Invertible Linear Transforms of Numerical Abstract Domains

Francesco Ranzato[0000−0003−0159−0068] and Marco Zanella

Dipartimento di Matematica, University of Padova, Italy

Abstract. We study systematic changes of numerical domains in abstract interpretation through invertible linear transforms of the Euclidean vector space, namely, through invertible real square matrices. We provide a full generalization, including abstract transfer functions, of the parallelotopes abstract domain, which turns out to be an instantiation of an invertible linear transform to the interval abstraction. Given an invertible square matrix M and a numerical abstraction A, we show that for a linear program P (i.e., using linear assignments and linear tests only), the analysis using the linearly transformed domain $M(A)$ can be obtained by analysing on the original domain A a linearly transformed program P^M. We also investigate completeness of abstract domains for invertible linear transforms. In particular, we show that, perhaps counterintuitively, octagons are not complete for 45 degrees rotations and, additionally, cannot be derived as a complete refinement of intervals for some family of invertible linear transforms.

1 Introduction

In abstract interpretation [6,7], the choice of an abstract domain determines which program properties will be analysed as well as the precision and efficiency of the corresponding program analysis. A vast array of abstract domains for analysing properties of numerical program variables is available as well as a number of operators for their combination, refinement and transformation which have been defined since the beginning of abstract interpretation [6,7,9,10,11,12] — see [18] for a recent and comprehensive tutorial on numerical abstract domains. The abstract domain of parallelotopes has been introduced and studied in [1,2,3,4] as a linear transform of the standard interval abstract domain [6]. Any invertible $n \times n$ real matrix M defines a domain of M-parallelotopes which consists of (vectors of) intervals $\langle [\mathbf{l}_i, \mathbf{u}_i] \rangle_{i=1}^n$, for $\mathbf{l}, \mathbf{u} \in (\mathbb{R} \cup \{\pm\infty\})^n$, whose concrete meaning is recast as the set of vectors $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{l} \leq M\mathbf{x} \leq \mathbf{u}$. The basic idea is that the matrix M represents a change of basis of the Euclidean vector space $\mathbb{R}^n$, which can be always converted back through its inverse matrix M^{-1}. Hence, $\langle [\mathbf{l}_i, \mathbf{u}_i] \rangle_{i=1}^n$ is a symbolic representation of the vectors $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{l} \leq M\mathbf{x} \leq \mathbf{u}\}$ in the new coordinate system based on M, which is therefore its concretization for the parallelotopes domain.

Parallelotopes can be used in program analysis in two different ways. In the first approach described in [1,2], the matrix M is fixed and is purposely synthesized for a program P through some statistical inference of the data gathered by a dynamic analysis of P, typically a variation of principal component analysis. On the other hand, [3,4] put

forward a program analysis where the abstract values are pairs consisting of an interval together with a matrix M, so that here the matrix is not computed a priori but rather the abstract transfer functions may change it during program analysis (as happens for convex polyhedra).

We study here a generalization of the first approach to the abstract domain of parallelotopes. An invertible square matrix M can be applied for systematically transforming any numerical abstract domain $\mathcal{A}$ together with all its abstract transfer functions. This is called an invertible linear transform of $\mathcal{A}$ and denoted by $\mathcal{A}^M$. This linear transform M preserves the whole structure of the abstract domain $\mathcal{A}$, meaning that if $\mathcal{A}$ is defined by a Galois connection/insertion then this also holds for $\mathcal{A}^M$, although this M-transform may also preserve domains defined through a concretization map only. Furthermore, it turns out that M systematically transforms the abstract transfer functions available in $\mathcal{A}$. More precisely, for the standard abstract transfer functions and operators used in abstract interpretation, namely, lub and glb, (single, parallel and backward) assignment, Boolean test, widening and narrowing, we provide a simple technique for designing the abstract transfer functions in the transformed domain $\mathcal{A}^M$ in terms of the abstract transfer functions in the original domain $\mathcal{A}$. Moreover, this transform of abstract functions preserves all their significant properties: soundness, best correct approximation, completeness and exactness. As a consequence, we show that an analysis with the transformed abstraction $\mathcal{A}^M$ of a program P consisting of linear assignments and tests can be obtained by analysing with the original abstraction $\mathcal{A}$ a transformed program P^M which is obtained from P by transforming all its linear assignments and tests while maintaining the same control flow graph. It should be remarked that this program change may transform single linear assignments of P into parallel linear assignments in P^M. If the analysis in $\mathcal{A}$ of the transformed program P^M relies on abstract transfer functions which are best correct approximations then this technique computes at each program point of P^M precisely the best abstract value for $\mathcal{A}^M$ at the same program point of P. This technique is illustrated through a couple of examples different from parallelotopes, namely linear transforms of constant propagation and octagon analysis.

As an example, a linear transform of Kildall's [15] standard constant propagation domain Const for three program variables through the invertible matrix $M = \left(\begin{smallmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{smallmatrix}\right)$ results in a transformed domain Const^M which is able to represent program invariants of type $x_1 = k_1$, $x_1 + x_2 = k_2$, $x_1 + x_3 = k_3$, where x_i's are variables and k_i's range in Const and therefore represent either a constant value or unreachability or no information. For instance, an analysis based on Const^M of the following program:

$x_1 := 2;\ x_2 := 3;\ x_3 := 6;$
while $(x_2 < x_3)$ **do**
$\{\ x_1 := x_1 - 2;\ x_3 := x_1 + x_2 + x_3 - 1;\ x_2 := x_2 + 2;\ \}$

is able to compute the abstract loop invariant $\langle \top, 5, 8 \rangle$ meaning that the additions $x_1 + x_2$ and $x_1 + x_3$ are always equal to, respectively, 5 and 8.

We also investigate completeness and exactness [7,13] of abstract domains for invertible linear transforms. Firstly, we show that a linearly transformed domain $\mathcal{A}^M$ is useless — meaning that it is equivalent to $\mathcal{A}$ itself — precisely when $\mathcal{A}$ is both complete and exact for the linear transform M. In particular, as expected, it turns out that

any linear transform of Karr's [14], templates [19] and convex polyhedra [8] abstract domains is ineffective. Instead, we prove that a linear transform M of intervals and octagons [17] is useless exactly when M is a monomial matrix, namely each row and column of M has exactly one nonzero entry. This characterization is expected since monomial matrices intuitively encode nonrelational linear transforms. Finally, we show that octagons cannot be obtained from intervals as the minimal refinement which is complete for some family of invertible linear transforms (this is called complete shell in [13]). This is somehow against the graphical intuition that octagons are complete for rotations of $\frac{\pi}{4}$ radians and therefore could be designed through a complete shell of intervals for this family of rotations. Rather, this intuition holds just in 2D, namely for two variables only. What we instead prove is that octagons can be synthesized through a suitable reduced product of $\frac{\pi}{4}$ rotations of intervals.

Due to lack of space all the proofs are omitted.

2 Background

Linear Transformations. We denote by $\overline{\mathbb{R}}$ the set of real numbers $\mathbb{R}$ augmented with $+\infty$ and $-\infty$, where ordering and numeric operations are extended from $\mathbb{R}$ to $\overline{\mathbb{R}}$ in the standard way. Vectors $\mathbf{x} \in \mathbb{R}^n$ (or $\mathbf{x} \in \overline{\mathbb{R}}^n$) are usually intended as column vectors, while $\mathbf{x}^T$ denotes the corresponding (transpose) row vector and $\mathbf{x}_i \in \mathbb{R}$, with $i \in [1, n]$, denotes its i-th component. If $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $a \in \mathbb{R}$ then $\mathbf{x} \cdot \mathbf{y}$, $\mathbf{x} + \mathbf{y}$ and $a\mathbf{x}$ denote, respectively, scalar product, addition of vectors and scalar multiplication in $\mathbb{R}^n$. The canonical orthonormal basis of $\mathbb{R}^n$ is denoted by $\langle \mathbf{e_1}, ..., \mathbf{e_n} \rangle$, where $\mathbf{e_{i}}_i = 1$ and, for any $j \neq i$, $\mathbf{e_{i}}_j = 0$. $\mathbb{R}^{m \times n}$ denotes the set of all $m \times n$ matrices with entries in $\mathbb{R}$, while $\mathrm{GL}(n)$ denotes the general linear group of $n \times n$ invertible square matrices with entries in $\mathbb{R}$. $\mathbf{0}_n \in \mathbb{R}^{n \times n}$ denotes the square zero matrix, $I_n \in \mathrm{GL}(n)$ denotes the identity matrix and A^{-1} and A^T denote, respectively, the inverse and transpose of A. A $1 \times n$ matrix is also used as a row vector, while a $n \times 1$ matrix as a column vector. A linear transformation of the n-dimensional Euclidean space $\mathbb{R}^n$ is a function in $\mathbb{R}^n \to \mathbb{R}^n$ of the form $\mathbf{x} \mapsto M\mathbf{x}$, where $M \in \mathbb{R}^{n \times n}$, which is simply denoted by $M : \mathbb{R}^n \to \mathbb{R}^n$. Given any set $X \in \wp(\mathbb{R}^n)$, we use the notation $M \cdot X \triangleq \{M\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \in X\}$ to denote the pointwise extension of M, and we also use $T_M : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ to denote the corresponding function on sets of vectors. Noteworthy examples of linear transformations include scalings, rotations, shearings and projections. Linear transformations M are partitioned between noninvertible and invertible: for example, (orthogonal or oblique) projections are noninvertible while rotations are always invertible. The set of invertible linear transformations of $\mathbb{R}^n$ endowed with function composition forms the well-known (noncommutative) general linear group $\mathrm{GL}(n)$. Let us also recall that $M \cdot X \subseteq Y \Leftrightarrow X \subseteq M^{-1} \cdot Y$ always holds for any $M \in \mathrm{GL}(n)$.

