Polyhedral Approximation of Multivariate Polynomials using Handelman's Theorem *

A. Maréchal, A. Fouilhé, T. King, D. Monniaux and M. Périn

Université Grenoble-Alpes, VERIMAG, F-38000 Grenoble, France
CNRS, VERIMAG, F-38000 Grenoble, France
firstname.lastname@imag.fr

Abstract. Convex polyhedra are commonly used in the static analysis of programs to represent over-approximations of sets of reachable states of numerical program variables. When the analyzed programs contain nonlinear instructions, they do not directly map to standard polyhedral operations: some kind of linearization is needed. Convex polyhedra are also used in satisfiability modulo theory solvers which combine a propositional satisfiability solver with a fast emptiness check for polyhedra. Existing decision procedures become expensive when nonlinear constraints are involved: a fast procedure to ensure emptiness of systems of nonlinear constraints is needed. We present a new linearization algorithm based on Handelman's representation of positive polynomials. Given a polyhedron and a polynomial (in)equality, we compute a polyhedron enclosing their intersection as the solution of a parametric linear programming problem. To get a scalable algorithm, we provide several heuristics that guide the construction of the Handelman's representation. To ensure the correctness of our polyhedral approximation, our OCAML implementation generates certificates verified by a checker certified in COQ.

1 Numerical Static Analysis and Satisfiability Testing Using Convex Polyhedra

We present a new method for computing polyhedral approximations of polynomial guards, with applications in both static analysis and satisfiability modulo theory (SMT) solving. It is implemented in the Verimag Verified Polyhedra Library (VPL), a certified library written in OCAML for computing over convex polyhedra [20]. Its operators generate certificates which may optionally be checked by a verifier developed and proved correct in COQ. The VPL is used as an abstract domain within a COQ-certified static analyzer [26].

Convex polyhedra. A convex polyhedron is defined by a conjunction of affine constraints of the form $a_0 + \sum_{i=1}^{n} a_i x_i \geq 0$ where the x_i's are variables, the a_i's

* This work was partially supported by ANR project VERASCO (INS 2011) and the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement nr. 306595 "STATOR".

and a_0 are constants in $\mathbb{Q}$. We subsequently omit *convex* as we only deal with convex polyhedra. For instance, the polyhedron P defined by

$$P \triangleq \{ x - 1 \geq 0,\ y + 2 \geq 0,\ x - y \geq 0,\ 5 - x - y \geq 0 \} \tag{1}$$

is the set $\{ (x, y) \mid x \geq 1 \wedge y \geq -2 \wedge x \geq y \wedge x + y \leq 5 \}$ represented in Fig. 1. A bounded polyhedron is called a *polytope*.

Polyhedral static analysis. Static analyzers are verification tools that aim at proving properties true for all possible executions of a program; desirable properties include for instance the absence of arithmetic overflow. In the abstract interpretation framework, the analyzer attaches to each control point an *invariant* chosen within a given class, called *abstract domain* [11]. Here, we focus on the abstract domain of polyhedra which captures affine relations among program variables [22]. A static analyzer using polyhedra cannot directly infer any information on a variable z assigned with a non-linear expression *e.g.* $z := x{*}y$. A very rough abstraction is to consider that z is assigned any value in $(-\infty, +\infty)$; the consequence is a dramatic loss of precision which propagates along the analysis, possibly failing to prove a property.

Satisfiability modulo theory. The satisfiability of a quantifier-free formula of first-order linear arithmetic over the reals is usually decided by a "DPLL(T)" [21] combination of a propositional solver and a decision procedure for conjunctions of linear inequalities based on the simplex algorithm [18,17]. Nonlinear formulas are more challenging; some solvers implement a variant of cylindrical algebraic decomposition, a very complex and costly approach [27]; some replace the propositional abstraction of DPLL(T) by a direct search for a model [14].

Linearization techniques. Nonlinear relations between variables, such as $x^2 + y^2 \leq 1$, occur for instance in address computations over matrices, computational geometry, automatic control and in programs that approximate transcendental functions (sin, cos, log...) by polynomials [7,6]. Therefore, *linearization* techniques were developed to preserve precision in the presence of polynomials; they provide an over-approximation of a polynomial on an input polyhedron. Miné proposed two linearization techniques based on variable "intervalization" [34], where some variables of the polynomial are replaced by their interval of variation:

(1) Switching to the abstract domain of polyhedra with interval coefficients [5] to maintain precision, albeit at high algorithmic cost.
(2) Obtaining an affine expression with intervals as coefficients, which is then converted into a polyhedron. This solution was implemented in the APRON polyhedra library [24,34]: intervals are replaced with their center value and the right-hand side constant of the equality is enlarged accordingly. We developed an improved and certified version of this algorithm in the VPL [4]. This linearization technique is efficient but not very precise.

