Scalable Minimizing-Operators on Polyhedra via Parametric Linear Programming*

Alexandre Maréchal, David Monniaux, and Michaël Périn

Univ. Grenoble Alpes, CNRS, Grenoble INP***, VERIMAG, 38000 Grenoble, France
{alex.marechal, david.monniaux, michael.perin}@univ-grenoble-alpes.fr

Abstract. Convex polyhedra capture linear relations between variables. They are used in static analysis and optimizing compilation. Their high expressiveness is however barely used in verification because of their cost, often prohibitive as the number of variables involved increases. Our goal in this article is to lower this cost.
Whatever the chosen representation of polyhedra – as constraints, as generators or as both – expensive operations are unavoidable. That cost is mostly due to four operations: conversion between representations, based on Chernikova's algorithm, for libraries in double description; convex hull, projection and minimization, in the constraints-only representation of polyhedra.
Libraries operating over generators incur exponential costs on cases common in program analysis. In the Verimag Polyhedra Library this cost was avoided by a constraints-only representation and reducing all operations to variable projection, classically done by Fourier-Motzkin elimination. Since Fourier-Motzkin generates many redundant constraints, minimization was however very expensive.
In this article, we avoid this pitfall by expressing projection as a parametric linear programming problem. This dramatically improves efficiency, mainly because it avoids the post-processing minimization.
We show how our new approach can be up to orders of magnitude faster than the previous approach implemented in the Verimag Polyhedra Library that uses only constraints and Fourier-Motzkin elimination, and on par with the conventional double description approach, as implemented in well-known libraries.

Keywords: Polyhedra, Parametric Linear Programming, Projection

1 The Challenge of Verification using Polyhedra

Static analyzers establish the validity of assertions in programs by discovering inductive invariants that entail them. Analyzers based on abstract interpretation consider invariants within an *abstract domain* [5]. Invariants on numeric

*** Institute of Engineering Univ. Grenoble Alpes

* This work was partially supported by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement nr. 306595 "STATOR".

variables are of particular interest. They may entail that software produce no arithmetic overflow, no array index out of bounds — the user may be directly interested in such properties, or an optimizing compiler may discard runtime checks for violations that cannot occur. Furthermore, proofs of more complicated properties may use numerical invariants internally — for instance proofs of sorting algorithms need invariants on indices.

An example of abstract domain suitable for program states given by vectors of n numerical variables is the domain of products of n intervals; but such a domain cannot express relationships between variables. This hinders verification even if the final goal is to prove that a given variable lies within certain bounds, for instance to prove that a string length is less than a fixed buffer size: one may need to prove that the sum of the length of two strings is less than this size, thus a relation between these lengths.

The domain of *convex polyhedra* comprises sets of states defined by conjunctions of linear (in)equalities over the variables [6]. The analyzer needs to perform a variety of operations on these sets — least upper bound (convex hull, in the case of polyhedra), inclusion tests, projections, image and reverse image by program operations; also, in some cases, intersections and Minkowski sums. In addition to static analyzers, convex polyhedra are used inside highly optimizing compilers to reorganize loop nests [1].

Despite their expressiveness and 40 years of research, polyhedra are little used in verification because operations on polyhedra are still costly and do not scale to large programs [13]. Usually, they are restricted to a small subset of program variables such as loop indices [14] — including more variables would mean skyrocketing costs.

Most libraries for computing over convex polyhedra maintain a *double description*, both as *generators* (vertices, in the case of bounded polyhedra) or *constraints* (faces). A common case in program analysis is upper and lower bounds are known on all N variables — that is, the vector of variables lies within a distorted N-dimensional hypercube, which has 2^N vertices. This explains the reputation of polyhedra as unwieldy except in very low dimension, and motivated the design of the Verimag Verified Polyhedra Library (Vpl) that operates on constraints-only representations [8,10]. An advantage of that approach is that it is easy to log enough information to independently check that the computed polyhedron includes the exact polyhedron that should be computed, which suffices for proving that static analysis is sound [9,10]; the certificate checker was implemented and proved correct in Coq.[1] The consequence is that many operations of the Vpl, such as assignment, convex hull or Minkowski sum, were encoded as projection, finally performed by Fourier-Motzkin elimination [2]. Unfortunately, Fourier-Motzkin elimination generates numerous redundant constraints; and even by incrementally removing them after each elimination of a variable, intermediate steps may create large lists of constraints.

[1] Certifying a library in double description would have likely entailed implementing and proving in Coq the correctness of Chernikova's conversion algorithm from one representation to the other.

In 2013, the overall performance of VPL [10] on typical verification benchmarks was on a par with that of double description libraries, though the timings on individual operations differed: some operations are faster than in double description, some are slower — all those involving projection, including convex hull. Projection by Fourier-Motzkin was the bottleneck.

Contribution In this article we report on an algorithmic breakthrough that speeds up typical computations on polyhedra in constraints-only representation by several orders of magnitude when polyhedra becomes large (in number of relations) or dense (in number of variables involved in each relations). Scalability results from the inseparable combination of i) the formulation of the projection via Parametric Linear Programming (PLP) (§3); ii) the implementation of a PLP-solver over rationals, to get exact results (§5); iii) a new normalization criterion, which ensures the absence of redundant constraints and saves the post-processing elimination of redundancy (§6). This normalization, its proof and an certifying implementation are the main contributions of this paper.[2]

We demonstrate the scalability by comparing timings of projections between the PLP-based algorithm, Fourier-Motzkin elimination and an existing library based on double description (§7).

Related work The high cost of general convex polyhedra was long deplored. It motivated studying restricted classes of polyhedra, with simpler and faster algorithms, such as *octagons* [25]; and even these were found to be too slow, motivating recent algorithmic improvements [31]. We instead sought to conserve the domain of polyhedra as originally described [6,12], but with very different algorithms.

Our work was inspired by Howe *et al.*'s attempt to replace the Fourier-Motzkin elimination by a formulation as a Parametric Linear Optimization Problem (PLOP) [15], which they solved by an ad hoc algorithm. Unfortunately, their implementation is not available. We took a step further and developed a generic PLP-solver exploiting insights by [18,17]. Our solver, implemented in OCAML, works over rationals and generates COQ-certificates of correctness of its computations, similar to those in VPL [8,9,10].

Most libraries for computing over convex polyhedra for static analysis or compilation, including PolyLib,[3] Komei Fukuda's CDD,[4] the Parma Polyhedra Library,[5] the NEWPOLKA library included in Apron,[6] operate over the double description; see e.g. [27] for an introduction. The costliest and most complicated operation is the conversion from one representation to the other, using Chernikova's algorithm [3,19]. It is rather easy to prune redundant items from one representation if one has the other, which explains the attractiveness of that

[2] The VPL 0.2 is available at `https://github.com/VERIMAG-Polyhedra/VPL`
[3] `https://icps.u-strasbg.fr/polylib/`
[4] `https://www.inf.ethz.ch/personal/fukudak/cdd_home/`
[5] `http://bugseng.com/products/ppl/` [27]
[6] `http://apron.cri.ensmp.fr/library/` [16]

approach. Its only drawback is that, as explained above, the generator representation is exponential in the dimension on very common and simple cases.

The explosive nature of the generator representation motivated approaches that detect when a polyhedron is a Cartesian product of polyhedra and compute generator representations separately for each element of the product, thereby avoiding exponential blowup in the case of the hypercube [13,30].

General texts on polyhedra and linear programming include [7,4,28].

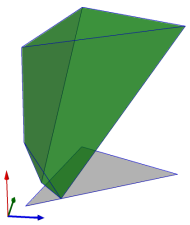
2 Basics

Throughout the article, vectors are written in boldface lowercase, and matrices in boldface uppercase. An affine form over $\boldsymbol{x}$ is a linear combination plus constant of $x_1, ..., x_n$. For two vectors $\boldsymbol{a_\ell}$ and $\boldsymbol{x}$ of the same length, the dot product $\langle \boldsymbol{a_\ell}, \boldsymbol{x} \rangle = \sum_i a_{\ell i}\, x_i$ is a linear function of $\boldsymbol{x}$. Thus, we often use the notation $\boldsymbol{a_\ell}(\boldsymbol{x})$ instead of $\langle \boldsymbol{a_\ell}, \boldsymbol{x} \rangle$.

A *convex polyhedron*[7] is the set of points $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Q}^n$ that satisfy a conjunction (or a set) of linear constraints of the form $\boldsymbol{C_\ell} : \sum_{i=1}^{n} a_{\ell i}\, x_i \bowtie b_\ell$ where x_i are program variables, $a_{\ell i}$ and b_ℓ are constants in $\mathbb{Q}$, and $\bowtie \in \{ \leq, =, \geq \}$. All constraints can be assumed to use only $\geq$.[8] Such a constraint is the ℓ^{th} row of a vector inequality $\boldsymbol{Ax} \geq \boldsymbol{b}$. We use $[\![\mathcal{P}]\!]$ to specifically refer to the set of points defined by the set of constraints $\mathcal{P}$. Given a polyhedron $[\![\mathcal{P}]\!] = \{ \boldsymbol{x} \mid \boldsymbol{Ax} \geq \boldsymbol{b} \}$, the same system with strict inequalities defines $\mathring{\mathcal{P}}$, *the interior of* $\mathcal{P}$, and $\mathring{x}$ denotes a point of $[\![\mathring{\mathcal{P}}]\!] \stackrel{def}{=} \{ \boldsymbol{x} \mid \boldsymbol{Ax} > \boldsymbol{b} \}$. In all the paper and without loss of generality, we focus on polyhedra with non-empty interior, meaning that equalities (explicit or implicit) are extracted and treated separately, as in most polyhedra libraries.

Before presenting our encoding of the projection operator as a PLOP, we start by recalling the fundamental Farkas' Lemma and Fourier-Motzkin's Algorithm for variable elimination.