An affine transformation of $\mathbb{R}^n$ is a composition of a linear transformation with a transalation, i.e., it is a function in $\mathbb{R}^n \to \mathbb{R}^n$ of the form $\mathbf{x} \mapsto N\mathbf{x} + \mathbf{t}$, where $N \in \mathbb{R}^{n \times n}$ and $\mathbf{t} \in \mathbb{R}^n$. A pure translation $\mathrm{Tr}_{\mathbf{t}}(\mathbf{x}) \triangleq \mathbf{x} + \mathbf{t}$, for some vector $\mathbf{t} \in \mathbb{R}^n$, is the simplest example of (invertible) affine transformation.

Notable Linear Transformations. A *scaling* by a vector $\mathbf{s} \in \mathbb{R}^n$ is the linear transformation $\mathbf{x} \mapsto D^{\mathbf{s}}\mathbf{x}$, where $D^{\mathbf{s}} \in \mathbb{R}^{n \times n}$ is the diagonal matrix defined by $(D^{\mathbf{s}})_{ii} \triangleq \mathbf{s}_i$ and for $i \neq j$, $(D^{\mathbf{s}})_{ij} \triangleq 0$. A scaling transform is invertible iff for any i, $\mathbf{s}_i \neq 0$.

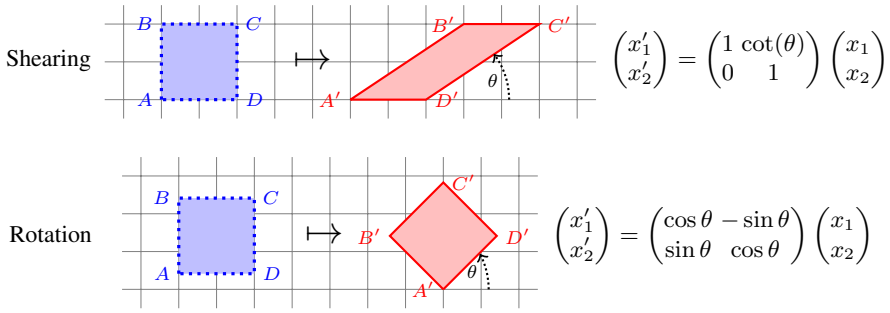

Fig. 1: An example of shearing and rotation transforms for two variables.

Let $n \geq 2$. Given some $\lambda \in \mathbb{R}$ and $a, b \in [1, n]$ with $a \neq b$, the invertible shear matrix $Sh^{a,b,\lambda} \in \mathrm{GL}(n)$ is defined as follows: $(Sh^{a,b,\lambda})_{ii} = 1$, $(Sh^{a,b,\lambda})_{ab} = \lambda$, otherwise $(Sh^{a,b,\lambda})_{ij} = 0$. This defines an invertible linear transformation called *shearing* (often used in computer graphics) which preserves the area of geometric figures and the alignment and relative distances of collinear points (a 2D example is in Fig. 1). The inverse of $Sh^{a,b,\lambda}$ is simply the shearing $Sh^{a,b,-\lambda}$ and, in general, shearings are not closed w.r.t. composition and their composition is not commutative.

A *Givens rotation* (or principal rotation) is the linear transformation which maps $\mathbf{x} \in \mathbb{R}^n$ into the point $\mathbf{x}' \in \mathbb{R}^n$ obtained by rotating $\mathbf{x}$ counterclockwise in a (a, b) plane of $\mathbb{R}^n$ (i.e., generated by $\mathbf{e_a}$ and $\mathbf{e_a}$), where $a, b \in [1, n]$ with $a \neq b$, by an angle of $\theta \in \mathbb{R}$ radians around the origin (a 2D example is in Fig. 1). This transformation is represented by an invertible Givens rotation matrix $R^{a,b,\theta} \in \mathrm{GL}(n)$ which is defined as follows: $R^{a,b,\theta}$ differs from the identity matrix I_n in the four entries (a, a), (a, b), (b, a), (b, b), where it assumes, respectively, the values $\cos\theta$, $-\sin\theta$, $\sin\theta$, $\cos\theta$. Clearly, $R^{a,b,-\theta}$ is the inverse of $R^{a,b,\theta}$. Givens rotations are closed by composition and When the rotation angle is $\theta = (2\pi)/m$ for some $m \in \mathbb{N} \setminus \{0\}$, it turns out that $R^{a,b,\theta}$ is cyclic, namely $(R^{a,b,\theta})^k = I_n$ for some integer $k > 0$.

Numerical Abstract Domains. According to the most general definition, a numerical abstract domain is a tuple $\langle A, \leq, \gamma \rangle$ where $\langle A, \leq \rangle$ is at least a preordered set and the concretization function $\gamma : A \to \wp(\mathbb{R}^n)$, where $n \geq 1$, preserves the relation $\leq$, i.e., $a \leq a'$ implies $\gamma(a) \subseteq \gamma(a')$. Thus, A plays the usual role of set of symbolic representations for sets of vectors of $\mathbb{R}^n$. If the base field of real numbers $\mathbb{R}$ is replaced by the field of rationals $\mathbb{Q}$, which is a possible choice for an abstract interpretation framework (see [18]), then completeness of the lattice $\langle \mathbb{R}, \leq \rangle$ is lost (i.e., $\langle \mathbb{Q}, \leq \rangle$ is not a complete lattice) so that some linear transformations cannot be taken into account, e.g., a Givens rotation of $\pi/4$. Also, linear transformations preserving integer vectors in $\mathbb{Z}^n$ (the n-dimensional integer lattice) have a narrow scope (they are studied in lattice geometry) and are not considered here (see [2, Section 7.5] for a discussion). Well-known examples of numerical abstract domains include signs, constants, intervals, affine equalities, zones, pentagons, octagons, parallelotopes, templates, convex polyhedra (the interested reader is referred to the recent tutorial [18]). Some numerical

domains just form preorders (e.g., standard representations of octagons by DBMs allow multiple representations) while other domains give rise to posets (e.g., signs, constants and intervals). Of course, any preordered abstract domain $\langle A, \leq, \gamma \rangle$ can be canonically quotiented to a poset $\langle A_{/\cong}, \leq, \gamma \rangle$ where $a \cong a'$ iff $a \leq a'$ and $a' \leq a$. While a monotone concretization γ is enough for reasoning about soundness of static analysis on numerical domains, the notions of best correct approximation and completeness rely on the existence of an abstraction function $\alpha : \wp(\mathbb{R}^n) \to A$ which requires that $\langle A, \leq \rangle$ is (at least) a poset and that the pair (α, γ) forms a Galois connection (GC), i.e. for any $X \subseteq \mathbb{R}^n, a \in A$, $\alpha(X) \leq a \Leftrightarrow X \subseteq \gamma(a)$ holds, which becomes a Galois insertion when γ is injective (or, equivalently, α is surjective). Most numerical domains admit a definition through Galois connections, while for some domains this is impossible, notably for convex polyhedra. Let us recall that the nonrelational interval domain $\mathcal{I}nt = \langle \mathrm{Int}, \leq, \gamma, \alpha \rangle$ is defined by: $\mathrm{Int} \triangleq \{\langle [l_i, u_i] \rangle_{i \in [1,n]} \mid l_i, u_i \in \overline{\mathbb{R}}, l_i \leq u_i\} \cup \bot$, $\gamma(\langle [l_i, u_i] \rangle_{i \in [1,n]}) = \{\mathbf{x} \in \mathbb{R}^n \mid \forall i \in [1, n].\ l_i \leq \mathbf{x}_i \leq u_i\}$, $\gamma(\bot) = \varnothing$, and $\alpha(X) \triangleq \langle \inf_{\mathbf{x} \in X} \mathbf{x}_i, \sup_{\mathbf{x} \in X} \mathbf{x}_i \rangle_{i \in [1,n]}$.

A function $f^\sharp : A \to A$ is a sound approximation of a concrete (transfer) function $f : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ when, for any $a \in A$, $f(\gamma(a)) \subseteq \gamma(f^\sharp(a))$ holds, while $f^\sharp$ is forward-complete (or f-complete or exact) when $f \circ \gamma = \gamma \circ f^\sharp$ holds. Assume that a Galois connection (α, γ) for A exists. The abstract function $f^A \triangleq \alpha \circ f \circ \gamma$ is called the best correct approximation (bca) of f on A. Also, soundness of $f^\sharp$ can be equivalently stated by $\alpha(f(X)) \leq f^\sharp(\alpha(X))$, for any $X \in \wp(\mathbb{R}^n)$, while $f^\sharp$ is defined to be backward-complete (or b-complete or just complete) when $\alpha \circ f = f^\sharp \circ \alpha$ holds.

3 Linear Transforms of Abstract Domains

Linear transformations can be used to recast any existing numerical abstract domains: an invertible linear transformation performs a change of basis of the n-dimensional Euclidean space $\mathbb{R}^n$ and the transformed abstract domain is accordingly interpreted with this transformed coordinate system.