Another well known linearization method consists in representing polynomials in the Bernstein basis. Bernstein coefficients give a bounding polyhedron,

made as precise as needed by increasing the degree of the basis [35]. Bernstein's linearization works on systems of generators, either to get the range of each variable, or to refer to variables as barycentric coordinates of the vertices [9]. It would be well-suited for most libraries (APRON [24], PPL [1], POLYLIB [31]), as they maintain a *double representation* of polyhedra: as systems of constraints, and as systems of generators (in the case of polytopes, the generators are the vertices). In contrast, our work aims at adding a precise linearization to the VPL. In order to make certification more convenient, the VPL uses only the constraint representation of polyhedra. Therefore, using Bernstein's method would be hardly appropriate as it would require expensive conversions between representations [32].

Contributions. We present a new algorithm to linearize polynomial guards which only needs constraint representation of polyhedra. Section 2 shows how any other polynomial statement reduces to guards. As explained in Section 3, our approach is based on Handelman's theorem [23], which states that a polynomial that is positive on a polytope can always be expressed as a nonnegative linear combination of products of constraints of the polytope. The algorithm consists in computing linear relaxations as solutions of a Parametric Linear Programming Problem (PLOP). Section 4 sketches the principle of PLOP solvers and focuses on an improvement we made to reduce exploration of branches that would yield redundant constraints. The method presented in this paper requires only the constraint representation of the polyhedron, as provided by the VPL or by a DPLL(T) SMT-solver, and returns a polyhedron directly as constraints as well as an emptiness flag. It soundly approximates polynomial operations over convex polyhedra and generates certificates that are checked by a verifier developed and proved in COQ. The precision of the approximation is arbitrary depending on the degree and the number of Handelman products in use; the selection of which is delegated to the heuristics presented in Section 5. Precision and efficiency of our algorithm are shown through a comparison with SMT-solvers on Quantifier-Free Nonlinear Real Arithmetic benchmarks in Section 6.

This paper elaborates on a preliminary work by the authors [33], which presented the encoding of the linear relaxation problem as a PLOP, focusing on the certification in COQ of the resulting approximation. We reuse the encoding of [33] and we extend the previous work with heuristics, an experimental evaluation and a new application.

2 Focusing on Approximation of Polynomial Guards

The goal of linearization is to approximate nonlinear relations with linear ones. The approximation is sound if it contains the original nonlinear set. In other words, linearization must produce an *over-approximation* of the nonlinear set. In this work, we consider polynomial expressions formed of $(+, -, \times)$, such as $4 - x \times x - y \times y$. More general algebraic expressions, including divisions and root operators, may be reduced to that format; for instance $y = \sqrt{x^2 + 1}$ is equivalent

```
1. int x, y, z ;
2. if ( x ≥ 1 && y ≥ −2 &&
        x ≥ y && x ≤ 5 − y )
3. { if ( x ∗ x + y ∗ y ≤ 4 )
4.       {z = y ∗ x ;}
5.     else
6.       {z = 0 ;}
7. }
```

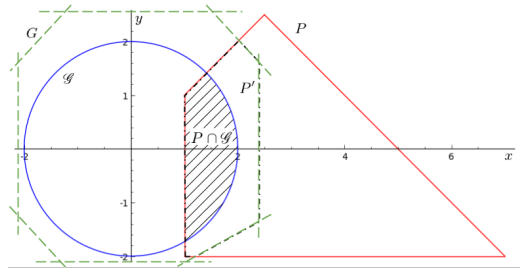


Fig. 1: A C program fragment with non-linear expressions $x * x + y * y \leq 4$ and $y * x$. The first guard defines the polyhedron $P \triangleq \{x \geq 1,\ y \geq -2,\ x - y \geq 0,\ x + y \leq 5\}$; the disc $\mathscr{G} \triangleq \{(x, y) \mid x^2 + y^2 \leq 4\}$ corresponds to the second guard; the octagon G is a polyhedral approximation of $\mathscr{G}$; the hashed region is the set $P \cap \mathscr{G}$; the desired approximation of $P \cap \mathscr{G}$ is the polyhedron $P' \triangleq P \wedge G$, drawn with dotted lines.

to $y^2 = x^2 + 1 \wedge y \geq 0$ [36]. The symbol g shall represent a polynomial expression on the variables $x_1, .., x_n$ of a program. We only consider constraints in a positive form $g \geq 0$ or $g > 0$: any other form (including equalities and negation) can be changed into a disjunction of conjunctions of positive constraints, for example $\neg(g_1 = g_2) \ \equiv\ (g_1 < g_2 \vee g_1 > g_2) \ \equiv\ (g_2 - g_1 > 0 \vee g_1 - g_2 > 0)$.