Example 1.1. The right-hand side figure shows the geometrical space defined by the polyhedron $\mathcal{P} = \{\boldsymbol{C_1} : -x_1 - 2x_2 + 2x_3 \geq -7, \boldsymbol{C_2} : -x_1 + 2x_2 \geq 1, \boldsymbol{C_3} : 3x_1 - x_2 \geq 0, \boldsymbol{C_4} : -x_3 \geq -10, \boldsymbol{C_5} : x_1 + x_2 + x_3 \geq 5\}$ and its projection on dimensions (x_1, x_2) resulting from the elimination of variable x_3. Projecting variable x_3 from $\mathcal{P}$ – noted $\mathcal{P}_{\backslash \{ x_3 \}}$ – by Fourier-Motzkin elimination consists in eliminating x_3 by combining constraints with opposite signs for x_3. Constraints that do not involve x_3 remain unchanged. This retains constraints $\boldsymbol{C_2}, \boldsymbol{C_3}$ and produces two new constraints: $\boldsymbol{C_1} + 2\boldsymbol{C_4} : -x_1 - 2x_2 \geq -27$ and



[7] We only deal with convex polyhedra. For readability, we will omit the adjective *convex* in the following.

[8] An equality $a = b$ corresponds to the conjunction of inequalities $a \geq b \wedge a \leq b$ and $a \leq b$ is equivalent to $-a \geq -b$.

$\boldsymbol{C_4} + \boldsymbol{C_5} : x_1 + x_2 \geq -5$. By Farkas' Lemma, the latter is redundant with respect to $\boldsymbol{C_2}$ and $\boldsymbol{C_3}$ as it can be expressed as a nonnegative combination of $\boldsymbol{C_2}$, $\boldsymbol{C_3}$.

Lemma 1 (Farkas' lemma [28, 7.1h, p.93]). *A constraint $\boldsymbol{C}'$ is a logical consequence of a non-contradictory set of constraints $\mathcal{P} = \{\boldsymbol{C_1}, \ldots, \boldsymbol{C_p}\}$ iff there exists $\lambda_0, ..., \lambda_p \geq 0$ such that $\boldsymbol{C}' = \lambda_0 + \sum_{i=1}^{p} \lambda_i \boldsymbol{C_i}$, called a* Farkas decomposition *of $\boldsymbol{C}'$ on $\mathcal{P}$.*

Example 1.2. The combination $\frac{4}{5}\boldsymbol{C_2} + \frac{3}{5}\boldsymbol{C_3} : x_1 + x_2 \geq \frac{4}{5}$ is a logical consequence of $\boldsymbol{C_2}$ and $\boldsymbol{C_3}$ and it is a stronger condition than $\boldsymbol{C_4} + \boldsymbol{C_5} : x_1 + x_2 \geq -5$ since $\frac{4}{5} > -5$. Thus, the constraint $\boldsymbol{C_4} + \boldsymbol{C_5}$ is redundant with respect to $\boldsymbol{C_2}$ and $\boldsymbol{C_3}$. Therefore the polyhedron $\mathcal{P}_{\setminus\{x_3\}}$ is only formed of three constraints $\{\boldsymbol{C_2}, \boldsymbol{C_3}, \boldsymbol{C_1} + 2\boldsymbol{C_4}\}$.

3 Projection via Parametric Linear Programming

Naive Fourier-Motzkin elimination produces $O\big((\frac{|\mathcal{P}|}{2})^{2^k}\big)$ constraints when eliminating k variables of a polyhedron with $|\mathcal{P}|$ constraints [29]. Most of them are redundant: indeed, the number of faces of the projected polyhedron is $O(|\mathcal{P}|^k)$ [26, §4.1].[9] Removing the redundant constraints is costly, even though there exists improved algorithms [20].

Jones *et al.* [17] then Howe *et al.* [15] noticed that the projection of a polyhedron can be expressed as a Parametric Linear Programming problem. In fact, PLP naturally arises when trying to generalize Fourier-Motzkin method to eliminate several variables simultaneously. In this article we achieve the work initiated by [15], whose goal was to compute the projected polyhedron without generating redundant constraints. Let us first explain their approach.

Example 1.3. As a consequence of Farkas lemma, any constraint implied by $\{\boldsymbol{C_1}, ..., \boldsymbol{C_5}\}$ is a nonnegative combination of them, written $\lambda_0 + \sum_{i=1}^{5} \lambda_i \boldsymbol{C_i}$ with $\lambda_i \geq 0$, *i.e.*

$$\begin{aligned} &\lambda_0 + \lambda_1(-x_1 - 2x_2 + 2x_3) + \lambda_2(-x_1 + 2x_2) + \lambda_3(3x_1 - x_2) \\ &+ \lambda_4(-x_3) + \lambda_5(x_1 + x_2 + x_3) \quad \geq \quad -7\lambda_1 + \lambda_2 - 10\lambda_4 + 5\lambda_5 \end{aligned}$$

The left-hand side of the inequality can be rearranged to reveal the coefficient of each variable x_i and we can bring the right-hand side term of $\geq$ to the left.

$$\begin{aligned} \lambda_0 + (-\lambda_1 - \lambda_2 + 3\lambda_3 + \lambda_5)\, x_1 + (-2\lambda_1 + 2\lambda_2 - \lambda_3 + \lambda_5)\, x_2 \\ + (2\lambda_1 - \lambda_4 + \lambda_5)\, x_3 - (-7\lambda_1 + \lambda_2 - 10\lambda_4 + 5\lambda_5) \geq 0 \end{aligned} \tag{1}$$

Then, any instantiation of that inequality with λ_i canceling the coefficient of x_3, *i.e.* that satisfies (α) $2\lambda_1 - \lambda_4 + \lambda_5 = 0$, is an over-approximation of $\mathcal{P}_{\setminus\{x_3\}}$.

[9] This follows from McMullen's bound on the number of $n - k - 1$-faces of the polyhedron [23,24].

Indeed, it does not involve x_3 and, as a Farkas combination, it is by construction a logical consequence of $\mathcal{P}$. Constraints found by the FM elimination of x_3 correspond to the solutions $(\lambda_0, \ldots, \lambda_5) \in \{(0,0,1,0,0,0), (0,0,0,1,0,0), (0,1,0,0,2,0), (0,0,0,0,1,1)\}$ of Equation $(\mathcal{O}\alpha)$. Note that it is possible to eliminate several variables simultaneously by setting an elimination equation for each variable that must be discarded.

Here is a first formulation of a projection as a PLOP. We will refine it later, as it is not sufficient to avoid redundancies in the result. Given a polyhedron $\mathcal{P} = \{ \boldsymbol{C_1} : \boldsymbol{a_1}(\boldsymbol{x}) \geq b_1, \ldots, \boldsymbol{C_p} : \boldsymbol{a_p}(\boldsymbol{x}) \geq b_p \}$ on variables $x_1, \ldots, x_n$, the projection of $\mathcal{P}$ by elimination of k variables $x_{e_1}, \ldots, x_{e_k}$ can be obtained as the solution of the optimization problem:

$$\left.\begin{array}{ll} \textbf{minimize the objective function} & \mathcal{Z}(\boldsymbol{x}) = \lambda_0 + \sum_{i=1}^{p} \lambda_i \times (\boldsymbol{a_i}(\boldsymbol{x}) - b_i) \\ \textbf{under the constraints} & (F)\ \lambda_0 \geq 0, ..., \lambda_p \geq 0 \\ & (\dagger)\ \sum_{i=0}^{p} \lambda_i = 1 \\ & (\mathcal{O}\alpha)\ \boldsymbol{\alpha_{e_1}}(\boldsymbol{\lambda}) = 0, ..., \boldsymbol{\alpha_{e_k}}(\boldsymbol{\lambda}) = 0 \end{array}\right\} (2)$$

where $\boldsymbol{\alpha_i}(\boldsymbol{\lambda})$ denotes the coefficient of x_i in the reformulation of the objective as $\boldsymbol{\alpha_1}(\boldsymbol{\lambda}) \times x_1 + \ldots + \boldsymbol{\alpha_n}(\boldsymbol{\lambda}) \times x_n + \boldsymbol{\alpha_0}(\boldsymbol{\lambda})$. The unknowns λ_i are called the *decision variables*: the solver must find a solution for them. Note the inequalities (F) from Farkas' Lemma in addition to the $(\mathcal{O}\alpha)$ equations defining a projection. This problem has a *parametric objective*: the objective function depends on parameters $x_1, \ldots, x_n$ due to the terms $\boldsymbol{a_i}(\boldsymbol{x})$ in the coefficients of the decision variables. But once $x_1, \ldots, x_n$ are fixed, both the objective function and the constraints become linear in the decision variables, thus this problem belongs to *parametric linear programming*.

An additional constraint, here $\sum_i \lambda_i = 1$, is needed to prevent the solver from obtaining the optimal solution $\boldsymbol{\lambda} = \boldsymbol{0}$ which is always valid in a projection problem, whatever the parameter values. The $(\dagger)$ condition only excludes this useless null solution because any other solution can be scaled so that $\sum_i \lambda_i = 1$. The presence of λ_0 in the objective can seem useless and strange to readers who are familiar with linear programming: the solution $\lambda_0 = 1$ and $\lambda_1 = ... = \lambda_p = 0$ becomes feasible and generates a trivially redundant constraint $\boldsymbol{C_{triv}} : 1 \geq 0$. The role of λ_0 will become clear in §4 and §6.