Definition 3.1 (Linear Transform of Abstractions). Consider any invertible matrix $M \in \mathrm{GL}(n)$ and a numerical abstract domain $\mathcal{A} = \langle A, \leq, \gamma \rangle$. The *$M$-transform* of $\mathcal{A}$ is given by $\mathcal{A}^M \triangleq \langle A, \leq, \gamma^M \rangle$, also denoted by $M(\mathcal{A})$, where the concretization map $\gamma^M : A \to \wp(\mathbb{R}^n)$ is defined by $\gamma^M(a) \triangleq M^{-1} \cdot \gamma(a)$. If $\mathcal{A}$ admits an abstraction map $\alpha : \wp(\mathbb{R}^n) \to A$ then $\mathcal{A}^M$ is also endowed with a function $\alpha^M : \wp(\mathbb{R}^n) \to A$ defined by $\alpha^M(X) \triangleq \alpha(M \cdot X)$. □

Equivalently, we have that $\gamma^M = T_{M^{-1}} \circ \gamma$ and $\alpha^M = \alpha \circ T_M$. The basic idea is that the invertible matrix M represents a change of basis for $\mathbb{R}^n$, which can be always converted back through its inverse matrix M^{-1}. According to this view, an abstract value $a \in A$ becomes a symbolic representation of the set of vectors $\gamma(a) \in \wp(\mathbb{R}^n)$ in the new coordinate system based on M, so that the concretization γ^M of a in the original coordinate system of $\mathbb{R}^n$ is given by the conversion of $\gamma(a)$ through M^{-1} back to the original basis of $\mathbb{R}^n$, namely $\gamma^M(a) = M^{-1} \cdot \gamma(a) \in \wp(\mathbb{R}^n)$. Dually, if $\mathcal{A}$ admits an abstraction function α, so that $\langle A, \leq \rangle$ is (at least) partially ordered, then $\mathcal{A}^M$ also has an abstraction map α^M which provides the best approximation of some $X \in \wp(\mathbb{R}^n)$ in A when interpreted w.r.t. the new coordinate system based on M, namely

$\alpha^M(X) = \alpha(M \cdot X)$. Hence, Definition 3.1 is a straightforward generalization of the parallelotope domain defined in [2, Definition 2], since a parallelotope domain indexed by $M \in \mathrm{GL}(n)$ boils down to the M-transform of the interval abstraction $\mathcal{I}nt$.

If $\mathcal{A}$ is a numerical domain equipped with a concretization map γ only, then $\mathcal{A}^M$ is clearly a sound numerical domain, since we just need to check that γ^M still preserves the relation $\leq$ on A: if $a \leq a'$ then $\gamma(a) \subseteq \gamma(a')$, so that $\gamma^M(a) = M^{-1} \cdot \gamma(a) \subseteq M^{-1} \cdot \gamma(a') = \gamma^M(a')$. Moreover, any order-theoretic property of the abstract domain A is obviously preserved when interpreted in its M-transform, e.g., bottom and top elements, lub's and glb's, chains, etc. It is also easy to observe that linear transforms of numerical domains also preserve the existence of abstraction maps.

Lemma 3.2. *If $\mathcal{A} = \langle A, \leq, \gamma, \alpha \rangle$ is a Galois connection (insertion) then its M-transform $\mathcal{A}^M = \langle A, \leq, \gamma^M, \alpha^M \rangle$ is a Galois connection (insertion).*

Example 3.3 (Linear Transform of Constant Propagation) Constant propagation is a well-known and simple abstract interpretation used in compiler optimization for detecting whether a variable at some program point always stores a single constant value for all possible program executions (see, e.g., [18, Section 4.3]). Constant propagation relies on the nonrelational constant abstract domain, which is here given for variables assuming real values: $\mathrm{Const} \triangleq \mathbb{R} \cup \{\bot, \top\}$. Const is endowed with the usual flat partial order: for any $x \in \mathrm{Const}$, $\bot \leq x \leq \top$ (and $x \leq x$), which makes it an infinite complete lattice with height 2. Const is easily defined by a Galois insertion with its standard abstraction and concretization maps $\alpha : \wp(\mathbb{R}) \to \mathrm{Const}$ and $\gamma : \mathrm{Const} \to \wp(\mathbb{R})$.

$$\alpha(X) \triangleq \begin{cases} \bot & \text{if } X = \varnothing \\ z & \text{if } X = \{z\} \\ \top & \text{otherwise} \end{cases} \qquad \gamma(a) \triangleq \begin{cases} \varnothing & \text{if } a = \bot \\ \{a\} & \text{if } a \in \mathbb{R} \\ \mathbb{R} & \text{if } a = \top \end{cases}$$

Let us consider 3 variables and the invertible matrix $S = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$ which is obtained as composition of the two shearing matrices $Sh^{2,1,-1}$ and $Sh^{3,1,-1}$. Its inverse is $S^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$. Let us consider Const for three variables, namely as an abstraction of $\wp(\mathbb{R}^3)$. The matrix S thus induces the transformed domain Const^S, where a vector $\langle a_1, a_2, a_3 \rangle \in \mathrm{Const}^S$, by Definition 3.1, has the following meaning:

$$\gamma^S(\langle a_1, a_2, a_3 \rangle) = S^{-1}\gamma(\langle a_1, a_2, a_3 \rangle) = \\ \{\langle z_1, z_1 + z_2, z_1 + z_3 \rangle \in \mathbb{R}^3 \mid z_1 \in \gamma(a_1), z_2 \in \gamma(a_2), z_3 \in \gamma(a_3)\}.$$

Moreover, if $k_i \in \mathbb{R}$ then $\alpha^S(\{\langle k_1, k_2, k_3 \rangle\}) = \alpha(S\langle k_1, k_2, k_3 \rangle) = \alpha(\{\langle k_1, k_2 - k_1, k_3 - k_1 \rangle\}) = \langle k_1, k_2 - k_1, k_3 - k_1 \rangle$. For instance, if $k_i \in \mathbb{R}$ then $\gamma^S(\langle \top, k_2, k_3 \rangle) = \{\langle z, z + k_2, z + k_3 \rangle \mid z \in \mathbb{R}\}$, $\gamma^S(\langle k_1, \top, k_3 \rangle) = \{\langle k_1, z, k_1 + k_3 \rangle \mid z \in \mathbb{R}\}$, while $\alpha^S(\{\langle -1, 0, 1 \rangle, \langle 1, 1, 3 \rangle\}) = \langle \top, \top, 2 \rangle$ and $\alpha^S(\{\langle -1, 0, 1 \rangle, \langle 1, 2, 3 \rangle\}) = \langle \top, 1, 2 \rangle$. This abstraction Const^S is therefore able to represent invariants for program variables x_i of type $x_1 \in \gamma(a_1) \wedge x_1 + x_2 \in \gamma(a_2) \wedge x_1 + x_3 \in \gamma(a_3)$, where $a_i \in \mathrm{Const}$. For instance, for the following program P already considered in Section 1 and here decorated with program points:

(1) $x_1 := 2;\ x_2 := 3;\ x_3 := 6;$ (2)
while (3) $(x_2 < x_3)$ **do**
    (4) $x_1 := x_1 - 2;$
    (5) $x_3 := x_1 + x_2 + x_3 - 1;$
    (6) $x_2 := x_2 + 2;$
od (7)

while a constant analysis with Const derives no information at program point (3), namely the abstract value $\langle \top, \top, \top \rangle$, we expect that an analysis based on Const^S is able to compute the abstract value $\langle \top, 5, 8 \rangle$ which represents that at program point (3) the additions $x_1 + x_2$ and $x_1 + x_3$ are always equal to, respectively, 5 and 8. □

4 Linear Transforms of Abstract Functions

Background. An abstract interpretation-based static analysis of programs with numeric variables relies on sound approximations of the standard transfer functions on the concrete domain $\wp(\mathbb{R}^n)$ used by the collecting program semantics (see, e.g., [18]): binary set unions and intersections, variable assignments, Boolean tests, widening and narrowing operators. Let us briefly recall the definitions for assignments and tests.

The most general form of variable assignment is given by a parallel (or simultaneous) assignment $[\mathbf{x}_i := f_i(\mathbf{x})]_{i \in [1,n]}$ (as in Python and JavaScript), with generic (possibly nonlinear) functions $f_i : \mathbb{R}^n \to \mathbb{R}$ which define a n-dimensional transform $f : \mathbb{R}^n \to \mathbb{R}^n$ by $f(\mathbf{x}) \triangleq \langle f_1(\mathbf{x}), ..., f_n(\mathbf{x}) \rangle$. The transfer function $\mathbf{assign}(f) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ is the corresponding pointwise extension of f defined by $\mathbf{assign}(f)(X) \triangleq \{f(\mathbf{x}) \mid \mathbf{x} \in X\}$. If $i \in [1, n]$ and $f : \mathbb{R}^n \to \mathbb{R}$ then a single assignment $\mathbf{x}_i := f(\mathbf{x})$ for the i-th variable is defined by $\mathbf{assign}(i, f) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ as the following specific instance: $\mathbf{assign}(i, f)(X) \triangleq \{\langle \mathbf{x}_1, ..., \mathbf{x}_{i-1}, f(\mathbf{x}), \mathbf{x}_{i+1}, ..., \mathbf{x}_n \rangle \mid \mathbf{x} \in X\}$. Linear parallel assignments rely on a square matrix $N \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{R}^n$ which define the transfer function $\mathbf{assign}(N, \mathbf{b}) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ as follows: $\mathbf{assign}(N, \mathbf{b})(X) \triangleq \{N\mathbf{x} + \mathbf{b} \mid \mathbf{x} \in X\}$. As a particular case, linear (single) assignments for the i-th variable consider a vector $\mathbf{a} \in \mathbb{R}^n$ and a constant $b \in \mathbb{R}$ which define the affine transformation $\mathbf{x} \mapsto (\mathbf{e_i}\mathbf{a}^T)\mathbf{x} + b\mathbf{e_i}$ whose corresponding transfer function is $\mathbf{assign}(i, \mathbf{a}, b) \triangleq \mathbf{assign}(\mathbf{e_i}(\mathbf{a} - \mathbf{e_i})^T + I, b\mathbf{e_i})$, namely,

$$\mathbf{assign}(i, \mathbf{a}, b)(X) = \{\langle \mathbf{x}_1, ..., \mathbf{x}_{i-1}, \mathbf{a} \cdot \mathbf{x} + b, \mathbf{x}_{i+1}, ..., \mathbf{x}_n \rangle \mid \mathbf{x} \in X\}.$$