We will use the program of Fig. 1 as a running example: our goal is to compute a polyhedral over-approximation of the polynomial guard $x^2 + y^2 \leq 4$ on line 3, which is equivalent to $g \geq 0$ with $g(x, y) \triangleq 4 - x^2 - y^2$, in the context of the polytope $P \triangleq \{x - 1 \geq 0,\ y + 2 \geq 0,\ x - y \geq 0,\ 5 - x - y \geq 0\}$ that corresponds to the condition on line 2.

Note that assignments $x := e$ reduce to guards. Let $\tilde{x}$ denote the value of variable x after the assignment, while x denotes its value before the assignment. Then, the effect of the assignment on a polyhedron P is $\big((P \wedge \tilde{x} \leq e \wedge \tilde{x} \geq e)_{/x}\big)[\tilde{x}/x]$, where $\cdot_{/x}$ denotes the elimination of x using projection and $[\tilde{x}/x]$ is the renaming of $\tilde{x}$ as x. This works when e is affine. When it is nonlinear, $\tilde{x}$ is approximated by linearizing guards $x' \leq e$ and $x' \geq e$. Therefore, we will exclusively focus on the linearization of polynomial guards.

The effect of a guard $g \geq 0$ on a polyhedron P consists in the intersection of the set of points of P with $\mathscr{G} \triangleq \{(x_1, \ldots, x_n) \mid g(x_1, \ldots, x_n) \geq 0\}$. When the guard is linear, say $x - 2y \geq 0$, $P \cap \mathscr{G}$ is simply the conjunction of P and the constraint $x - 2y \geq 0$; it is already a polyhedron. When the guard is not linear, we approximate $P \cap \mathscr{G}$ by a polyhedron P' such that $P \cap \mathscr{G} \subseteq P'$. Computing, instead, a polyhedral enclosure G of the set $\mathscr{G}$ would not be a practical solution. Indeed, it can be very imprecise: if $\mathscr{G} = \{(x, y) \mid y \leq x^2\}$, then $G = \mathbb{Q}^2$. Moreover, it is superfluous work: only three of the eight constraints of polyhedron G on Fig. 1 are actually useful for the intersection.

3 Linearizing Using Handelman's Representation

Consider an input polyhedron $P \triangleq \{C_1 \geq 0, \ldots, C_p \geq 0\}$ defined on variables $(x_1, \ldots, x_n)$ and a polynomial guard $g \geq 0$. Our goal is to find an affine term $\alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$ denoted by *aff* such that $P \Rightarrow \mathit{aff} > g$, meaning that *aff* bounds g on P. By transitivity, we will conclude that $P \wedge g \geq 0 \Rightarrow P \wedge \mathit{aff} > 0$, which can be expressed in terms of sets[1] as $(P \cap g \geq 0) \subseteq (P \sqcap \mathit{aff} > 0)$. Our linearization based on Handelman's theorem provides several affine constraints $\mathit{aff}_1, \ldots, \mathit{aff}_k$ whose conjunction with P forms the approximation of $P \cap g \geq 0$. In static analysis, where P describes the possible values of the program variables $(x_1, \ldots, x_n)$ before a polynomial guard $g \geq 0$, the result $P \sqcap_{i=1}^{i=k} \mathit{aff}_i > 0$ will be a polyhedral approximation of the program state after the guard. When this polyhedron is empty, it means that the original guard $P \wedge g \geq 0$ is unsatisfiable.