Example 1.4. The elimination of x_3 via PLP is defined by two matrices: $\boldsymbol{O}$ is built from $[-\boldsymbol{b}|\boldsymbol{A}]^\intercal$ and encodes the objective. The other one captures the requirement $(\mathcal{O}\alpha)$ and $(\dagger)$. As usual in solvers, Farkas constraints (F) are left implicit.

$$\left.\begin{array}{l}\textbf{minimize the objective function} \\ (1, x_1, x_2, x_3)^\intercal \overbrace{\underbrace{\begin{pmatrix} 0 & 1 & 7 & -1 & 0 & 10 & -5 \\ 0 & 0 & -1 & -1 & 3 & 0 & 1 \\ 0 & 0 & -2 & 2 & -1 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 & -1 & 1 \end{pmatrix}}_{[-\boldsymbol{b}|\boldsymbol{A}]^\intercal}}^{\boldsymbol{O}} \begin{pmatrix} 1 \\ \lambda_0 \\ \vdots \\ \lambda_5 \end{pmatrix} = \mathcal{Z}(\boldsymbol{x}) \\ \textbf{under the constraints} \quad \underbrace{\overbrace{\left(\begin{array}{ccccccc} -1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 2 & 0 & 0 & -1 & 1 \end{array}\right)}^{(\dagger)}}_{\boldsymbol{\alpha}} \begin{pmatrix} 1 \\ \lambda_0 \\ \vdots \\ \lambda_5 \end{pmatrix} = \mathbf{0} \end{array}\right\} \quad (3)$$

This formulation of the projection is correct. Unfortunately, it may still generate redundant constraints: the solutions $(\lambda_0, \ldots, \lambda_5) \in \{$(1,0,0,0,0,0), (0,0,1,0,0,0), (0,0,0,1,0,0), $(0, \frac{1}{3}, 0, 0, \frac{2}{3}, 0)$, $(0, 0, 0, 0, \frac{1}{2}, \frac{1}{2})\}$ include the trivial constraint $1 \geq 0$ and $\frac{1}{2}\boldsymbol{C_4} + \frac{1}{2}\boldsymbol{C_5}$ which is equivalent to the redundant constraint $\boldsymbol{C_4} + \boldsymbol{C_5}$ found by Fourier-Motzkin elimination. We shall address this point in §6.

4 Polyhedra as Solutions of Parametric Linear Optimization Problems

In the previous section we encoded the projection of a polyhedron as a PLOP. For interpreting the result of a PLP-solver as a polyhedron we need to go one step further into the field of PLP and look at the solutions of a PLOP.

To summarize, Parametric Linear Programming is an extension of Linear Programming where the constants in the constraints or the coefficients in the objective function may be replaced by affine combinations of parameters [11]. In this article, we only deal with the case where parameters appear in the objective function. The general form of a PLOP that stems from projection is

$$\left.\begin{array}{l}\textbf{minimize the objective function } \mathcal{Z}(\boldsymbol{x}) \stackrel{def}{=} \lambda_0 + \sum_{i=1}^{p} \lambda_i \times (\boldsymbol{a_i}(\boldsymbol{x}) - b_i) \\ \textbf{under the constraints } \lambda_0, ..., \lambda_p \geq 0,\ (\dagger)\ \sum_{i=0}^{p} \lambda_i = 1,\ \boldsymbol{\alpha\lambda} = \mathbf{0} \end{array}\right\} \quad (4)$$

where $\boldsymbol{x}$ is the vector of parameters $(x_1, ..., x_n)$; $(\boldsymbol{a_i}(\boldsymbol{x}) - b_i)$ are *affine forms on the parameters*; and $\boldsymbol{\alpha}$ is a matrix. In a projection problem the system of equations $\boldsymbol{\alpha\lambda} = \mathbf{0}$ constrains the *decision variables* $\lambda_1, ..., \lambda_p$ but not λ_0.

The solution is a concave, piecewise affine function $\mathcal{Z}^\star$, mapping the parameters to the optimal solution:

$$\mathcal{Z}^\star \stackrel{def}{=} \boldsymbol{x} \mapsto \begin{cases} \mathcal{Z}_1^\star(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}_1 \\ \vdots & \\ \mathcal{Z}_r^\star(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \mathcal{R}_r \end{cases} \quad (5)$$

Each piece is an affine form over $\boldsymbol{x}$, obtained by instantiating the objective function $\mathcal{Z}$ with a solution $\boldsymbol{\lambda}$; a piece can also be denoted by $\mathcal{Z}^\star_{\boldsymbol{\lambda}}$. Each $\mathcal{Z}^\star_i$ is associated to a *region of optimality* $\mathcal{R}_i$ that designates the set of $\boldsymbol{x}$ for which the minimum of $\mathcal{Z}^\star(\boldsymbol{x})$ is $\mathcal{Z}^\star_i(\boldsymbol{x})$. Regions of optimality are polyhedra; that will be clear in §5 when we will explain how they are computed by our solver (see Example 1.6.). They form a *quasi-partition* of the space of parameters: their union covers $\mathbb{Q}^n$ and the intersection of *the interior* of two distinct regions is empty. They however do not form a partition because two regions $\mathcal{R}_i, \mathcal{R}_j$ may overlap on their frontiers; then, their solutions $\mathcal{Z}^\star_i, \mathcal{Z}^\star_j$ coincide on the intersection.

From optimal function to polyhedron. A PLOP can be thought of as a *declarative description of the projection operator*. The solution $\mathcal{Z}^\star$ can be interpreted as a polyhedron $\mathcal{P}^\star$ that is the projection of an input polyhedron $\mathcal{P}$. This requires some explanations:

- Due to the Farkas conditions $\lambda_0, ..., \lambda_p \geq 0$ which preserve the direction of inequalities, the objective function of PLOP (4), *i.e.* $\lambda_0 + \sum_{i=1}^{p} \lambda_i \times (\boldsymbol{a_i}(\boldsymbol{x}) - b_i)$ can be interpreted as a constraint implied by the input polyhedron $\mathcal{P} = \{\boldsymbol{C_1} : \boldsymbol{a_1}(\boldsymbol{x}) \geq b_1, ..., \boldsymbol{C_p} : \boldsymbol{a_p}(\boldsymbol{x}) \geq b_p\}$. Actually, for a given $\boldsymbol{\lambda}$, the statement $\mathcal{Z}^\star_{\boldsymbol{\lambda}}(\boldsymbol{x}) \geq 0$ is equivalent to the constraint

$$\lambda_0 + \sum_{i=1}^{p} \lambda_i \times \boldsymbol{a_i}(\boldsymbol{x}) \geq \sum_{i=1}^{p} \lambda_i \times b_i \tag{6}$$

- Minimizing the objective ensures that the λ_0-shift of the constraint will be minimal, meaning that the constraint $\mathcal{Z}^\star_{\boldsymbol{\lambda}}(\boldsymbol{x}) \geq 0$ will be tightly adjusted.
- The requirement $\boldsymbol{\alpha\lambda} = \boldsymbol{0}$ captures the expected effect of the projection. Thus, any solution $\boldsymbol{\lambda}$ defines a constraint $\mathcal{Z}_{\boldsymbol{\lambda}}(\boldsymbol{x}) \geq 0$ of the polyhedron $\mathcal{P}^\star$.

Now recall that a polyhedron is a set of points that satisfy linear inequalities. Therefore, it is natural to define $[\![\mathcal{P}^\star]\!]$ as $\{\, \boldsymbol{x} \mid \mathcal{Z}^\star(\boldsymbol{x}) \geq 0 \,\}$. The following lemma proves that this set of points is a polyhedron.

Lemma 2. $\{\, \boldsymbol{x} \mid \mathcal{Z}^\star(\boldsymbol{x}) \geq 0 \,\} = \bigcap_{k=1}^{r} \{\, \boldsymbol{x} \mid \mathcal{Z}^\star_k(\boldsymbol{x}) \geq 0 \,\}$

Proof. Let us prove the mutual inclusion.

(⊆) Pick up a point $\boldsymbol{x'} \in \{\, \boldsymbol{x} \mid \mathcal{Z}^\star(\boldsymbol{x}) \geq 0 \,\}$. By definition of $\mathcal{Z}^\star$ as a piecewise function defined on the whole space of parameters, then there exists i such that $\boldsymbol{x'} \in \mathcal{R}_i$ and $\mathcal{Z}^\star(\boldsymbol{x'}) = \mathcal{Z}^\star_i(\boldsymbol{x'})$. It follows that $\mathcal{Z}^\star_i(\boldsymbol{x'}) \geq 0$ since $\boldsymbol{x'}$ belongs to the set of points where $\mathcal{Z}^\star$ is non-negative. Moreover, the fact that $\boldsymbol{x'}$ belongs to $\mathcal{R}_i$ – the region of optimality of $\mathcal{Z}^\star_i$ in a minimization problem – ensures that $\mathcal{Z}^\star_k(\boldsymbol{x'}) \geq \mathcal{Z}^\star_i(\boldsymbol{x'})$ for all k and therefore, $\mathcal{Z}^\star_k(\boldsymbol{x'}) \geq 0$ for all k. Thus, $\boldsymbol{x'} \in \{\, \boldsymbol{x} \mid \mathcal{Z}^\star_k(\boldsymbol{x}) \geq 0 \,\}$ for all $k = 1..r$. Finally, $\boldsymbol{x'} \in \bigcap_{k=1..r} \{\, \boldsymbol{x} \mid \mathcal{Z}^\star_k(\boldsymbol{x}) \geq 0 \,\}$.

($\supseteq$) Pick up a point $\boldsymbol{x'} \in \bigcap_{k=1}^{r} \{\, \boldsymbol{x} \mid \mathcal{Z}_k^\star(\boldsymbol{x}) \geq 0 \,\}$. Then, $\boldsymbol{x'}$ belongs to a least one $\mathcal{R}_i$ because the regions form a (pseudo) partition of the whole space of parameters $\mathbb{Q}^n$, thus $\bigcup_{k=1}^{r} \mathcal{R}_k = \mathbb{Q}^n$. Yet, the affine piece that defines $\mathcal{Z}^\star$ on $\boldsymbol{x'}$ is $\mathcal{Z}_i^\star$ and $\mathcal{Z}^\star(\boldsymbol{x'}) = \mathcal{Z}_i^\star(\boldsymbol{x'})$. Moreover, all the affine pieces of $\mathcal{Z}^\star$ are nonnegative on $\boldsymbol{x'}$ since $\boldsymbol{x'} \in \bigcap_{k=1}^{r} \{\, \boldsymbol{x} \mid \mathcal{Z}_k^\star(\boldsymbol{x}) \geq 0 \,\}$. Then, in particular $\mathcal{Z}_i^\star(\boldsymbol{x'}) \geq 0$ and the same goes for $\mathcal{Z}^\star(\boldsymbol{x'})$. Finally, $\boldsymbol{x'} \in \{\, \boldsymbol{x} \mid \mathcal{Z}^\star(\boldsymbol{x}) \geq 0 \,\}$.