Let us also recall backward assignment, namely the adjoint of a (forward) assignment, which is typically used in backward abstract interpretation [5] for refining the output of a forward abstract interpretation. In general, the transfer function $\mathbf{assign}^{\leftarrow}(f) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ of the backward parallel assignment for $[\mathbf{x} := f(\mathbf{x})]$ is simply given by the inverse image $\mathbf{assign}^{\leftarrow}(f)(Y) \triangleq f^{-1}(Y) = \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \in Y\}$, so that for a single assignment $\mathbf{x}_i := f(\mathbf{x})$, we have that $\mathbf{assign}^{\leftarrow}(i, f)(Y) \triangleq \{\mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{x}_1, ..., f(\mathbf{x}), ..., \mathbf{x}_n \rangle \in Y\}$. In turn, the transfer function of the backward linear parallel assignment for $N \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$ is $\mathbf{assign}^{\leftarrow}(N, \mathbf{b}) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ defined by $\mathbf{assign}^{\leftarrow}(N, \mathbf{b})(Y) \triangleq \{\mathbf{x} \in \mathbb{R}^n \mid N\mathbf{x} + \mathbf{b} \in Y\}$.

A nondeterministic assignment for the i-th variable $x_i := ?$ is modeled by the transfer function $\mathbf{forget}(i) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ defined as follows: $\mathbf{forget}(i)(X) \triangleq$

$\{\langle \mathbf{x}_1, ..., \mathbf{x}_{i-1}, z, \mathbf{x}_{i+1}, ..., \mathbf{x}_n \rangle \mid \mathbf{x} \in X,\ z \in \mathbb{R}\}$. This can be viewed as an instance of a more general function $\mathbf{forget}(\mathbf{v}) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ indexed by a vector $\mathbf{v} \in \mathbb{R}^n$ and defined by $\mathbf{forget}(\mathbf{v})(X) \triangleq \{\mathbf{x} + z\mathbf{v} \mid \mathbf{x} \in X,\ z \in \mathbb{R}\}$. Thus, it turns out that $\mathbf{forget}(i)$ can be retrieved by considering $\mathbf{v} = \mathbf{e_i}$, that is, $\mathbf{forget}(i) = \mathbf{forget}(\mathbf{e_i})$.

The most general form of Boolean test considers any predicate $p : \mathbb{R}^n \to \{\mathtt{t}, \mathtt{f}\}$ and selects those program states that make the predicate p true. This is modeled by a transfer function $\mathbf{test}(p) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ defined by $\mathbf{test}(p)(X) \triangleq X \cap \{\mathbf{x} \in \mathbb{R}^n \mid p(\mathbf{x}) = \mathtt{t}\}$. A linear Boolean test is defined by a matrix $N \in \mathbb{R}^{m \times n}$, a vector $\mathbf{b} \in \mathbb{R}^m$ and some comparison relation $\bowtie \subseteq \mathbb{R}^m \times \mathbb{R}^m$, here used in infix notation, which define a transfer function $\mathbf{test}(N, \mathbf{b}, \bowtie) : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ as follows: $\mathbf{test}(N, \mathbf{b}, \bowtie)(X) \triangleq X \cap \{\mathbf{x} \in \mathbb{R}^n \mid N\mathbf{x} \bowtie \mathbf{b}\}$. As a particular case, we have that if $\mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ then $\mathbf{test}(\mathbf{a}, b, \bowtie)(X) \triangleq X \cap \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a} \cdot \mathbf{x} \bowtie b\}$.

Linear Transforms. Let us consider how abstract operations can be defined on a linear transform of a numerical abstract domain. Consider a numerical abstract domain $\mathcal{A} = \langle A, \leq, \gamma \rangle$, possibly endowed with an abstraction function α. Consider any concrete transfer function $f : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ and a corresponding abstract transfer function $f^\sharp : A \to A$, which may be sound, bca, b-/f-complete w.r.t. f. The following result provides a precise guideline in order to design an abstract transfer function on a transformed domain $\mathcal{A}^M$, with $M \in \mathrm{GL}(n)$.

Lemma 4.1. *$f^\sharp : A \to A$ is sound (bca, b-complete, f-complete) w.r.t. f for the abstract domain $\mathcal{A}^M$ iff $f^\sharp : A \to A$ is sound (bca, b-complete, f-complete) w.r.t. the concrete function $T_M \circ f \circ T_{M^{-1}} : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ for the abstract domain $\mathcal{A}$.*

By analogy with the standard notion of matrix conjugation, the transformed concrete transfer function $T_M \circ f \circ T_{M^{-1}} : \wp(\mathbb{R}^n) \to \wp(\mathbb{R}^n)$ in Lemma 4.1 may be called M-conjugation of the original function f. Indeed, if f is a transfer function for a linear map N then its conjugation $T_M \circ f \circ T_{M^{-1}}$ involves the standard matrix conjugation of N. Lemma 4.1 allows us to design abstract transfer functions for f on the transformed abstraction $\mathcal{A}^M$ by considering the abstract transfer functions on the original abstraction $\mathcal{A}$ but w.r.t. the M-conjugation of f. Hence, if the family of abstract transfer functions handled by some numerical abstract interpretation $\mathcal{A}$ is closed under conjugation then Lemma 4.1 yields a straight and practical technique for designing a full abstract interpretation on the transformed abstraction $\mathcal{A}^M$. The following result provides the linear transformations of abstract functions for $\mathcal{A}^M$ for all the standard operators and linear transfer functions.

Theorem 4.2. *Let $\mathcal{A} = \langle A, \leq, \gamma \rangle$ be a numerical abstract domain, possibly with abstraction map α, and let $M \in \mathrm{GL}(n)$.*

(1) *Let $\mathbf{assign}_A(N, \mathbf{b})$ be a sound abstract transfer function in $\mathcal{A}$ of a linear parallel assignment $\mathbf{assign}(N, \mathbf{b})$. Then, $\mathbf{assign}_A(MNM^{-1}, M\mathbf{b})$ is the corresponding sound transfer function in $\mathcal{A}^M$.*
(2) *Let $\overleftarrow{\mathbf{assign}}_A(N, \mathbf{b})$ be a sound abstract transfer function in $\mathcal{A}$ of a backward linear parallel assignment $\overleftarrow{\mathbf{assign}}(N, \mathbf{b})$. Then, $\overleftarrow{\mathbf{assign}}_A(MNM^{-1}, M\mathbf{b})$ is the corresponding sound transfer function in $\mathcal{A}^M$.*

(3) *Let* $\mathbf{forget}_A(\mathbf{v})$ *be a sound abstract transfer function in* $\mathcal{A}$ *of a nondeterministic assignment* $\mathbf{forget}(\mathbf{v})$*. Then,* $\mathbf{forget}_A(M\mathbf{v})$ *is the corresponding sound transfer function in* $\mathcal{A}^M$*.*
(4) *Let* $\mathbf{test}_A(N, \mathbf{b}, \bowtie)$ *be a sound abstract transfer function in* $\mathcal{A}$ *of a linear Boolean* $\mathbf{test}(N, \mathbf{b}, \bowtie)$*. Then,* $\mathbf{test}_A(NM^{-1}, \mathbf{b}, \bowtie)$ *is the corresponding sound transfer function in* $\mathcal{A}^M$*.*
(5) *If* $\sqcup$ *and* $\sqcap$ *are sound abstract lub and glb in* $\mathcal{A}$ *then* $\sqcup$ *and* $\sqcap$ *are also sound in* $\mathcal{A}^M$*.*
(6) *If* ∇ *and* Δ *are correct widening and narrowing operators in* $\mathcal{A}$ *then* ∇ *and* Δ *are also widening and narrowing in* $\mathcal{A}^M$*.*

As an instance of Theorem 4.2 (1)-(4) to single assignments and tests, we obtain:

Corollary 4.3.

(1) *Let* $\mathbf{assign}_A(i, \mathbf{a}, b)$ *be a sound abstract transfer function in* $\mathcal{A}$ *for a linear single assignment* $x_i := \mathbf{a} \cdot \mathbf{x} + b$*. Then,* $\mathbf{assign}_A(M(\mathbf{e_i}(\mathbf{a} - \mathbf{e_i})^T + I)M^{-1}, M(b\mathbf{e_i}))$ *is the corresponding sound transfer function in* $\mathcal{A}^M$*.*
(2) *Let* $\overleftarrow{\mathbf{assign}}_A(i, \mathbf{a}, b)$ *be a sound abstract transfer function in* $\mathcal{A}$ *for a backward linear single assignment* $x_i := \mathbf{a} \cdot \mathbf{x} + b$*. Then,* $\overleftarrow{\mathbf{assign}}_A(M(\mathbf{e_i}(\mathbf{a} - \mathbf{e_i})^T + I)M^{-1}, M(b\mathbf{e_i}))$ *is the corresponding sound transfer function in* $\mathcal{A}^M$*.*
(3) *Let* $\mathbf{forget}_A(i)$ *be a sound abstract transfer function in* $\mathcal{A}$ *of a nondeterministic assignment* $x_i := ?$*. Then,* $\mathbf{forget}_A(M\mathbf{e_i})$ *is the corresponding sound transfer function in* $\mathcal{A}^M$*.*
(4) *Let* $\mathbf{test}_A(\mathbf{a}, b, \bowtie)$ *be a sound abstract transfer function in* $\mathcal{A}$ *of a linear Boolean test* $\mathbf{a} \cdot \mathbf{x} \bowtie b$*. Then,* $\mathbf{test}_A(\mathbf{a}^T M^{-1}, b, \bowtie)$ *is the corresponding sound transfer function in* $\mathcal{A}^M$*.*

It is important to remark that since Lemma 4.1 goes beyond soundness and also holds for best correct approximations, and backward- and forward-completeness, we also obtain the following consequence of Theorem 4.2 (1)-(5).