3.1 Representation of Positive Polynomials on a Polytope

Notations. Tuples $\boldsymbol{x} = (x_1, \ldots, x_n)$ and multi-indices $\boldsymbol{I} = (i_1, ..., i_n) \in \mathbb{N}^n$ are set in boldface. The set of Handelman products associated to a polyhedron $P \triangleq \{C_1 \geq 0, \ldots, C_p \geq 0\}$ is the set $\mathscr{H}_P$ of all products of constraints C_i of P:

$$\mathscr{H}_P = \{ C_1^{i_1} \times \cdots \times C_p^{i_p} \mid (i_1, \ldots, i_p) \in \mathbb{N}^p \} \tag{2}$$

Given a multi-index $\boldsymbol{I} = (i_1, \ldots, i_p)$, $H^{\boldsymbol{I}} \triangleq C_1^{i_1} \times \ldots \times C_p^{i_p}$ denotes an element of $\mathscr{H}_P$. In our running example, $H^{(0,2,0,0)} = (y+2)^2$, $H^{(1,0,1,0)} = (x-1)(x-y)$ and $H^{(1,0,0,3)} = (x-1)(-x-y+5)^3$ all belong to $\mathscr{H}_P$. The $H^{\boldsymbol{I}}$'s are nonnegative polynomials on P as products of nonnegative constraints of P. Handelman's representation of a positive polynomial $g(\boldsymbol{x})$ on P is

$$g(\boldsymbol{x}) = \sum_{\boldsymbol{I} \in \mathbb{N}^p} \underbrace{\lambda_{\boldsymbol{I}}}_{\geq 0} \underbrace{H^{\boldsymbol{I}}}_{\geq 0} \text{ with } \lambda_{\boldsymbol{I}} \in \mathbb{R}^+ \tag{3}$$

The $\lambda_{\boldsymbol{I}}$'s form a *certificate* that $g(\boldsymbol{x})$ is nonnegative on P. Handelman's theorem states the non-trivial opposite implication: any positive polynomial on P can be expressed in that form [23][38, Th. 5.5][37, Th. 5.4.6][30, Th. 2.24]; a similar result already appeared in Krivine's work on decompositions of positive polynomials on semialgebraic sets [29].

Theorem 1 (Handelman, 1988) *Let $P = \{C_1 \geq 0, \ldots, C_p \geq 0\}$ be a polytope where each C_i is an affine form over $\boldsymbol{x} = (x_1, \ldots, x_n)$. Let $g(\boldsymbol{x})$ be a positive polynomial on P, i.e. $g(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \in P$. Then there exists a finite subset $\mathcal{I}$ of $\mathbb{N}^p$ and $\lambda_{\boldsymbol{I}} \in \mathbb{R}^+$ for all $\boldsymbol{I} \in \mathcal{I}$, such that $g(\boldsymbol{x}) = \sum_{\boldsymbol{I} \in \mathcal{I}} \lambda_{\boldsymbol{I}} H^{\boldsymbol{I}}$.*

Remark 1. This does not necessarily hold if $g(\boldsymbol{x})$ is only assumed to be nonnegative. Consider the inequalities $x + 1 \geq 0$ and $1 - x \geq 0$ and the nonnegative polynomial x^2. Assume the existence of a decomposition and apply (3) at $x = 0$: $H^{\boldsymbol{I}}(0) > 0$ for any $\boldsymbol{I}$, it follows that $\lambda_{\boldsymbol{I}} = 0$. This null decomposition is absurd.

[1] $\cap$ denotes the usual intersection of sets; $\sqcap$ is reserved for the intersection of polyhedra.

Remark 2. One can look for a Handelman representation of a polynomial even on *unbounded polyhedra*: its positivity will then be ensured. The existence of such representation is not guaranteed though.

The common use of Handelman's representation of a polynomial $g(\boldsymbol{x}) - \Delta$ is to determine a lower bound Δ of $g(\boldsymbol{x})$ on P. For instance, Boland *et al.* use it to compute an upper bound of the polynomial, in $\boldsymbol{x}$ and the error ϵ, which defines the cascading round-off effects of floating-point calculation [3]. Schweighofer's algorithm [38] can iteratively improve such a bound by increasing the degree of the $H^{\boldsymbol{I}}$'s. We present here another use of Handelman's theorem: we are not interested in just one bound but in a whole set of affine constraints dominating the polynomial $g(\boldsymbol{x})$ on P.