□

Constructing the vector inequality $\boldsymbol{Z^\star}\ \boldsymbol{x} \geq \boldsymbol{b^\star}$ that defines the polyhedron $\mathcal{P}^\star$ is straightforward from the solution $\mathcal{Z}^\star$. If suffices to get rid of the regions of optimality and to interpret each affine piece of $\mathcal{Z}^\star$ as an inequality: $\{\, \boldsymbol{x} \mid \mathcal{Z}^\star(\boldsymbol{x}) \geq 0 \,\}$ = (by Lemma 2) $\bigcap_{k=1}^{r} \{\, \boldsymbol{x} \mid \mathcal{Z}_k^\star(\boldsymbol{x}) \geq 0 \,\} = \{\boldsymbol{x} \mid \bigwedge_{k=1}^{r} \mathcal{Z}_k^\star(\boldsymbol{x}) \geq 0\}$ $= \{\boldsymbol{x} \mid \bigwedge_{k=1}^{r} \langle \boldsymbol{z_k^\star}, \boldsymbol{x} \rangle - b_k^\star \geq 0\} = \{\, \boldsymbol{x} \mid \boldsymbol{Z^\star x} \geq \boldsymbol{b^\star} \,\}$. Let us detail this construction.

Each piece $\mathcal{Z}_k^\star$ of the solution is a affine form over $\boldsymbol{x}$ and $\mathcal{Z}_k^\star(\boldsymbol{x}) \geq 0$ defines a constraint in the form (6) which can be written $\sum_{i=1}^{n} z_{ki}^\star\, x_i \geq b_k^\star$ *i.e.* $\langle \boldsymbol{z_k^\star}, \boldsymbol{x} \rangle \geq b_k^\star$ for some vector $\boldsymbol{z_k^\star} = (z_{k1}^\star, ..., z_{kn}^\star)$ and some constant $b_k^\star$. It follows from Lemma 2 that the set of points $\boldsymbol{x}$ where $\mathcal{Z}^\star(\boldsymbol{x})$ is nonnegative is a polyhedron defined by the vector inequality $\boldsymbol{Z^\star x} \geq \boldsymbol{b^\star}$ where the rows of $\boldsymbol{Z^\star}$ are the vectors $\boldsymbol{z_1^\star}, \ldots, \boldsymbol{z_r^\star}$ and $\boldsymbol{b^\star}$ is the column vector $(b_1^\star, \ldots, b_r^\star)^\intercal$.

Example 1.5. On our running projection problem, the PLP-solver returns the following optimal function, and the instantiation of the decision variables λ_i that defines each affine piece:

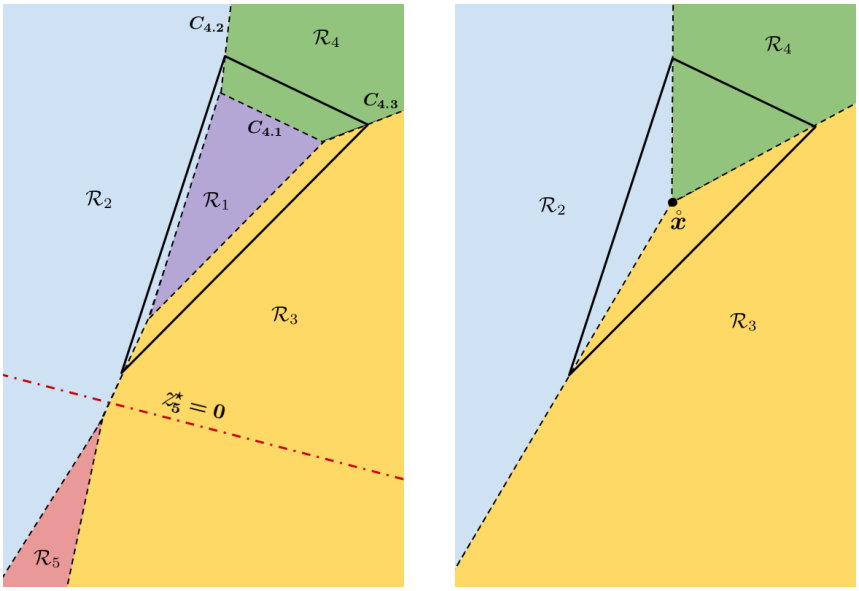
$$\mathcal{Z}^\star \stackrel{def}{=} (x_1, x_2) \mapsto \begin{cases} \mathcal{Z}_2^\star : & -x_1 & +2x_2 & -1 & \text{on} & \mathcal{R}_2 & (\text{for } \lambda_2 = 1) \\ \mathcal{Z}_3^\star : & 3x_1 & -x_2 & & \text{on} & \mathcal{R}_3 & (\text{for } \lambda_3 = 1) \\ \mathcal{Z}_4^\star : & -\frac{1}{3}x_1 & -\frac{2}{3}x_2 & +9 & \text{on} & \mathcal{R}_4 & (\text{for } \lambda_1 = \frac{1}{3},\ \lambda_4 = \frac{2}{3}) \\ \mathcal{Z}_5^\star : & \frac{1}{2}x_1 & +\frac{1}{2}x_2 & +\frac{5}{2} & \text{on} & \mathcal{R}_5 & (\text{for } \lambda_4 = \frac{1}{2},\ \lambda_5 = \frac{1}{2}) \\ \mathcal{Z}_1^\star : & & & 1 & \text{on} & \mathcal{R}_1 & (\text{for } \lambda_0 = 1) \end{cases}$$

from which we construct the polyhedron

$$\mathcal{P}^\star = \overbrace{\begin{pmatrix} -1 & 2 & 0 \\ 3 & -1 & 0 \\ -\frac{1}{3} & -\frac{2}{3} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 \end{pmatrix}}^{\boldsymbol{Z^\star}} \overbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}^{\boldsymbol{x}} \geq \overbrace{\begin{pmatrix} 1 \\ 0 \\ -9 \\ -\frac{5}{2} \\ -1 \end{pmatrix}}^{\boldsymbol{b^\star}} = \left\{ \begin{array}{rrl} \boldsymbol{C_2} : & -x_1 + 2x_2 \geq & 1 \\ \boldsymbol{C_3} : & 3x_1 - x_2 \geq & 0 \\ \frac{1}{3}\boldsymbol{C_1} + \frac{2}{3}\boldsymbol{C_4} : & -\frac{1}{3}x_1 - \frac{2}{3}x_2 \geq & -9 \\ \frac{1}{2}\boldsymbol{C_4} + \frac{1}{2}\boldsymbol{C_5} : & \frac{1}{2}x_1 + \frac{1}{2}x_2 \geq & -\frac{5}{2} \\ \boldsymbol{C_{triv}} : & 0 \geq & -1 \end{array} \right\}$$

Variable x_3 does not appear anymore in the constraints of $\mathcal{P}^\star$ because its column in $\boldsymbol{Z^\star}$ is made of 0. The regions of optimality, shown on Fig.1(a) form a pseudo-partition of the whole space of parameters (x_1, x_2): regions $\mathcal{R}_2, \ldots, \mathcal{R}_5$ are unbounded; the central triangle is the region $\mathcal{R}_1$ associated to the constant

affine form $\mathcal{Z}_1^\star = 1$ which produces the trivial constraint $\boldsymbol{C_{triv}} : 1 \geq 0$. Each boundary of $\mathcal{P}^\star$ (shown as bold lines in the figure) is the intersection of a region of optimality $\mathcal{R}_i$ with the space where the associated affine form $\mathcal{Z}_i^\star$ evaluates to zero. We retrieve constraints equivalent to those of Example 1.1., except that the redundant constraint $\frac{1}{2}\boldsymbol{C_4} + \frac{1}{5}\boldsymbol{C_5}$ generated by $\mathcal{Z}_5^\star$ is still present. The drawing of the regions reveals that $\mathcal{Z}_5^\star$ does not vanish on its region of optimality, *i.e.* $[\![\mathcal{Z}_5^\star = 0]\!] \cap [\![\mathring{\mathcal{R}}_5]\!] = \emptyset$. Actually, this is true for any redundant constraint. Indeed, we will prove in §6 (Lemma 5) that $[\![\mathcal{Z}_i^\star = 0]\!] \cap [\![\mathring{\mathcal{R}}_i]\!] \neq \emptyset$ ensures the irredundancy of the constraint $\mathcal{Z}_i^\star \geq 0$ in $\mathcal{P}^\star$.



(*a*) (*b*)

Fig. 1. The regions of optimality of the solution $\mathcal{Z}^\star$ of Example 1.5. obtained by solving PLOP (3). The bold lines are the boundaries of the projected polyhedron $\mathcal{P}^\star$. Figure (*a*) shows regions obtained when the PLOP contains the constraint $\sum_i \lambda_i = 1$. Figure (*b*) shows regions obtained when constraints are normalized on point $\hat{\boldsymbol{x}}$ (see §6).

5 Principle of a PLP-solver

Due to space limitations we shall only sketch how our parametric linear programming solver works. It is based on a recent algorithm by Jones et al. [18] with some improvements: it uses a fast simplification of regions [22] and performs exact computations in rationals so as to avoid rounding errors.

This algorithm for solving a PLOP is a generalization of the simplex algorithm which can itself be seen as an extension of Gaussian elimination for solving a system of linear equations.