Corollary 4.4. *In Theorem 4.2* (1)-(5) *and in Corollary 4.3, sound can be replaced with bca, b-complete and f-complete.*

Hence, this allows us to retrieve as an instance to the transformed interval domain $\mathcal{I}nt^M$ all the corresponding results by Amato et al. [2, Theorems 3, 4, 5, 6] on the best correct approximations of, respectively, lub/glb, linear single assignments, nondeterministic assignments and single Boolean tests for parallelotopes. In particular, Corollary 4.3 holds for best correct approximations, so that once the abstraction $\mathcal{A}$ provides definitions of abstract tests $\mathbf{test}_A(\mathbf{a}, b, \bowtie)$ which are bca's and closed by the matrix multiplications then this same abstraction $\mathcal{A}$ also gives the corresponding bca's in $\mathcal{A}^M$, which are thus given by $\mathbf{test}_A(\mathbf{a}^T M^{-1}, b, \bowtie)$. For linear assignments, it is important to remark that the linear transform of abstract single assignments may well lead to abstract parallel assignments, as shown by the following example for the parallelotope domain.

Example 4.5 Let $M = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \in \mathrm{GL}(2)$, as considered in [2, Example 1] and obtained by composing a scaling with a Givens rotation, namely $M = D^{(\sqrt{2},\sqrt{2})}R^{1,2,\frac{\pi}{4}}$. Consider two program variables and a single assignment such as $x_1 := k$, for some

constant $k \in \mathbb{R}$, whose best correct approximation for the interval domain Int for two variables is given by $\mathbf{assign}_{\mathrm{Int}}(1,(0,0),k)$. Then, by Corollary 4.4, the best correct approximation of $x_1 := k$ for the parallelotope $\mathcal{I}nt^M$ is given by the parallel assignment $\mathbf{assign}_{\mathrm{Int}}(M\begin{pmatrix}0&0\\0&1\end{pmatrix}M^{-1}, M(k,0)) = \mathbf{assign}_{\mathrm{Int}}(\begin{pmatrix}0.5&-0.5\\-0.5&0.5\end{pmatrix},(k,k))$, namely it coincides with the best correct approximation in Int of the following linear parallel assignment: $[x_1 := 0.5x_1 - 0.5x_2 + k;\ \ x_2 := -0.5x_1 + 0.5x_2 + k;]$. □

5 Transforming Linear Programs

We observe that in the proof of Theorem 4.2, and in turn in Corollary 4.3, the implications from sound (bca, b-complete, f-complete) abstract transfer functions in $\mathcal{A}$ to corresponding sound (bca, b-complete, f-complete) abstract transfer functions in $\mathcal{A}^M$ are indeed *equivalences*. This is a straight consequence of Lemma 4.1, which indeed shows an equivalence between the abstract transfer functions for $\mathcal{A}$ and $\mathcal{A}^M$. Thus, since best correct approximations are always unique, as well as (backward or forward) complete abstract functions, when they exist, are unique, we obtain the following characterizations of linear single assignments and tests.

Theorem 5.1.

(1) *The bca in $\mathcal{A}^M$ of a linear single assignment $x_i := \mathbf{a}\cdot\mathbf{x} + b$ coincides with the bca in $\mathcal{A}$ of the linear (possibly) parallel assignment* $[\mathbf{x} := M(\mathbf{e_i}(\mathbf{a}-\mathbf{e_i})^T + I)M^{-1}\mathbf{x} + M(b\mathbf{e_i});]$.

(2) *The bca in $\mathcal{A}^M$ of a linear Boolean test $\mathbf{a}\cdot\mathbf{x} \bowtie b$ coincides with the bca in $\mathcal{A}$ of the linear Boolean test $\mathbf{a}^T M^{-1}\mathbf{x} \bowtie b$.*

Moreover, both in (1) *and* (2)*, the bca in $\mathcal{A}^M$ is b-complete (f-complete) iff the bca in $\mathcal{A}$ is b-complete (f-complete), and in this case they coincide.*

This means that existence of the bca in either domain $\mathcal{A}$ or $\mathcal{A}^M$ implies the existence of the bca in the other domain. This also hints that an analysis with the transformed abstraction $\mathcal{A}^M$ of a program P consisting of linear assignments and tests only can be obtained by analysing with the original abstraction $\mathcal{A}$ a program P^M which is obtained from P by transforming all its linear assignments and tests by exploiting Theorem 5.1, so as to maintain the same program points (i.e., the control flow graphs of P and P^M coincide). In particular, if the analysis in $\mathcal{A}$ of the assignments and tests occurring in the transformed program P^M relies on abstract transfer functions which are the best correct approximations in $\mathcal{A}$ then Theorem 5.1 guarantees that at each program point of P^M we obtain *exactly* the same (best) abstract value that we would have obtained at the same program point by analysing P in $\mathcal{A}^M$. Instead, if the analysis of P^M in $\mathcal{A}$ exploits some abstract transfer functions which are not bca's in $\mathcal{A}$, then we achieve abstract values for P which are still sound in $\mathcal{A}^M$, although, of course, they are not guaranteed to be the best possible abstract values in $\mathcal{A}^M$, since possible losses of precision in $\mathcal{A}$ are shifted to $\mathcal{A}^M$.

As shown in Example 4.5, it should be noted that even if P does not contain parallel assignments, the transformed program P^M may well include parallel assignments. Thus, the program analysis design in $\mathcal{A}$ should also include abstract transfer functions

for parallel linear assignments. Of course, this program transformation has a cost. The computational time complexity of the transform $P \mapsto P^M$ of Theorem 5.1 is $O(n^2)$ for each linear assignment and test occurring in P, as argued in [2, Section 5] for the case of parallelotopes (the transforms are exactly the same). We envision that this program transform can be implemented as a preprocessing step of the analysis in $\mathcal{A}^M$. Let us consider a first example with parallelotopes.

Example 5.2 (Parallelotopes) Consider the following program P taken from [2]:

$x_1 := 4;\ x_2 := -4;$
while $(x_1 > x_2)$ **do**
    $x_1 := x_1 - 1;$
    $x_2 := x_2 + 1;$

As argued in [2, Section 1] and [20], a statistical dynamic analysis such as orthogonal simple component analysis may determine that the analysis of P using the parallelotope instance $\mathcal{I}nt^M$ may provide precise results when the matrix is $M = \left(\begin{smallmatrix} 1 & -1 \\ 1 & 1 \end{smallmatrix}\right)$, namely when M is the matrix of Example 4.5 obtained by first applying a $\frac{\pi}{4}$ clockwise rotation matrix followed by a $\sqrt{2}$ scaling for both x_1 and x_2. Let us also recall that $M^{-1} = \left(\begin{smallmatrix} 0.5 & 0.5 \\ -0.5 & 0.5 \end{smallmatrix}\right)$. Any vector $\langle x_1, x_2 \rangle \in \mathbb{R}^2$ is thus transformed into $M\left(\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} x_1 - x_2 \\ x_1 + x_2 \end{smallmatrix}\right)$, namely Int^M is able to represent the program invariants: $\{l_1 \leq x_1 - x_2 \leq u_1,\ l_2 \leq x_1 + x_2 \leq u_2\}$, with $l_i, u_i \in \overline{\mathbb{R}}$. Conversely, any vector of intervals $\langle [l_1, u_1], [l_2, u_2] \rangle \in \mathrm{Int}^M$ represents the set of stores $M^{-1} \cdot \gamma_{\mathrm{Int}}(\langle [l_1, u_1], [l_2, u_2] \rangle) = \{\langle 0.5z_1 + 0.5z_2, -0.5z_1 + 0.5z_2 \rangle \in \mathbb{R}^2 \mid l_1 \leq z_1 \leq u_1,\ l_2 \leq z_2 \leq u_2\}$. By Theorem 5.1, P is transformed into the following program P^M, where, for the sake of clarity, we use variables y_i:

$y_1 := 8;\ y_2 := 0;$
while $(y_1 > 0)$ **do**
    $y_1 := y_1 - 2;$

This transformed program P^M is obtained as follows. The initializations $\{x_1 := 4;\ x_2 := -4;\}$ coincide with the parallel assignment $[\mathbf{x} := \mathbf{0}_2\mathbf{x} + \left(\begin{smallmatrix} 4 \\ -4 \end{smallmatrix}\right)]$ whose M-transform is $[\mathbf{y} := (M\mathbf{0}_2M^{-1})\mathbf{y} + M\left(\begin{smallmatrix} 4 \\ -4 \end{smallmatrix}\right)] = [\mathbf{y} := \mathbf{0}_2\mathbf{y} + \left(\begin{smallmatrix} 8 \\ 0 \end{smallmatrix}\right)]$, namely $[y_1 := 8;\ y_2 := 0;]$. The guard $(x_1 > x_2)$ corresponds to the Boolean test $(1\ -1)\mathbf{x} > 0$, whose M-transform is $\big((1\ -1)M^{-1}\big)\mathbf{y} > 0$, namely $(1\ 0)\mathbf{y} > 0$, which is the guard $(y_1 > 0)$. Finally, the assignments $\{x_1 := x_1 - 1;\ x_2 := x_2 + 1;\}$ correspond to the parallel assignment $[\mathbf{x} = I_2\mathbf{x} + \left(\begin{smallmatrix} -1 \\ 1 \end{smallmatrix}\right)]$, which is M-transformed to $[\mathbf{y} = (MI_2M^{-1})\mathbf{y} + M\left(\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right)]$, that is, $[y_1 := y_1 - 2;\ y_2 := y_2;]$.
Since the interval abstraction $\mathcal{I}nt$ provides best correct approximations for all the transfer functions of the statements occurring in the transformed program P_M, by Theorem 5.1, it turns out that the analysis of P_M using $\mathcal{I}nt$ gives exactly the most precise program invariants for P in $\mathcal{I}nt^M$. The analysis of P_M using $\mathcal{I}nt$ with widening provides $\{y_1 \leq 8,\ y_2 = 0\}$ as loop invariant, so that at the exit point we obtain $\{y_1 \leq 0,\ y_2 = 0\}$. Hence, the concrete interpretation of the output of this analysis states that at the exit point of the original program P the invariant $x_1 - x_2 \leq 0 \wedge x_1 + x_2 = 0$

holds, whose abstraction in Int is $\{x_1 \leq 0,\ x_2 \geq 0\}$. The analysis of P using $\mathcal{I}nt$ with widening is much less precise, since it yields the interval $\{x_1 \leq 4, x_2 \geq -4\}$ both as loop invariant and at the exit point. □

In the following we consider a couple of examples different from parallelotopes, namely linear transforms of constant propagation and octagon analysis.

5.1 Linear Transform of Constant Propagation

Let us carry on Example 3.3 on the linear transform Const^S of the constant propagation domain, which is able to represent invariants of type $\{x_1 = a_1,\ x_1{+}x_2 = a_2,\ x_1{+}x_3 = a_3\}$, where $a_i \in \mathrm{Const}$. In order to analyze the program P in Example 3.3 using the abstraction Const^S, we compute its transform P^S by exploiting Theorem 5.1.
The initializations within the program points (1)-(2) correspond to the parallel assignment $[\mathbf{x} = \mathbf{0}_3\mathbf{x} + \begin{pmatrix}2\\3\\6\end{pmatrix}]$, whose S-transform is: $[\mathbf{y} = (S\mathbf{0}_3S^{-1})\mathbf{y} + S\begin{pmatrix}2\\3\\6\end{pmatrix}]$, which is $[y_1 := 2;\ y_2 := 5;\ y_3 := 8;]$. The guard $(x_2 < x_3)$ corresponds to the Boolean test $(0\ 1\ -1)\mathbf{x} < 0$, which is transformed into $\big((0\ 1\ -1)S^{-1}\big)\mathbf{y} < 0$, which leaves it unchanged, i.e. $(y_2 < y_3)$. The S-transforms, denoted by $\Rightarrow^S$, of the three assignments in the body of the while-loop at program points (4)-(5)-(6) are computed in Fig. 2. We obtain the following transformed program P^S:

(1) $y_1 := 2;\ y_2 := 5\ y_3 := 8;$ (2)
while (3) $(y_2 < y_3)$ **do**
  (4) $y_1 := y_1 - 2;\ y_2 := y_2 - 2;\ y_3 := y_3 - 2;$
  (5) $y_3 := y_2 + y_3 - 1;$
  (6) $y_2 := y_2 + 2;$
od (7)

All the abstract transfer functions for linear assignments and tests in the constant propagation abstraction Const are best correct approximations. Hence, the optimal analysis of P with Const^S is achieved by analysing P^S with Const, where widening is obviously not needed. The analysis of P^S at program point (3) computes the invariant $\langle y_1 = \top, y_2 = 5, y_3 = 8\rangle \in \mathrm{Const}$. Thus, at the exit point (7), we obtain $\mathbf{test}_{\mathrm{Const}}(\neg(y_2 < y_3))\langle y_1 = \top, y_2 = 5, y_3 = 8\rangle = \bot_{\mathrm{Const}}$, which allows us to derive that the exit point (7) is unreachable. By contrast, constant propagation analysis of the original program P gives no information in (3), namely it computes the invariant $(x_1 = \top, x_2 = \top, x_3 = \top)$, so that nothing can be derived at the exit point (7).

5.2 Linear Transform of Octagons

Recall that the weakly-relational octagon abstract domain $\mathcal{O}ct = \langle \mathrm{Oct}, \leq, \gamma_{\mathrm{Oct}}, \alpha_{\mathrm{Oct}}\rangle$ represents program invariants of type $l \leq \pm x_i \pm x_j \leq u$ and $l \leq x_i \leq u$ for $l, u \in \overline{\mathbb{R}}$ [16,17]. Assume that we want to infer that program point (5) of program P in Fig. 3 is unreachable. The analysis of P using $\mathcal{O}ct$ with its standard widening operator computes the invariant $\{x_1 \leq 2, x_2 \leq 4, x_2 - x_1 \leq 2\}$ at program point (2), so that at program point (4) we get $\{x_1 \leq 2, x_2 \leq 0, x_2 - x_1 \leq 2\}$, and, in turn, $\{0 < x_1 \leq 2,\ x_2 \leq$

$$
\begin{array}{rll}
(4)\ x_1 := x_1 - 2; & \Leftrightarrow & [\mathbf{x} = I_3\mathbf{x} + \begin{pmatrix} -2 \\ 0 \\ 0 \end{pmatrix}] \Rightarrow^S \\
& & [\mathbf{y} = (SI_3S^{-1})\mathbf{y} + S\begin{pmatrix} -2 \\ 0 \\ 0 \end{pmatrix}] = [\mathbf{y} = I_3\mathbf{y} + \begin{pmatrix} -2 \\ -2 \\ -2 \end{pmatrix}] \Leftrightarrow \\
& & [y_1 := y_1 - 2;\ y_2 := y_2 - 2;\ y_3 := y_3 - 2;] \\
(5)\ x_3 := x_1 + x_2 + x_3 - 1; & \Leftrightarrow & [\mathbf{x} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}\mathbf{x} + \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}] \Rightarrow^S \\
& & [\mathbf{y} = (S\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}S^{-1})\mathbf{y} + S\begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}] = \\
& & [\mathbf{y} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}\mathbf{y} + \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}] \Leftrightarrow \\
& & [y_1 := y_1;\ y_2 := y_2;\ y_3 := y_2 + y_3 - 1;] \\
(6)\ x_2 := x_2 + 2; & \Leftrightarrow & [\mathbf{x} = I_3\mathbf{x} + \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix}] \Rightarrow^S \\
& & [\mathbf{y} = (SI_3S^{-1})\mathbf{y} + S\begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix}] = [\mathbf{y} = I_3\mathbf{y} + \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix}] \Leftrightarrow \\
& & [y_1 := y_1;\ y_2 := y_2 + 2;\ y_3 := y_3;]
\end{array}
$$

Fig. 2: Linear transforms of assignments.

(1) $x_1 := 2;\ x_2 := 4;$	(1) $[y_1 := 2;\ y_2 := 2;]$
while (2) $(x_2 > 0)$ **do**	**while** (2) $(y_1 + y_2 > 0)$ **do**
(3) $x_1 := x_1 - 1;\ x_2 := x_2 - 2;$	(3) $[y_1 := y_1 - 1;\ y_2 := y_2 - 1;]$
od	**od**
if (4) $(x_1 > 0)$ **then** (5) ...	**if** (4) $(y_1 > 0)$ **then** (5) ...

Fig. 3: The program P, on the left, and its M-transform P^M, on the right.

$0,\ x_2 - x_1 \leq 2\} = \{0 < x_1 \leq 2, x_2 \leq 0\}$ at program point (5), which does not allow us to detect that (5) is an unreachable program point.