3.2 Linearization as a Parametric Linear Optimization Problem

Recall that we are looking for an affine constraint $\mathit{aff} \triangleq \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$ that approximates a non-linear guard g, meaning $\mathit{aff} > g$ on P. According to Theorem 1, if P is bounded, $\mathit{aff} - g$ which is positive on the polytope P has a Handelman representation as a nonnegative linear combination of products of the constraints of P, *i.e.*

$$\exists \mathcal{I} \subset \mathbb{N}^p,\ \mathit{aff} - g = \sum_{\boldsymbol{I} \in \mathcal{I}} \lambda_{\boldsymbol{I}} H^{\boldsymbol{I}},\ \lambda_{\boldsymbol{I}} \in \mathbb{R}^+,\ H^{\boldsymbol{I}} \in \mathscr{H}_P \tag{4}$$

Relation (4) ensures that there exists some positive combinations of g and some $H^{\boldsymbol{I}} \in \mathscr{H}_P$ that remove the monomials of degree >1 and lead to affine forms:

$$\alpha_0 + \alpha_1 x_1 + \ldots + \alpha_n x_n = \mathit{aff} = 1 \cdot g + \sum_{\boldsymbol{I} \in \mathbb{N}^p} \lambda_{\boldsymbol{I}} H^{\boldsymbol{I}}$$

Remark 3. This decomposition is not unique in general. Consider $P = \{\, x \geq 0,\ y \geq 0,\ x - y \geq 0,\ x + y \geq 0 \,\}$. The polynomial $x^2 + 2xy + y^2$ is equal to both $H^{(0,0,0,2)} = (x+y)^2$ and $H^{(2,0,0,0)} + 2H^{(1,1,0,0)} + H^{(0,2,0,0)} = (x^2) + 2(xy) + (y^2)$.

Design of our linearization method. The principle of our algorithm is to take advantage of the non-uniqueness of representation to get a precise approximation of the guard: we suppose that a set $\mathcal{I} = \{\, \boldsymbol{I_1}, \ldots, \boldsymbol{I_q} \,\}$ of indices is given and we show how to obtain every possible affine form aff_i that can be expressed as $g + \sum_{\ell=1}^{\ell=q} \lambda_\ell H^{\boldsymbol{I_\ell}}$. Each of these aff_i bounds g on P and their conjunction forms a polyhedron that over-approximates the set $P \cap (g \geq 0)$. A major difference between our work and previous work by Schweighofer [38] and Boland [3] is that we are not interested in a constant bound α_0 but an affine bound $\alpha_0 + \alpha_1 x_1 + \ldots + \alpha_n x_n$ which still depends on *parameters* $x_1, \ldots, x_n$. We now show that our problem belongs to the class of *parametric linear problems* ; Section 5 then describes the heuristics used to determine $\mathcal{I}$.

Example 1. For $g = 4 - x^2 - y^2$, we choose $\mathcal{I}$ that gives these 15 products:

$$
\begin{array}{ll}
H^{\boldsymbol{I_1}} = H^{(0,0,0,0)} = 1 & H^{\boldsymbol{I_2}} = H^{(1,0,0,0)} = x-1 \\
H^{\boldsymbol{I_3}} = H^{(0,1,0,0)} = y+2 & H^{\boldsymbol{I_4}} = H^{(0,0,1,0)} = x-y \\
H^{\boldsymbol{I_5}} = H^{(0,0,0,1)} = -x-y+5 & H^{\boldsymbol{I_6}} = H^{(2,0,0,0)} = (x-1)^2 \\
H^{\boldsymbol{I_7}} = H^{(0,2,0,0)} = (y+2)^2 & H^{\boldsymbol{I_8}} = H^{(0,0,2,0)} = (x-y)^2 \\
H^{\boldsymbol{I_9}} = H^{(0,0,0,2)} = (-x-y+5)^2 & H^{\boldsymbol{I_{10}}} = H^{(1,1,0,0)} = (x-1)(y+2) \\
H^{\boldsymbol{I_{11}}} = H^{(1,0,1,0)} = (x-1)(x-y) & H^{\boldsymbol{I_{12}}} = H^{(1,0,0,1)} = (x-1)(-x-y+5) \\
H^{\boldsymbol{I_{13}}} = H^{(0,1,1,0)} = (y+2)(x-y) & H^{\boldsymbol{I_{14}}} = H^{(0,1,0,1)} = (y+2)(-x-y+5) \\
H^{\boldsymbol{I_{15}}} = H^{(0,0,1,1)} = (x-y)(-x-y+5) &
\end{array}
$$