Gaussian elimination proceeds by rewriting: each equation defines a variable in terms of the other ones. This equation can be used to eliminate the variable from the other equations by substitution. This operation is called a pivot. Gauss pivoting strategy leads to an equivalent system in echelon form where un/satisfiability becomes trivial.

The simplex algorithm follows the same principle but differs in the selection of the variable to eliminate. First, each inequality $C_\ell : \sum_{i=1}^{n} a_{\ell i}\, x_i \leq b_\ell$ is changed into an equality $\sum_{i=1}^{n} a_{\ell i}\, x_i + x_{n+\ell} = b_\ell$ by introducing a variable $x_{n+\ell} \geq 0$ called a *slack variable*. Second, the objective function is added to the system as an extra equation defining the variable $\mathcal{Z}$ as a linear form $\mathcal{Z} = \sum_{i=1}^{n+r} o_i x_i$. Then, the simplex performs pivots as in Gaussian elimination until reaching an equivalent system of equations where the optimality of $\mathcal{Z}$ becomes syntactically obvious. Let us take an example.

Example 1.6. To illustrate the behavior of a LP-solver, such as the simplex, let us instantiate the objective of PLOP (3), e.g. with $x_1 = 5, x_2 = 11, x_3 = 1$, to obtain a non-parametric version: $\mathcal{Z} = \lambda_0 - 18\,\lambda_1 + 16\,\lambda_2 + 4\,\lambda_3 + 9\,\lambda_4 + 12\,\lambda_5$. The simplex strategy chooses to define λ_1 and λ_4 in terms of the other decision variables. It exploits the equations (†) and ($\mathcal{O}$) of PLOP (3) and gets (i) $\lambda_1 = -\frac{1}{3}\,\lambda_0 - \frac{1}{3}\,\lambda_2 - \frac{1}{3}\,\lambda_3 - \frac{2}{3}\,\lambda_5 + \frac{1}{3}$ using ($\mathcal{O}$) to eliminate λ_4 in (†), and (ii) $\lambda_4 = -\frac{2}{3}\,\lambda_0 - \frac{2}{3}\,\lambda_2 - \frac{2}{3}\,\lambda_3 - \frac{1}{3}\,\lambda_5 + \frac{2}{3}$ using (†) to eliminate λ_1 in ($\mathcal{O}$). Then, it performs two rewritings using equations (i, ii) and returns an equivalent version of the objective $\mathcal{Z} = \lambda_0 + 16\,\lambda_2 + 4\,\lambda_3 + 21\,\lambda_5$ on which it is clear that choosing $\lambda_0, \lambda_2, \lambda_3, \lambda_5$ greater than 0 would increase the value of $\mathcal{Z}$ because their coefficients are positive. Thus, the minimum value of $\mathcal{Z}$ is reached for $\lambda_0 = \lambda_2 = \lambda_3 = \lambda_5 = 0$ which entails $\lambda_1 = \frac{1}{3}$ and $\lambda_4 = \frac{2}{3}$ using equations (i) and (ii). This example summarizes the principle of linear programming.

Now consider our minimization problem (3) with its *parametric objective* $\mathcal{Z}(x_1, x_2, x_3) = \lambda_1(-x_1 - 2\,x_2 + 2\,x_3 + 7) + \lambda_2(-x_1 + 2\,x_2 - 1) + \lambda_3(3\,x_1 - x_2) + \lambda_4(-x_3 + 10) + \lambda_5(x_1 + x_2 + x_3 - 5) + \lambda_0$. Our PLP-solver uses the previous instantiated problem to discover the useful pivots (i, ii). Then, it replays the same rewritings on the parametric version. Those operations are efficiently implemented using the matrix representation of (3): they boils down to the addition of combinations of rows of (†) and $\boldsymbol{\alpha}$ to those of $\boldsymbol{O}$. We end up with the following objective:

$$\underbrace{-\frac{1}{3}\,x_1 - \frac{2}{3}\,x_2 + 9}_{\mathcal{Z}^\star_4} + \lambda_0 \underbrace{\frac{1}{3}\,(x_1 + 2\,x_2 - 24)}_{\geq 0\,:\,\boldsymbol{C_{4.1}}} + \lambda_2 \underbrace{\frac{2}{3}\,(-x_1 + 4\,x_2 - 15)}_{\geq 0\,:\,\boldsymbol{C_{4.2}}}$$
$$+ \lambda_3 \underbrace{\frac{1}{3}\,(10\,x_1 - x_2 - 27)}_{\geq 0\,:\,\boldsymbol{C_{4.3}}} + \lambda_5 \underbrace{\frac{1}{3}\,(5\,x_1 + 7\,x_2 - 39)}_{\geq 0\,:\,\boldsymbol{C_{4.4}}}$$

We recognize the 4^{th} piece of $\mathcal{Z}^\star$. The argument for optimality used in the non-parametric version can be generalized: The minimality of $\mathcal{Z}_4^\star$ holds if *the parametric coefficients of the remaining variables are nonnegative*, since increasing the values of $\lambda_0, \lambda_2, \lambda_3, \lambda_5$ (which must be nonnegative) would make the objective value grow. This condition defines the region of optimality $\mathcal{R}_4$ of $\mathcal{Z}_4^\star$ as the polyhedron $\{ \boldsymbol{C_{4.1}}, \boldsymbol{C_{4.2}}, \boldsymbol{C_{4.3}} \}$, see Fig.1(a). $\boldsymbol{C_{4.4}}$ is actually redundant with respect to $\boldsymbol{C_{4.1}}$, $\boldsymbol{C_{4.2}}$ and $\boldsymbol{C_{4.3}}$. It is thus eliminated from the representation of $\mathcal{R}_4$, and therefore does not appear on Fig.1(a).

The PLP-solver then chooses an opposite sign condition of a parametric coefficient $\boldsymbol{C_{4.i}}$ – that means exploring an adjacent region by crossing a frontier – and selects a new instantiation point on this side of the constraint. The objective is then instantiated accordingly and submitted to the simplex which provides the meaningful pivots leading to another optimal affine form and its region of optimality. The benefit of PLP is that the exploration of one instance with the simplex is generalized into a whole region of optimality. The exploration goes on until the whole space of parameters has been covered by the union of regions: any new instantiation point falls in an already explored region.

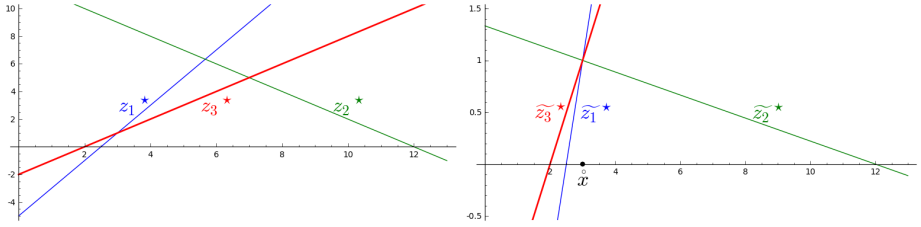
6 Polyhedra in Minimal Form for Free

The previous sections showed how to compute the optimal solution of a PLOP and how to interpret the solution $\mathcal{Z}^\star$ as a polyhedron $\mathcal{P}^\star = \bigwedge_{k=1}^{r} \mathcal{Z}_k^\star(\boldsymbol{x}) \geq 0$. Still, the representation of $\mathcal{P}^\star$ may not be minimal: some constraints $\mathcal{Z}_k^\star(\boldsymbol{x}) \geq 0$ may be redundant in $\mathcal{P}^\star$. We could remove those redundancies afterwards but, as noticed by Howe *et al.* [15], it is highly preferable to prevent their generation by adding a *normalization constraint* to the PLOP. We adapt their intuition to our formulation of the problem and we bring the proof that it indeed avoids redundancies. This requires to make a detour via normalized solutions to explain the expected effect of a normalization constraint.

6.1 Normalizing the Projection PLOP

Let us normalize the function $\mathcal{Z}^\star$ so that it evaluates to 1 on a given point $\mathring{\boldsymbol{x}}$ in the interior of $\mathcal{P}^\star$. Formally, we consider a solution $\widetilde{\mathcal{Z}}^\star(\boldsymbol{x}) \stackrel{def}{=} \frac{\mathcal{Z}^\star(\boldsymbol{x})}{\mathcal{Z}^\star(\mathring{\boldsymbol{x}})}$ or equivalently $\forall k,\ \widetilde{\mathcal{Z}}_k^\star(\boldsymbol{x}) \stackrel{def}{=} \frac{\mathcal{Z}_k^\star(\boldsymbol{x})}{\mathcal{Z}_k^\star(\mathring{\boldsymbol{x}})}$. The key point of this transformation is that the space $[\![\mathcal{Z}^\star \geq 0]\!]$, which is the polyhedron $\mathcal{P}^\star$ of interest, is unchanged. The normalized solution $\widetilde{\mathcal{Z}}^\star$ will differ from the original one but must fulfills $[\![\widetilde{\mathcal{Z}}^\star \geq 0]\!] = [\![\mathcal{Z}^\star \geq 0]\!]$ which is true on the main functions if it holds on each of their pieces, *i.e.* $\forall k, [\![\widetilde{\mathcal{Z}}_k^\star \geq 0]\!] = [\![\mathcal{Z}_k^\star \geq 0]\!]$. The normalization preserves the nonnegativity space of each $\mathcal{Z}_k^\star$ because $\frac{1}{\mathcal{Z}_k^\star(\mathring{\boldsymbol{x}})}$ is a positive scalar: Indeed, $\mathring{\boldsymbol{x}}$ belongs to the interior of $\mathcal{P}^\star$, *i.e.* $[\![\bigwedge_k \mathcal{Z}_k^\star > 0]\!]$ by Lemma 2. The proof of this remark is given in Lemma 7, available in the appendix of the extended version of this paper [21].