Let us consider the matrix $M = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \in \mathrm{GL}(2)$, which is the shearing matrix $Sh^{2,1,-1}$ and whose inverse is $M^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. A vector $\langle x_1, x_2 \rangle \in \mathbb{R}^2$ is transformed into $M\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ -x_1 + x_2 \end{pmatrix}$, so that Oct^M is able to represent the program invariants: $\{l \leq x_1 \leq u,\ l \leq x_2 \leq u,\ l \leq x_1 - x_2 \leq u,\ l \leq 2x_1 - x_2 \leq u\}$. On the other hand, an octagon $oct \in \mathrm{Oct}^M$ represents the set of vectors $M^{-1} \cdot \gamma_{\mathrm{Oct}}(oct) = \{\langle z_1, z_1 + z_2 \rangle \in \mathbb{R}^2 \mid \langle z_1, z_2 \rangle \in \gamma_{\mathrm{Oct}}(oct)\}$. The computations of the M-transform P^M are given in Figure 4 by exploiting Theorem 5.1. The assignments (which are single assignments) and tests occurring in P^M are of type $y_i := u_i + k$, $(y_i + y_j \leq k)$ and $(y_i \leq k)$, and Oct provides best correct approximations for them [17, Sections 4.4, 4.5]. Hence, the analysis of P^M using the original octagon abstraction $\mathcal{O}ct$ with widening operator is optimal. This analysis computes the invariant $\{y_1 \leq 2, y_2 \leq 2, y_1 - y_2 = 0\}$ at program point (2), so that we obtain $\{y_1 \leq 2, y_2 \leq 2, y_1 - y_2 = 0, y_1 + y_2 \leq 0\} =$

$$(1)\ x_1 := 2;\ x_2 := 4;\ \Leftrightarrow\ [\mathbf{x} = \mathbf{0}_2\mathbf{x} + \tbinom{2}{4}]\ \Rightarrow^M\ [\mathbf{y} = (M\mathbf{0}_2M^{-1})\mathbf{y} + M\tbinom{2}{4}]\ =$$
$$[\mathbf{y} = \mathbf{0}_2\mathbf{y} + \tbinom{2}{2}]\ \Leftrightarrow [y_1 := 2;\ y_2 := 2;]$$

$$(2)\ (x_2 > 0)\ \Leftrightarrow\ (0\ \ 1)\mathbf{x} > 0\ \Rightarrow^M\ \big((0\ \ 1)M^{-1}\big)\mathbf{y} > 0\ =$$
$$(1\ \ 1)\mathbf{y} > 0\ \Leftrightarrow\ (y_1 + y_2 > 0)$$

$$(3)\ x_1 := x_1 - 1;\ x_2 := x_2 - 2;\ \Leftrightarrow\ [\mathbf{x} = I_2\mathbf{x} + \tbinom{-1}{-2}]\ \Rightarrow^M\ [\mathbf{y} = (MI_3M^{-1})\mathbf{y} + M\tbinom{-1}{-2}]\ =$$
$$[\mathbf{y} = I_3\mathbf{y} + \tbinom{-1}{-1}]\ \Leftrightarrow [y_1 := y_1 - 1;\ y_2 := y_2 - 1;]$$

$$(4)\ (x_1 > 0)\ \Leftrightarrow\ (1\ \ 0)\mathbf{x} > 0\ \Rightarrow^M\ \big((1\ \ 0)M^{-1}\big)\mathbf{y} > 0\ =$$
$$(1\ \ 0)\mathbf{y} > 0\ \Leftrightarrow\ (y_1 > 0)$$

Fig. 4: M-transform of statements occurring in P.

```
(1) x1 := 2; x2 := 4;
while (2) (x2 > 0) do
    (3) x1 := x1 − 1;
        if (rnd > 0) x2 := x2 − 2;
        else x2 := x2 − 1;
od
if (4) (x1 > 0) then (5) ...
```

```
(1) y1 := 2; y2 := 2;
while (2) (y1 + y2 > 0) do
    (3) [y1 := y1 − 1; y2 := y2 + 1;]
        if (rnd > 0) y2 := y2 − 2;
        else y2 := y2 − 1;
od
if (4) (y1 > 0) then (5) ...
```

Fig. 5: The program Q, on the left, and its M-transform Q^M, on the right.

$\{y_1 \leq 0, y_2 \leq 0, y_1 - y_2 = 0\}$ at program point (4). Hence, this analysis infers the invariant $\{y_1 \leq 0, y_2 \leq 0, y_1 - y_2 = 0, y_1 > 0\}$ at program point (5). The reduction of this octagonal constraint shows that (5) is an unreachable program point in P_M, thus proving that (5) is an unreachable program point in the original program P.

Consider now the program Q in Figure 5, where **rnd** outputs a random value. Here again, the goal is to check that (5) is an unreachable program point. By comparison, let us first consider the analysis of Q using parallelotopes. A dynamic analysis of Q typically derives from the partial traces at program point (2) (e.g., $\langle 2,4\rangle \rightarrow \langle 1,3\rangle \rightarrow \langle 0,2\rangle \rightarrow \langle -1,1\rangle \rightarrow \langle -2,0\rangle$; $\langle 2,4\rangle \rightarrow \langle 1,2\rangle \rightarrow \langle 0,0\rangle$; $\langle 2,4\rangle \rightarrow \langle 1,3\rangle \rightarrow \langle 0,1\rangle \rightarrow \langle -1,0\rangle$) that an analysis based on parallelotopes should represent precisely the program invariants $l \leq x_1 - x_2 \leq u$ and $l \leq 2x_1 - x_2 \leq u$, corresponding to the matrix $N = \binom{1\ -1}{2\ -1} \in \mathrm{GL}(2)$. The analysis of Q using these N-parallelotopes with widening computes the loop invariant $prl \triangleq \{-2 \leq x_1 - x_2, 2x_1 - x_2 \leq 0\}$ at program point (2). Consequently, the most precise parallelotope approximating $prl \cap \{x_2 \leq 0\}$ at program point (4) still is prl itself. In turn, at program point (5) the best possible approximation of $prl \cap \{x_1 > 0\}$ is given again by prl, and prl does not allow to infer that (5) is unreachable. By contrast, let us consider the M-transform Q^M in Figure 5, which is obtained simply by adding the transform of $x_2 := x_2 - 1$ to the transforms in Figure 4. The analysis of Q^M using the original octagon abstraction $\mathcal{O}ct$ with widening computes the invariant

$\{y_1 \leq 2, y_2 \leq 2, y_1 - y_2 \leq 0\}$ at program point (2), and $\{y_1 \leq 2, y_2 \leq 2, y_1 - y_2 \leq 0, y_1 + y_2 \leq 0\}$ at program point (4). Hence, after a reduction of this latter octagon, one obtains $\{y_1 \leq 0, y_2 \leq 2, y_1 - y_2 \leq 0, y_1 + y_2 \leq 0\}$ at program point (4). In turn, $\{y_1 \leq 0, y_2 \leq 2, y_1 - y_2 \leq 0, y_1 + y_2 \leq 0\} \cap \{y_1 > 0\}$ allows us to derive that that (5) is an unreachable program point in P_M. Hence, the analysis of P with $\mathcal{O}ct^M$ is able to infer that the program point (5) in P is unreachable. Finally, let us observe that even the analysis of Q using M-parallelotopes, which represent invariants of type $l \leq x_1 \leq u$ and $l \leq x_2 - x_1 \leq u$, remains inconclusive: here the loop invariant computed at (2) is $\{x_1 \leq 2,\ x_2 - x_1 \leq 2\}$, which is also the best invariant at (4), and therefore does not allow to infer that (5) is unreachable.

6 Completeness for Linear Transforms

Let us recall [13] that if $\mathcal{A} = \langle A, \leq, \gamma, \alpha \rangle$ is a Galois Connection and an abstract function $f^\sharp : A \to A$ is f-complete or b-complete for a concrete function $f : C \to C$ then $f^\sharp = f^A$ holds, so that the property of being f- or b-complete for $f^\sharp$ actually depends on the domain A only, i.e., it is an abstract domain property. Hence, by defining the closure operator $\rho \triangleq \gamma \circ \alpha : C \to C$, which encodes an abstraction independently of the representation of its elements, an abstract domain A is defined to be f-complete for f when $\rho \circ f \circ \rho = f \circ \rho$ holds and b-complete for f when $\rho \circ f \circ \rho = \rho \circ f$ holds. It is shown in [13, Section 5] that any abstract domain can be refined to its so-called *complete shell* to attain b-completeness, namely, for any domain A and any set of concrete functions $F \subseteq C \to C$ there exists the least refinement $\mathrm{Shell}_F(A)$ of A which is b-complete for F, provided that C is a complete lattice and the functions in F are Scott-continuous.

The following result shows that a linear transform $M(\mathcal{A})$ is equivalent to its input abstract domain $\mathcal{A}$ exactly when $\mathcal{A}$ is backward and forward complete for the linear transformation T_M. This formalizes the intuition that in order to be beneficially used in program analysis, a linear transform $M(\mathcal{A})$ must be applied to abstractions $\mathcal{A}$ which are either backward or forward *incomplete* for M. Recall that two abstract domains $\mathcal{A}_i = \langle A_i, \leq_i, \gamma_i \rangle$, $i = 1, 2$, are equivalent, denoted by $\mathcal{A}_1 \cong \mathcal{A}_2$, when they represent the same concrete sets, i.e., when $\gamma_1(A_1) = \gamma_2(A_2)$ holds.

Theorem 6.1. *Let $\mathcal{A} = \langle A, \leq, \gamma, \alpha \rangle$ be a numerical abstract domain defined by a GC and let $M \in \mathrm{GL}(n)$. Then, $M(\mathcal{A}) \cong \mathcal{A}$ iff A is b- and f-complete for T_M.*

Let $\mathcal{K}$, $\mathcal{T}_X$ and $\mathcal{P}$ denote, respectively, the relational abstract domains of affine equalities, also called Karr's domain [14], templates for some $m \times n$ matrix X [19] and convex polyhedra [8]. As expected, it turns out that any linear transform of these numerical domains is ineffective.

Lemma 6.2. *For any $M \in \mathrm{GL}(n)$, $M(\mathcal{K}) \cong \mathcal{K}$, $M(\mathcal{T}_X) \cong \mathcal{T}_X$ and $M(\mathcal{P}) \cong \mathcal{P}$.*

As a consequence of Theorem 6.1 and Lemma 6.2, since both abstract domains $\mathcal{K}$ and $\mathcal{T}$ can be defined through a Galois connection (see, e.g., [18, Section 5]), we derive that for any $M \in \mathrm{GL}(n)$, Karr's $\mathcal{K}$ and template $\mathcal{T}$ abstract domains are backward and forward complete for T_M, as hinted by the intuition. Convex polyhedra do not have an abstraction map, so that completeness does not play a role.