Considering the products $\{ H^{\boldsymbol{I_1}}, \ldots, H^{\boldsymbol{I_q}} \}$, finding the Handelman representation of $\mathit{aff} - g$ can be expressed as a linear problem. Relation (4) amounts to finding $\lambda_1, \ldots, \lambda_q \geq 0$ such that

$$
\begin{array}{ccc}
\mathit{aff} & = 1 \cdot g + \sum_{\ell=1}^{\ell=q} \lambda_\ell H^{\boldsymbol{I_\ell}} = \underbrace{(\lambda_g, \lambda_1, \ldots, \lambda_q)}_{\boldsymbol{\lambda}^\intercal} \cdot \underbrace{(g, H^{\boldsymbol{I_1}}, \ldots, H^{\boldsymbol{I_q}})^\intercal}_{\mathcal{H}_g{}^\intercal \cdot \boldsymbol{\mathcal{M}}} \\
\| & \| \\
\alpha_0 + \alpha_1 x_1 + \ldots + \alpha_n x_n & \boldsymbol{\lambda}^\intercal \cdot \mathcal{H}_g{}^\intercal \cdot \boldsymbol{\mathcal{M}} \\
\| & \| \\
\boldsymbol{\mathcal{M}}^\intercal \cdot (\alpha_0, \ldots, \alpha_n, 0, \ldots, 0) = & \boldsymbol{\mathcal{M}}^\intercal \cdot \mathcal{H}_g \cdot \boldsymbol{\lambda}
\end{array}
$$

where:

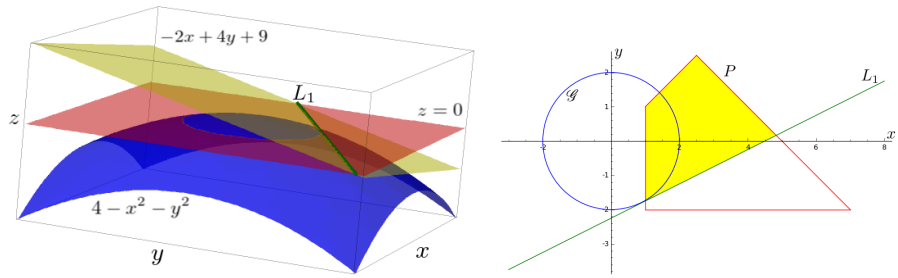
(1) $\mathcal{H}_g$ is the matrix of the coefficients of g and the $H^{\boldsymbol{I_\ell}}$ organized with respect to $\boldsymbol{\mathcal{M}}$, the sorted list of monomials that appear in the Handelman products generated by $\mathcal{I}$.
(2) the column vector $\boldsymbol{\lambda} = (\lambda_g, \lambda_1, \ldots, \lambda_q)^\intercal = (1, \lambda_1, \ldots, \lambda_q)^\intercal$ characterizes the combination of g and the $H^{\boldsymbol{I_\ell}}$. We added a constant coefficient $\lambda_g = 1$ for convenience of notations.

The product $\mathcal{H}_g \cdot \boldsymbol{\lambda}$ is a vector $\boldsymbol{\alpha} \triangleq (\alpha_0, \ldots, \alpha_{|\boldsymbol{\mathcal{M}}|-1})^\intercal$ representing the constraint $\alpha_0 + \alpha_1 x_1 + \ldots + \alpha_n x_n + \sum_{i=n+1}^{i=|\boldsymbol{\mathcal{M}}|-1} \alpha_i \cdot (\boldsymbol{\mathcal{M}})_i$ where $(\boldsymbol{\mathcal{M}})_i$ denotes the ith monomial of $\boldsymbol{\mathcal{M}}$. Since we seek an affine constraint aff we are finally interested in finding $\boldsymbol{\lambda} \in \{1\} \times (\mathbb{R}^+)^q$ such that $\mathcal{H}_g \cdot \boldsymbol{\lambda} = (\alpha_0, \ldots, \alpha_n, 0, \ldots, 0)^\intercal$. By construction, each $\boldsymbol{\lambda}$ gives an affine constraint aff that bounds g on P.