Example 2. The transformation of the solution only changes the inclination of $\mathcal{Z}^\star_k$, not the space where they cross 0. This can easily be illustrated on one-variable constraints. Consider three constraints $\boldsymbol{C_1} : 2x \geq 5$, $\boldsymbol{C_2} : x \leq 12$ and a redundant one $\boldsymbol{C_3} : x \geq 2$, corresponding to three affine forms $\mathcal{Z}^\star_1(x) = 2x - 5$, $\mathcal{Z}^\star_2(x) = 12 - x$ and $\mathcal{Z}^\star_3(x) = x - 2$. On the left-hand side we plotted the functions $z = \mathcal{Z}^\star_i(x)$ for $i \in \{1, 2, 3\}$ and, on the right-hand side, their normalizations with respect to the point $\mathring{x} = 3$.



The most interesting consequence of the normalization is that *a constraint is redundant iff its normalized affine form is nowhere minimal.* This property does not hold on the non-normalized forms: although $\boldsymbol{C_3}$ is redundant *w.r.t.* $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$, $\mathcal{Z}^\star_3$ is minimal *w.r.t.* $\mathcal{Z}^\star_1$ and $\mathcal{Z}^\star_2$ on $x \in [3, 7]$. On the contrary, considering the normalized forms, $\widetilde{\mathcal{Z}^\star_3}$ is no longer minimal, thus *it will be absent from the piecewise solution of a minimization problem.* One of our contribution is the proof of this result (§6.2).

Last, but no least, the normalized pieces are not computed *a posteriori* from the original solutions. They are obtained directly by enforcing the normalization of the objective through an additional constraint $\mathcal{Z}(\mathring{\boldsymbol{x}}) = 1$. Recall from (4) that the objective of the PLOP is $\mathcal{Z}(\boldsymbol{x}) \stackrel{def}{=} \lambda_0 + \sum_{i=1}^{p} \lambda_i \times (\boldsymbol{a_i}(\boldsymbol{x}) - b_i)$. Then, the *normalization constraint* becomes (‡) $\lambda_0 + \sum_{i=1}^{p} \lambda_i \times (\boldsymbol{a_i}(\mathring{\boldsymbol{x}}) - b_i) = 1$ where the $\boldsymbol{a_i}(\mathring{\boldsymbol{x}})$ are coefficients in $\mathbb{Q}$, obtained by evaluating the constraints of the input polyhedron at $\mathring{\boldsymbol{x}}$. The normalization constraint replaces the previous requirement (†) $\sum_i \lambda_i = 1$ in the PLOP: like (†) it excludes the solution $\lambda_0 = \ldots = \lambda_p = 0$. Back to Example 1.5., our PLP-solver running on the normalized PLOP only builds the irredundant constraints $\mathcal{Z}^\star_2 \geq 0$, $\mathcal{Z}^\star_3 \geq 0$ and $\mathcal{Z}^\star_4 \geq 0$ associated to the regions of Fig.1(b).

Note that we must be able to provide a point $\mathring{\boldsymbol{x}}$ in the interior of $\mathcal{P}^\star$ while $\mathcal{P}^\star$ is not already known. Finding such a point is obvious for projection, convex-hull and Minkowski sum. It is feasible because the operators based on PLP are applied on polyhedra with non-empty interior; the treatment of polyhedra with equalities is explained in Example 3 below. For projection, $\mathring{\boldsymbol{x}}$ is obtained from a point $\boldsymbol{x}$ in the interior of the input polyhedron $\mathcal{P}$. Removing the coordinates of variables marked for elimination provides a point $\mathring{\boldsymbol{x}}$ that will be in the interior of the projected polyhedron $\mathcal{P}^\star$.

Example 3. Consider the case of a polyhedron over variables x, x', x'' made of inequalities $\mathcal{P}$ and an equality $E : x'' = f(x, x')$. The computation of the projection $(\mathcal{P} \wedge E)_{/\{x', x''\}}$ is done in two steps: we use equation $x'' = f(x, x')$ to eliminate x'' from $\mathcal{P}$ by substitution. If implicit equalities show up we exploit them in

the same way, otherwise we apply the projection via PLP on $\mathcal{P}[x''/f(x,x')]$ to eliminate the remaining variable x'.

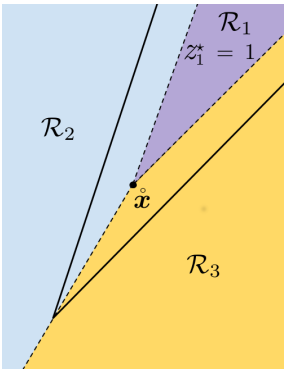
6.2 A Normalized PLOP is Free of Redundancy

The advantage of PLP over Fourier-Motzkin comes from the following theorem:

Theorem 1. *Let* $\widetilde{\mathcal{Z}}^\star \stackrel{def}{=} \min\{\widetilde{\mathcal{Z}}_1^\star, \ldots, \widetilde{\mathcal{Z}}_r^\star\}$ *be the optimal solution of a* normalized parametric minimization problem. *Then each solution* $\widetilde{\mathcal{Z}}_k^\star$ *that is not the constant function* $\boldsymbol{x} \mapsto 1$ *is irredundant with respect to polyhedron* $[\![\widetilde{\mathcal{Z}}^\star \geq 0]\!]$.

Proof. Theorem 1 is a direct consequence of three intermediates results: (i) each region of optimality in a normalized PLOP is a cone pointed in $\mathring{\boldsymbol{x}}$ (Lemma 3); (ii) each piece $\mathcal{Z}_k^\star$ which is not constant, is decreasing on its region of optimality along lines starting at $\mathring{\boldsymbol{x}}$ (Lemma 4); (iii) each piece that crosses 0 on its region produces an irredundant constraint (Lemma 5). □

Let us summarize the key facts that are needed for exposing the proof of the lemmata: Projection via PLP leads to a parametric linear minimization problem whose solution is a function $\widetilde{\mathcal{Z}}^\star$ defined by pieces $\{\widetilde{\mathcal{Z}}_1^\star$ on $\mathcal{R}_1, \ldots, \widetilde{\mathcal{Z}}_r^\star$ on $\mathcal{R}_r\}$; each $\mathcal{R}_k$ is the *region of optimality* of $\widetilde{\mathcal{Z}}_k^\star$, meaning that among all the pieces $\widetilde{\mathcal{Z}}_k^\star$ is the minimal one on $\mathcal{R}_k$, *i.e.* $\mathcal{R}_k = \{\boldsymbol{x} \mid \widetilde{\mathcal{Z}}^\star(\boldsymbol{x}) = \widetilde{\mathcal{Z}}_k^\star(\boldsymbol{x})\}$. By construction, $\widetilde{\mathcal{Z}}^\star(\boldsymbol{x})$ is the minimum of $\{\widetilde{\mathcal{Z}}_1^\star(\boldsymbol{x}), \ldots, \widetilde{\mathcal{Z}}_r^\star(\boldsymbol{x})\}$ and $\widetilde{\mathcal{Z}}^\star(\mathring{\boldsymbol{x}}) = \widetilde{\mathcal{Z}}_1^\star(\mathring{\boldsymbol{x}}) = \ldots = \widetilde{\mathcal{Z}}_r^\star(\mathring{\boldsymbol{x}}) = 1$ is enforced by the (‡)-normalization constraint. This is where λ_0 comes into play: the fact that $\boldsymbol{\lambda} = (1, 0, ..., 0)$ fulfills (‡) and ($\mathcal{O}$), hence leading to the constant function $\mathcal{Z}_{\boldsymbol{\lambda}}^\star = 1$, sets an upper-bound on $\mathcal{Z}^\star$. Therefore, any minimal piece $\widetilde{\mathcal{Z}}_k^\star$, which evaluates to 1 on $\mathring{\boldsymbol{x}}$, can not grow on its region of optimality otherwise it would not be minimal compared to $\mathcal{Z}_{\boldsymbol{\lambda}}^\star = 1$. Thus, $\widetilde{\mathcal{Z}}_k^\star$ is either constant and equal to 1 or it satisfies $\forall \boldsymbol{x} \in \mathring{\mathcal{R}}_k,\ 1 > \mathcal{Z}_k^\star(\boldsymbol{x})$ which entails its decline on the infinite region $\mathcal{R}_k$ as meant by Lemma 4, causing its nullification in $\mathcal{R}_k$, hence its irredundancy (Lemma 5). The constant piece $\mathcal{Z}_{\boldsymbol{\lambda}}^\star = 1$ arises among the solutions of a normalized PLOP when the resulting polyhedron $\mathcal{P}^\star$ is unbounded as illustrated alongside.



We focus on the proof of Lemma 5 which gives a criterion of irredundancy illustrated on Fig.1. The proofs of the other lemmata are just computational arguments; they are provided in the appendix of the extended version of this paper [21].

Lemma 3. $\forall \boldsymbol{x} \in \mathbb{Q}^n,\ \boldsymbol{x} \in \mathring{\mathcal{R}}_k \Rightarrow \mathring{\boldsymbol{x}} + \mu(\boldsymbol{x} - \mathring{\boldsymbol{x}}) \in \mathring{\mathcal{R}}_k,\ \forall \mu > 0.$

Lemma 4. *Either* $\widetilde{\mathcal{Z}}_k^\star$ *is the constant function* $\boldsymbol{x} \mapsto 1$*, or it decreases on lines of* $\mathcal{R}_k$ *starting at* $\mathring{\boldsymbol{x}}$*, i.e.* $\forall \boldsymbol{x} \in \mathring{\mathcal{R}}_k,\ \forall \mu > 1,\ \widetilde{\mathcal{Z}}_k^\star(\boldsymbol{x}) > \widetilde{\mathcal{Z}}_k^\star(\mathring{\boldsymbol{x}} + \mu(\boldsymbol{x} - \mathring{\boldsymbol{x}}))$.

