfberg. 2004. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *In Proceedings of the CACSD Conference.* Taipei, Taiwan.

[] M. Martel. 2003. Improving the Static Analysis of Loops by Dynamic Partitioning Techniques. In *(SCAM 2003), 26-27 September 2003, Amsterdam, The Netherlands.* 13–21.

[] L. Mauborgne and X. Rival. 2005. Trace Partitioning in Abstract Interpretation Based Static Analyzers. In *European Symposium on Programming (ESOP'05) (Lecture Notes in Computer Science)*, M. Sagiv (Ed.), Vol. 3444. Springer-Verlag, 5–20.

[] A. Miné. 2004. *Weakly Relational Numerical Abstract Domains.* Ph.D. Dissertation. École Polytechnique, Palaiseau, France.

[] A. Miné. 2006. Symbolic Methods to Enhance the Precision of Numerical Abstract Domains. In *VMCAI 2006, Charleston, SC, USA, January 8-10, 2006, Proceedings.* 348–363.

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

[] P. Nilsson, U. Boscain, M. Sigalotti, and J. Newling. 2013. Invariant sets of defocused switched systems. In *Conference of Decision and Control.*

[] M. Oulamara and A. J. Venet. 2015. *CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*. Springer International Publishing, Cham, Chapter Abstract Interpretation with Higher-Dimensional Ellipsoids and Conic Extrapolation, 415–430.

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

[] P. Roux and P.-L. Garoche. 2013. Integrating Policy Iterations in Abstract Interpreters. In *ATVA (Lecture Notes in Computer Science)*, D. Van Hung and M. Ogawa (Eds.), Vol. 8172. Springer, 240–254.

[] P. Roux, R. Jobredeaux, P.-L. Garoche, and E. Feron. 2012. A generic ellipsoid abstract domain for linear time invariant systems. In *Proceedings of HSCC*. 105–114.

[] Pierre Roux, Yuen-Lam Voronin, and Sriram Sankaranarayanan. 2016. *Validating Numerical Semidefinite Programming Solvers for Polynomial Invariants*. Springer Berlin Heidelberg, Berlin, Heidelberg, 424–446.

[] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. 2005. Scalable Analysis of Linear Systems Using Mathematical Programming. In *The Sixth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05) (LNCS)*, Vol. 3385. 25–41.

[] P. Sotin, B. Jeannet, F. Védrine, and E. Goubault. 2011. *Policy Iteration within Logico-Numerical Abstract Domains*. 290–305.

[] R. H. Tütüncü, K. C. Toh, and M. J. Todd. 2003. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming* 95, 2 (2003), 189–217.

[] A. Venet. 1996. Abstract Cofibered Domains: Application to the Alias Analysis of Untyped Programs. In *Static Analysis, Third International Symposium, SAS'96, Aachen, Germany, September 24-26, 1996, Proceedings*. 366–382.

[] A. Venet. 2002. Nonuniform Alias Analysis of Recursive Data Structures and Arrays. In *Static Analysis, 9th International Symposium, SAS 2002, Madrid, Spain, September 17-20, 2002, Proceedings*. 36–51.