Example 2. Here is the matrix $\mathcal{H}_g$ associated to $g \triangleq 4 - x^2 - y^2$ and the Handelman products from Example 1 with respect to $\boldsymbol{\mathcal{M}} = [1, x, y, xy, x^2, y^2]$.

$$
\begin{array}{c}
\\ 1 \\ x \\ y \\ xy \\ x^2 \\ y^2
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccccccc}
g & H^{\boldsymbol{I_1}} & H^{\boldsymbol{I_2}} & H^{\boldsymbol{I_3}} & H^{\boldsymbol{I_4}} & H^{\boldsymbol{I_5}} & H^{\boldsymbol{I_6}} & H^{\boldsymbol{I_7}} & H^{\boldsymbol{I_8}} & H^{\boldsymbol{I_9}} & H^{\boldsymbol{I_{10}}} & H^{\boldsymbol{I_{11}}} & H^{\boldsymbol{I_{12}}} & H^{\boldsymbol{I_{13}}} & H^{\boldsymbol{I_{14}}} & H^{\boldsymbol{I_{15}}}
\end{array} \\
\begin{pmatrix}
4 & 1 & -1 & 2 & 0 & 5 & 1 & 4 & 0 & 25 & -2 & 0 & -5 & 0 & 10 & 0 \\
0 & 0 & 1 & 0 & 1 & -1 & -2 & 0 & 0 & -10 & 2 & -1 & 6 & 2 & -2 & 5 \\
0 & 0 & 0 & 1 & -1 & -1 & 0 & 4 & 0 & -10 & -1 & 1 & 1 & -2 & 3 & -5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & 1 & -1 & -1 & 1 & -1 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & -1 & 0 & 0 & -1 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & 1
\end{pmatrix}
\end{array}
$$

The choices $\lambda_g = \lambda_6 = \lambda_7 = 1$ and every other $\lambda_\ell = 0$ are a solution to the problem $\mathcal{H}_g \cdot \boldsymbol{\lambda} = (\alpha_0, \alpha_1, \alpha_2, 0, 0, 0)^\intercal$. We obtain $\mathcal{H}_g \cdot \boldsymbol{\lambda} = (9, -2, 4, 0, 0, 0)^\intercal$ that corresponds to $9 - 2x + 4y + 0 \times xy + 0 \times x^2 + 0 \times y^2$. Thus, $\mathit{aff} = 9 - 2x + 4y$ is a constraint that bounds g on P, as shown on Fig. 2.



(a) (b)

Fig. 2: (b) is the cut at $z=0$ of (a) in which we added the polyhedron $P \triangleq \{x-1 \geq 0,\ y+2 \geq 0,\ x-y \geq 0,\ -x-y+5 \geq 0\}$: the circle $\mathscr{G}$ of (b) appears in (a) as the intersection of the surface $z = g(x,y) \triangleq 4-x^2-y^2$ with the plane $z=0$. The polyhedral approximation of g is the inclined plane $z = \mathit{aff}(x,y) \triangleq -2x+4y+9$ that dominates g. It cuts the plane $z=0$ along the line L_1 in (a) which is reported in (b). The line L_1 is the frontier of the affine constraint $-2x+4y+9 \geq 0$. The filled area is the polyhedron $P \wedge -2x+4y+9 \geq 0$ that over-approximates $P \cap \{(x,y) \mid g(x,y) \geq 0\}$.

By construction, any solution $\boldsymbol{\lambda}$ of the problem $\mathcal{H}_g \cdot \boldsymbol{\lambda} = (\alpha_0, \ldots, \alpha_n, 0, \ldots, 0)^\intercal$ is a polyhedral constraint *aff* that bounds g on P. Among all these solutions we are only interested in the best approximations. One constraint $\mathit{aff} > g$ is better than another $\mathit{aff}' > g$ at point $(x_1, \ldots, x_n)$ if $\mathit{aff}(x_1, \ldots, x_n) < \mathit{aff}'(x_1, \ldots, x_n)$. It then appears that for a given point $(x_1, \ldots, x_n)$ we are looking for the polyhedral constraint $\mathit{aff} > g$ that minimizes its value on that point. Therefore, we define a linear minimization problem that depends on some parameters: the point $(x_1, \ldots, x_n)$ of evaluation.

Finally, finding the tightest affine forms aff_i that bound g on P with respect to a given set of indices $\mathcal{I}$ can be expressed as the *Parametric Linear Optimization Problem* (PLOP) shown on Figure 3. Such optimization problems can be solved using the parametric simplex algorithm, which is outlined in Section 4. As we shall detail later, the solution of H-PLOP is a function associating an affine form aff_i to the region of the parameter space where aff_i is optimal. The over-approximation of $P \cap (g \geq 0)$ that we return is then $\bigsqcap_i \{\boldsymbol{x} \in \mathbb{Q}^n \mid \mathit{aff}_i(\boldsymbol{x}) \geq 0\}$.