Lemma 5. $\left(\llbracket \widetilde{z}^\star_k = 0 \rrbracket \cap \llbracket \mathring{\mathcal{R}}_k \rrbracket\right) \neq \emptyset \Rightarrow \widetilde{z}^\star_k \geq 0$ *is irredundant* w.r.t. $\widetilde{z}^\star \geq 0$.

Proof by contradiction. Consider $\widetilde{z}^\star_k$, a piece of $\widetilde{z}^\star$ such that $\llbracket \widetilde{z}^\star_k = 0 \rrbracket \cap \llbracket \mathring{\mathcal{R}}_k \rrbracket \neq \emptyset$. Let us assume that $\widetilde{z}^\star_k$ is redundant. Then, by Farkas Lemma 1, $\exists (\lambda_j)_{j\neq k} \geq 0,\ \forall \boldsymbol{x} \in \mathbb{Q}^n,\ \sum_{j\neq k} \lambda_j \widetilde{z}^\star_j(\boldsymbol{x}) \leq \widetilde{z}^\star_k(\boldsymbol{x})$. Let $\boldsymbol{x}$ be a point of the nonempty set $\llbracket \widetilde{z}^\star_k = 0 \rrbracket \cap \llbracket \mathring{\mathcal{R}}_k \rrbracket$. Then $\widetilde{z}^\star_k(\boldsymbol{x}) = 0$, as $\boldsymbol{x} \in \llbracket \widetilde{z}^\star_k = 0 \rrbracket$, and the previous Farkas inequality becomes

$$\sum_{j\neq k} \lambda_j \widetilde{z}^\star_j(\boldsymbol{x}) \leq 0 \tag{7}$$

Since $\boldsymbol{x} \in \mathring{\mathcal{R}}_k$, then, $\widetilde{z}^\star_k(\boldsymbol{x}) < \widetilde{z}^\star_j(\boldsymbol{x})$ for $j \neq k$ by definition of $\mathcal{R}_k$ as the region of optimality of $\widetilde{z}^\star_k$. More precisely, $0 < \widetilde{z}^\star_j(\boldsymbol{x})$ since $\boldsymbol{x} \in \llbracket \widetilde{z}^\star_k = 0 \rrbracket$. Therefore, $0 < \lambda_j \widetilde{z}^\star_j(\boldsymbol{x})$ for $j \neq k$ as $\lambda_j \geq 0$. Then, summing up this inequation for all $j \neq k$, we obtain

$$0 < \sum_{j\neq k} \lambda_j \widetilde{z}^\star_j(\boldsymbol{x}) \tag{8}$$

(7) and (8) are contradictory, proving thereby that $\widetilde{z}^\star_k$ is irredundant. □

6.3 Minimizing Operators based on Projection via PLP

As mentioned in introduction, several polyhedral operators, e.g. Minkowski sum, convex hull, assignment and linearization, are encoded using extra variables which are then eliminated by projection. If the projection is done by PLP, all these operators produce polyhedra free of redundancy if we can provide a normalization point in the interior of the expected polyhedron. Let us give insights of the encodings.

The Minkowski sum of two polyhedra $\mathcal{P}'$ and $\mathcal{P}''$ is the set of points $\boldsymbol{x} = \boldsymbol{x'} + \boldsymbol{x''}$ with $\boldsymbol{x'} \in \llbracket \mathcal{P}' \rrbracket$ and $\boldsymbol{x''} \in \llbracket \mathcal{P}'' \rrbracket$. It is computed by eliminating the variables of $\boldsymbol{x'}$ and $\boldsymbol{x''}$ from the polyhedron $\mathcal{P}'(\boldsymbol{x'}) \wedge \mathcal{P}''(\boldsymbol{x''}) \wedge \{\, \boldsymbol{x} = \boldsymbol{x'} + \boldsymbol{x''} \,\}$, where $\mathcal{P}'(\boldsymbol{x'})$ (resp. $\mathcal{P}''(\boldsymbol{x''})$) denotes the set of constraints of $\mathcal{P}'$ (resp. $\mathcal{P}''$) over variables $\boldsymbol{x'}$ (resp. $\boldsymbol{x''}$). We use $\mathring{\boldsymbol{x}} \stackrel{def}{=} \mathring{\boldsymbol{x}}' + \mathring{\boldsymbol{x}}''$ as normalization point where $\mathring{\boldsymbol{x}}'$ (resp. $\mathring{\boldsymbol{x}}''$) is a point lying within the interior of $\mathcal{P}'$ (resp. $\mathcal{P}''$).

The convex-hull of $\mathcal{P}'$ and $\mathcal{P}''$ is the smallest convex polyhedron that includes $\mathcal{P}'$ and $\mathcal{P}''$. It is the set of barycentres of $\boldsymbol{x'} \in \llbracket \mathcal{P}' \rrbracket$ and $\boldsymbol{x''} \in \llbracket \mathcal{P}'' \rrbracket$ which can be formally defined as $\mathcal{P}'(\boldsymbol{x'}) \wedge \mathcal{P}''(\boldsymbol{x''}) \wedge \{\boldsymbol{x} = \beta_1 \times \boldsymbol{x'} + \beta_2 \times \boldsymbol{x''}, \beta_1 + \beta_2 = 1, \beta_1 \geq 0, \beta_2 \geq 0\}$. The equation defining $\boldsymbol{x}$ is non-linear but it can be linearized using a simple change of variable [2]. Then, the convex-hull is obtained by elimination of β_1, β_2 and the variables of $\boldsymbol{x'}$ and $\boldsymbol{x''}$ to get a polyhedron over $\boldsymbol{x}$. We can use $\mathring{\boldsymbol{x}}'$ or $\mathring{\boldsymbol{x}}''$ as normalization point.

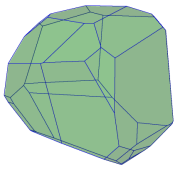
Assignment and more generally, image by an affine map represented by a matrix $\boldsymbol{M}$ can be encoded as intersection with equalities $\boldsymbol{x'} = \boldsymbol{M}\boldsymbol{x}$, projection of

the unprimed variables, and then renaming of the prime variables into unprimed ones; the reverse image is just substitution. We use the image by $\boldsymbol{M}$ of a point $\mathring{\boldsymbol{x}}$ in the interior of the input polyhedron for normalization.

Our linearization operator for computing a polyhedral over-approximation of a conjunctions of linear and polynomial constraints $\bigwedge_i g_i(\boldsymbol{x}) \geq 0$ is also implemented in the Vpl via PLP [20]. However, it does not prevent redundancies as we do not know how to provide a normalization point satisfying $\bigwedge_i g_i(\mathring{\boldsymbol{x}}) \geq 0$.

7 Experiments

Benchmarks. We reused the benchmark suite of [22]. It contains polyhedra generated randomly from several characteristics: number of constraints, number of variables and density (ratio of the number of zero coefficients by the number of variables). Constraints are created by picking up a random integer between -100 and 100 as coefficient of each variable. All constraints are attached the same constant bound ≤ 20. These polyhedra have a potatoid shape, as shown on the right-hand side figure.



We compare three libraries on projection/minimization problems: NewPolka [16] as representative of the double description framework, Vpl [10] based on Fourier-Motzkin elimination, and our implementation based on PLP. As we produce polyhedra in minimized form, we asked NewPolka and Vpl to perform a minimization afterwards.

On each problem we measure the execution time, with a timeout fixed at 300 seconds. In addition to the number of constraints C, the density D and the number of variables V, we consider the effect of the projection ratio P (number of projected variable over dimension). Fig.2 shows the effect of these characteristics on execution time (in seconds). The vertical axis is always displayed in log scale, for readability. Each point is the average execution time for the projection and minimization of 10 polyhedra sharing the same characteristics.

Fourier-Motzkin Elimination in the Vpl. As mentioned earlier, Fourier-Motzkin elimination generates many redundant constraints and the challenge of a good implementation is their fast removal. The Fourier-Motzkin elimination implemented in the Vpl uses well-known tricks for dynamically removing constraints that can be shown redundant by syntactic arguments [22]. However, as shown by [8, 3.2.3, p. 76], this forbids the use of Kohler's redundancy criterion: when eliminating k variables, a constraint resulting from the combination of $k+1$ constraints is redundant. When syntactic criteria fail to decide the redundancy of a constraint, the Vpl calls a LP solver. Hence, polyhedra are minimized after each single-variable elimination.

Projection Ratio. Fig.2(a) gives the time measurements when projecting polyhedra of 15 constraints, 10 variables and a density of 50%, with a projection ratio varying from 10 to 90%. Fourier-Motzkin is very efficient when projecting

a small number of variables. Its exponential behavior mainly occurs for high projection ratio, as it eliminates variables one after the other and the number of faces tends to grow at each projection. PLP is not suitable when there is only few variables to project, e.g. in the case of a single assignment. On the contrary, it becomes interesting compared to Fourier-Motzkin elimination when the projection ratio exceeds 50%, *i.e.* when projecting more than half of the variables. This ratio is always reached when computing Minkowski sums or convex hulls by projection (§6.3). It can also be the case on exits of program blocks where a whole set of local variables must be forgotten. As PLP usefulness grows with a high projection ratio we will focus on the case $P = 75\%$, studying the effect of other characteristics.