Let us now focus on intervals $\mathcal{I}nt$ and octagons $\mathcal{O}ct$. Recall (see e.g. [21]) that $M \in \mathrm{GL}(n)$ is a *monomial matrix* (or generalized permutation matrix) if each row and column of M has exactly one nonzero entry and all other entries are 0. It turns out that $M \in \mathrm{GL}(n)$ is a monomial matrix if and only if M can be written as a product of an invertible diagonal matrix and a permutation matrix (i.e., each row and column has exactly one 1 and all other entries are 0). We denote by $\mathrm{Mon}(n) \subseteq \mathrm{GL}(n)$ the subset of monomial matrices, which is actually a subgroup (for matrix multiplication). It turns out that monomial matrices characterize precisely the linear transforms which are ineffective for intervals and octagons, where the intuition is that a monomial matrix represents a nonrelational linear transform.

Lemma 6.3. *Let $M \in \mathrm{GL}(n)$. If $n \geq 3$ then $M \in \mathrm{Mon}(n)$ iff $M(\mathcal{I}nt) \cong \mathcal{I}nt$ iff $M(\mathcal{O}ct) \cong \mathcal{O}ct$. If $n = 2$ then $M \in \mathrm{Mon}(n)$ iff $M(\mathcal{I}nt) \cong \mathcal{I}nt$.*

By combining Theorem 6.1 and Lemma 6.3, we derive the following noteworthy consequence: octagons cannot be obtained as a completeness shell from intervals for some family of invertible linear transforms in $\mathrm{GL}(n)$.

Theorem 6.4. *For all $n \geq 3$ and $\mathcal{T} \subseteq \{T_M \mid M \in \mathrm{GL}(n)\}$, $\mathrm{Shell}_{\mathcal{T}}(\mathrm{Int}) \not\cong \mathrm{Oct}$.*

This result is somehow against the intuition that octagons are (backward and forward) complete for $\frac{\pi}{4}$ rotations and therefore could be designed through a complete shell of intervals for this family of rotations. Instead, this intuition holds just in 2D, namely for two variables only.

Lemma 6.5. *Let $\mathcal{R}_{1,2,\frac{\pi}{4}} : \wp(\mathbb{R}^2) \to \wp(\mathbb{R}^2)$ be transformation function for the $\frac{\pi}{4}$ rotation matrix in $\mathrm{GL}(2)$. Then, $\mathrm{Shell}_{\mathcal{R}_{1,2,\frac{\pi}{4}}}(\mathrm{Int}) \cong \mathrm{Oct}$.*

While octagons cannot be obtained from intervals through a complete shell for $\frac{\pi}{4}$ rotations when $n \geq 3$, they can still be synthesized through a suitable reduced (or Cartesian) product [7], here denoted by Π and $\sqcap$, of $\frac{\pi}{4}$ rotations of intervals.

Lemma 6.6. *For all $n \geq 3$, $\mathcal{O}ct \cong \Pi^n_{i,j=1,i<j} R^{i,j,\frac{\pi}{4}}(\mathcal{I}nt)$. Furthermore, $\mathcal{O}ct \cong \Pi^n_{i,j=1,i<j}\big(Sh^{i,j,1}(\mathcal{I}nt) \sqcap Sh^{i,j,-1}(\mathcal{I}nt)\big)$.*

The intuition is quite simple. Octagons can be viewed as the product of all the $\frac{\pi}{4}$ rotational transforms of intervals because any such transform $R^{i,j,\frac{\pi}{4}}(\mathcal{I}nt)$, with $i < j$, is able to represent the program invariants $l \leq x_i + x_j \leq u$, $l \leq x_i - x_j \leq u$ and $l \leq x_k \leq u$, for any $k \in [1, n] \setminus \{i, j\}$, so that their reduced product precisely expresses all the octagonal constraints in Oct. Similarly, a reduced product $Sh^{i,j,1}(\mathcal{I}nt) \sqcap Sh^{i,j,-1}(\mathcal{I}nt)$ of two shearing transforms represents $l \leq x_i + x_j \leq u$, $l \leq x_i - x_j \leq u$ and $l \leq x_k \leq u$, for any $k \neq i$, so that their reduced product still gets back all the octagons.

7 Further Work

We have shown how the idea behind the definition of the abstract domain of parallelotopes can be generalized and pushed forward to the class of numerical abstract domains which are not complete for invertible linear transforms. We proved how linear transforms of abstract domains closely correspond to linear transforms of programs, since

the analysis of a program P on a linearly transformed domain $M(\mathcal{A})$ can be designed as the analysis of a linearly transformed program $M(P)$ on the original abstract domain $\mathcal{A}$.

As argued in [1,2] for parallelotopes, a good linear transformation matrix M to be used for analyzing a program P with an abstraction $\mathcal{A}$ can be derived by resorting to some statistical technique applied to the data obtained by a dynamic analysis of P. This approach appears to be promising for parallelotopes [2] and therefore it is worth to pursue an investigation of it for linear transforms of octagons by exploiting the general framework of this article. In particular, this would be appealing since octagons have a cubic time complexity, while the cost of applying a linear transform to octagons is quadratic for any assignment and Boolean test occurring in the program to analyze. Moreover, one could also investigate how to adapt and generalize the dynamic approach studied in [3,4] where the linear transform M is part of the abstract value and therefore may be changed by the abstract transfer functions during the analysis of a program. Finally, let us observe that a broad perspective of our analysis technique with a linearly transformed abstraction $M(\mathcal{A})$ is that in order to analyze a program P with some abstraction $\mathcal{A}$, P is first transformed into P', then P' is analyzed with a different but related abstraction $\mathcal{A}'$, and the output of this latter analysis is projected back into $\mathcal{A}$ for the program P. In a sense, this can be seen as a proof-of-concept of a more general problem in program analysis. It is known that the precision of program analyses is an extensional property (analogously to computational complexity of programs), namely the precision of an analysis of P depends upon the way the code of P is written. The possibility of increasing or reducing the precision of the analysis of a program P by transforming the code of P has not been investigated and our transformational approach can be viewed as a step towards this goal.

Acknowledgements. We are grateful to the anonymous referees for their helpful remarks. The doctoral fellowship of Marco Zanella is funded by Fondazione Bruno Kessler (FBK), Trento, Italy.

References

1. G. Amato, M. Parton, and F. Scozzari. Deriving numerical abstract domains via principal component analysis. In *Proceedings 17th International Static Analysis Symposium (SAS'10)*, Springer LNCS vol. 6337, pp. 134-150, 2010.
2. G. Amato, M. Parton, and F. Scozzari. Discovering invariants via simple component analysis. *J. Symbolic Computation*, 47:1533-1560, 2012.
3. G. Amato, M. Rubino, and F. Scozzari. Inferring linear invariants with parallelotopes. *Science of Computer Programming*, 148:161-188, 2017.
4. G. Amato and F. Scozzari. The abstract domain of parallelotopes. In *Proc. 4th Int. Workshop on Numerical and Symbolic Abstract Domains (NSAD'12)*, ENTCS 287:17-28, 2012.
5. F. Bourdoncle. Abstract debugging of higher-order imperative languages. In *Proc. ACM Intern. Conf. on Programming Languages Design and Implementation (PLDI'93)*, pp. 46-55, ACM Press, 1993.
6. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixed points. In *Proc. 4th ACM Symposium on Principles of Programming Languages (POPL'77)*, pp. 238-252, ACM Press, 1977.
7. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of programming languages*, pages 269–282. ACM, 1979.

8. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proc. 5th ACM Symp. on Principles of Programming Languages (POPL'78)*, pp. 84-97, ACM, 1978.
9. G. Filé, R. Giacobazzi, and F. Ranzato. A unifying view of abstract domain design. *ACM Computing Surveys*, 28(2):333–336, 1996.
10. G. Filé and F. Ranzato. The powerset operator on abstract interpretations. *Theoretical Computer Science*, 222(1):77–111, 1999.
11. R. Giacobazzi and F. Ranzato. Refining and compressing abstract domains. In Proc. *24th International Colloquium on Automata, Languages, and Programming (ICALP'97)*, Springer LNCS vol. 1256, pp. 771-781, 1997.
12. R. Giacobazzi and F. Ranzato. The reduced relative power operation on abstract domains. *Theoretical Computer Science* 216(1-2):159-211, 1999.
13. R. Giacobazzi, F. Ranzato and F. Scozzari. Making abstract interpretations complete. *J. ACM*, 47(2):361-416, 2000.
14. M. Karr. Affine relationships among variables of a program. *Acta Informatica*, 6:133-151, 1976.
15. G.A. Kildall. A unified approach to global program optimization. In *Proc. 1st ACM Symp. on Principles of Programming Languages (POPL'73)*, pp. 194-206, ACM, 1973.
16. A. Miné. *Weakly Relational Numerical Abstract Domains*. PhD thesis, École Normale Supérieure, Paris, France, 2004.
17. A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, 2006.
18. A. Miné. Tutorial on static inference of numeric invariants by abstract interpretation. *Foundations and Trends in Programming Languages*, 4(3-4):120-372, 2017.
19. S. Sankaranarayanan, H. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *Proc. 6th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI'05)*, Springer LNCS vol. 3385, pp. 21-47, 2005.
20. Y. Seladji. Finding relevant templates via the principal component analysis. In *Proc. Verification, Model Checking, and Abstract Interpretation (VMCAI'17)*, Springer LNCS vol. 10145, pp. 483-499, 2017.
21. X. Zhan. *Matrix Theory.* American Mathematical Society, 2013.