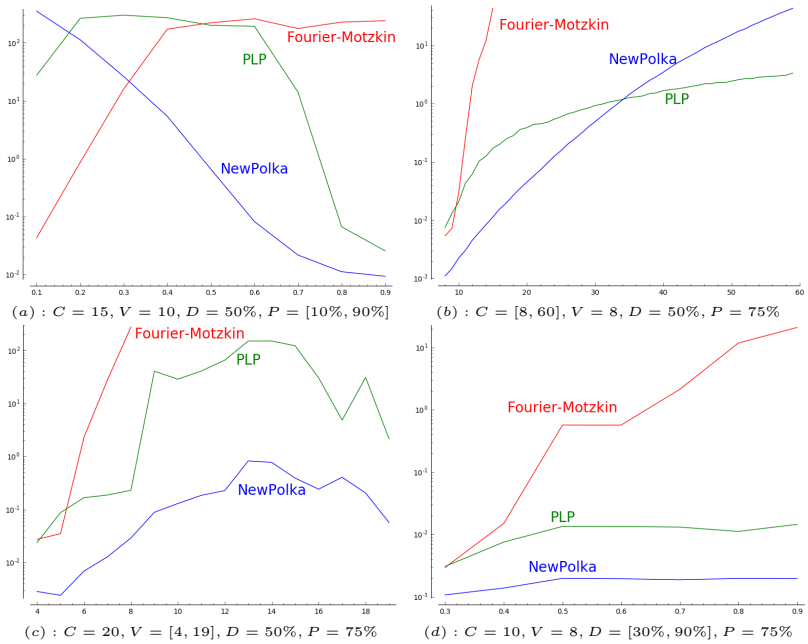


Fig. 2. Execution time in seconds of NewPolka (blue), Fourier-Motzkin (red) and PLP (green) depending on respectively (a) projection ratio, (b) number of constraints, (c) number of variables and (d) density.

Number of Constraints. Fig.2(b) shows the time measurements when projecting polyhedra with 8 variables, a density of 50% and a projection ratio of 75% (*i.e.* elimination of 6 variables). The number of constraints varies in $[8, 60]$. While Fourier-Motzkin blows up when reaching 15 constraints, PLP and NewPolka scale better and the curves shows that PLP wins when the number of constraints exceeds 35.

Dimension. The evolution of execution time in terms of dimension is given in Fig.2(c). With 20 constraints, the exponential behavior of Fourier-Motzkin elimination emerges. PLP and NewPolka show a similar curves with an overhead for PLP on a log scale, *i.e.* a proportionality factor on execution time. It would be interesting to see the effect of dimension beyond 20 variables, which takes considerable time since it requires increasing the number of constraints. Indeed, when the dimension is greater than the number of constraints, polyhedra have a really special shape with very few generators and the comparison would be distorted.

Density. The effect of density on execution time is shown on Fig.2(d). NewPolka and PLP are little sensitive to density. The case of Fourier-Motzkin can be explained: Elimination of a variable x with FM consists in combining every pair of constraints having an opposite sign for x. The more non-zero coefficients within the constraints, the greater the number of possible combinations.

What can we conclude from these experiments? On small problems our projection is less efficient than that of a double description (DD) library but the shape of the curves of NewPolka and PLP is similar on a logarithmic scale, meaning that there is a proportionality factor between the two algorithms. This is an encouraging result as projection – and the operators encoded as projection – are the Achilles heel of constraints-only representation whereas it is straightforward in DD: the complexity is exponential with FM elimination but linear in the number of generators. On the other hand, the conjunction operator, which, in constraints-only representation, consists in the union of two sets followed by a fast minimization [22], is less efficient in DD because it triggers one step of Chernikova's algorithm per constraint.

8 Conclusion & Future Work

We have shown how usual operations over convex polyhedra (projection, convex hull, Minkowski sums, image by an affine map, linearization) can be formulated as PLOP instances. In short, all costly operations on polyhedra in constraints-only representation can be implemented using PLOP.

This approach was made practical by the combination of an efficient PLP-solver and a normalization constraint ensuring that the solutions of the PLOP are free of redundancies, which avoids costly post-processing minimization. This makes the Vpl, a polyhedra library in constraints-only representation, competitive with other libraries in double description, and much faster on problems that have exponential generator representations.

Experiments on Minkowski sum met our expectations but raised an issue for convex-hull: On large problems with the same characteristics, we beat other libraries, but we suffer from an exponential blow-up of region subdivisions when the two polyhedra have many faces in common, which induces a high degree of degeneracy. Our PLP-solver does not have special counter-measures to deal with

degeneracy. Proposals exist for tackling primal and dual degeneracies but they come with an extra-cost [18]. Thus, dealing with degeneracy is a trade-off and we need a deeper understanding of the phenomenon before addressing it in our PLP-solver or by a pre-processing for convex-hull.

As future work, our approach can be combined with Cartesian product factorization [13,30]. While the main advantage of factorization is to avoid exponential generator representations, which we also do because we never compute generators, using low dimension factors is likely to speed up parametric linear programming.

Other avenues of research include experiments in the large on static analysis of actual programs, the parallelization of the algorithms (we already use a parallel minimization algorithm) and the increased use of floating-point computations instead of exact rational arithmetic without destroying soundness.

Acknowledgments. The authors would like to thank Alexis Fouilhé, Andy King, Jacob Howe, and Paul Feautrier for their help on the early stages of this work.

References

1. Bastoul, C.: Contributions to High-Level Program Optimization. Habilitation thesis, Université Paris-Sud (2012)
2. Benoy, F., King, A., Mesnard, F.: Computing convex hulls with a linear solver. Theory and Practice of Logic Programming (TPLP) 5(1-2), 259–271 (2005)
3. Chernikova, N.V.: Algorithm for discovering the set of all the solutions of a linear programming problem. USSR Computational Mathematics and Mathematical Physics (1968)
4. Chvatal, V.: Linear Programming. Series of books in the Mathematical Sciences, W. H. Freeman (1983)
5. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: ACM Principles of Programming Languages (POPL). pp. 238–252. ACM Press (1977)
6. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: ACM Principles of Programming Languages (POPL). pp. 84–97. ACM Press (1978)
7. Dantzig, G.B., Thapa, M.N.: Linear Programming 2: Theory and Extensions. Operations Research, Springer (2003)
8. Fouilhé, A.: Revisiting the abstract domain of polyhedra: constraints-only representation and formal proof. Ph.D. thesis, Université de Grenoble (2015)
9. Fouilhé, A., Boulmé, S.: A certifying frontend for (sub)polyhedral abstract domains. In: Verified Software: Theories, Tools and Experiments (VSTTE). LNCS, vol. 8471, pp. 200–215. Springer (2014)
10. Fouilhé, A., Monniaux, D., Périn, M.: Efficient generation of correctness certificates for the abstract domain of polyhedra. In: Static Analysis Symposium (SAS). LNCS, vol. 7935, pp. 345–365. Springer (2013)
11. Gal, T., Nedoma, J.: Multiparametric linear programming. Management Science 18(7), 406–422 (1972)
12. Halbwachs, N.: Détermination automatique de relations linéaires vérifiées par les variables d'un programme. Ph.D. thesis, Université Scientifique et Médicale de Grenoble (1979), (in french)

13. Halbwachs, N., Merchat, D., Gonnord, L.: Some ways to reduce the space dimension in polyhedra computations. Formal Methods in System Design 29(1), 79–95 (2006)
14. Henry, J., Monniaux, D., Moy, M.: PAGAI: A path sensitive static analyser. Electronic Notes in Theoretical Computer Science 289, 15–25 (2012)
15. Howe, J.M., King, A.: Polyhedral analysis using parametric objectives. In: Static Analysis Symposium (SAS). LNCS, vol. 7460, pp. 41–57 (2012)
16. Jeannet, B., Miné, A.: APRON: A library of numerical abstract domains for static analysis. In: Computer Aided Verification (CAV). LNCS, vol. 5643, pp. 661–667 (2009)
17. Jones, C., N., Kerrigan, E.C., Maciejowski, J.M.: On polyhedral projections and parametric programming. Jounral of Optimization Theory and Applications 138(2), 207–220 (2008)
18. Jones, C.N., Kerrigan, E.C., Maciejowski, J.M.: Lexicographic perturbation for multiparametric linear programming with applications to control. Automatica 43(10), 1808–1816 (2007)
19. Le Verge, H.: A note on Chernikova's algorithm. Tech. Rep. 1662, INRIA (1992)
20. Maréchal, A., Fouilhé, A., King, T., Monniaux, D., Périn, M.: Polyhderal Approximation of Multivariate Polynomials using Handelman's Theorem. In: Verification, Model Checking, and Abstract Interpretation (VMCAI). LNCS, vol. 9583, pp. 166–184. Springer (2016)
21. Maréchal, A., Monniaux, D., Périn, M.: Scalable minimizing-operators on polyhedra via parametric linear programming. Tech. Rep. 4, Verimag (June 2017)
22. Maréchal, A., Périn, M.: Efficient Elimination of Redundancies in Polyhedra by Raytracing. In: Verification, Model Checking, and Abstract Interpretation (VMCAI). LNCS, vol. 10145, pp. 367–385. Springer (2017)
23. McMullen, P.: The maximum numbers of faces of a convex polytope. Mathematika 17, 179–184 (1970)
24. McMullen, P., Shepard, G.C.: Convex polytopes and the upper bound conjecture, London Mathematical Society Lecture Note Series, vol. 3. Cambridge University Press (1971)
25. Miné, A.: The octagon abstract domain. Higher-Order and Symbolic Computation 19(1), 31–100 (2006)
26. Monniaux, D.: Quantifier elimination by lazy model enumeration. In: Computer Aided Verification (CAV). LNCS, vol. 6174, pp. 585–599. Springer (2010)
27. Roberto, B., Hill, P.M., Zaffanella, E.: The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Science of Computer Programming 72(1–2), 3–21 (2008)
28. Schrijver, A.: Theory of linear and integer programming. Wiley (1999)
29. Simon, A., King, A.: Exploiting sparsity in polyhedral analysis. In: Static Analysis Symposium (SAS). LNCS, vol. 3672, pp. 336–351 (2005)
30. Singh, G., Püschel, M., Vechev, M.: Fast polyhedra abstract domain. In: ACM Principles of Programming Languages (POPL). pp. 46–59. ACM Press (2017)
31. Singh, G., Püschel, M., Vechev, M.T.: Making numerical program analysis fast. In: Programming Language Design and Implementation (PLDI). pp. 303–313. ACM Press (2015)