Example 3. In our running example, the objective *aff*, *i.e.*, $g + \sum_{\ell=1}^{\ell=15} \lambda_\ell H^{\boldsymbol{I_\ell}}$, is

$$4 + \lambda_1 + \lambda_2(x-1) + \lambda_3(2+y) + \lambda_4(x-y) + \lambda_5(5-x-y) + \lambda_6(1-2x) + \lambda_7(4+4y)$$
$$+\lambda_9(25-10x-10y) + \lambda_{10}(2x-y-2) + \lambda_{11}(y-x) + \lambda_{12}(6x+y-5) + \lambda_{13}(2x-2y)$$
$$+\lambda_{14}(10-2x+3y) + \lambda_{15}(5x-5y).$$

In practice we use this presentation (without α) which exhibits the parametric coefficients in x, y of each variable λ. Nonlinear monomials do not appear since the problem imposes cancelling the non-linear part of $g + \sum_{\ell=1}^{\ell=15} \lambda_\ell H^{\boldsymbol{I_\ell}}$, *i.e.* $xy(-2\lambda_8+2\lambda_9+\lambda_{10}-\lambda_{11}-\lambda_{12}+\lambda_{13}-\lambda_{14})+x^2(-1+\lambda_6+\lambda_8+\lambda_9+\lambda_{11}-\lambda_{12}-\lambda_{15})+y^2(-1+\lambda_7+\lambda_8+\lambda_9-\lambda_{13}-\lambda_{14}+\lambda_{15})$. The solutions of the problem are the vectors $\boldsymbol{\lambda}$ that minimize the objective and cancel the coefficients of xy, x^2 and y^2.

Given a set of indices $\mathcal{I} \triangleq \{\boldsymbol{I}_1, \ldots, \boldsymbol{I}_q\}$,
minimize *aff ,i.e.,* $\alpha_0 + \alpha_1 x_1 + \ldots + \alpha_n x_n$, *also equal to* $g + \sum_{\ell=1}^{\ell=q} \lambda_\ell H^{\boldsymbol{I_\ell}}$ ***under the constraints***

$$\begin{cases} \mathcal{H}_g \cdot (\lambda_g, \lambda_1, \ldots, \lambda_q)^\intercal = (\alpha_0, \ldots, \alpha_n, 0, \ldots, 0)^\intercal \\ \lambda_g = 1, \quad \lambda_\ell \geq 0, \ \ell = 1..q \end{cases} \tag{H-PLOP}$$

where $\lambda_1, \ldots, \lambda_q$ *are the decision variables of the* PLOP*;* $x_1, \ldots, x_n$ *are the parameters ; and* $\alpha_0, \ldots, \alpha_n$ *are kept for the sake of presentation ; in practice they are substituted by their expression issued from* $\mathcal{H}_g \cdot \boldsymbol{\lambda}$.

Fig. 3: Linearization as a Parametric Linear Optimization Problem

4 The Parametric Simplex Algorithm

We use the simplex algorithm for parametric objective functions to find the solutions of the previous H-PLOP problem. This section explains how we obtain the output polyhedron over-approximating $P \cap g \geq 0$ from the solutions of H-PLOP. We assume the reader is familiar with the simplex algorithm (see [8] for an introduction) and we sketch the broad outlines of the parametric simplex algorithm (see [12,33] for more details).

Principle of the Algorithm. The standard simplex algorithm is used to find the optimal value of an affine function – called the objective – on a space delimited by affine constraints, which is thus a polyhedron. More precisely, it solves linear problems of the form

$$\textbf{\textit{minimize the objective}} \sum_{i=1}^{i=q} \lambda_i \cdot c_i \ \textbf{\textit{s.t.}} \ A \cdot \boldsymbol{\lambda} = \mathbf{0}, \ \boldsymbol{\lambda} \geq 0$$

where $A \in M_{p,q}(\mathbb{Q})$ is a matrix and the constants $c_i \in \mathbb{Q}$ define the costs associated to each decision variable $(\lambda_1, \ldots, \lambda_q) = \boldsymbol{\lambda}$. To decrease the objective value, recalling that each variable λ_i is nonnegative, a step in the standard simplex algorithm, called a *pivot*, consists in finding a negative coefficient c_i in the objective function and in decreasing the value of the associated variable λ_i as much as the constraints remain satisfied. The pivot operation modifies both the costs of the objective function and the constraints. The optimal value is reached when every c_i is nonnegative, meaning that the objective value cannot be decreased anymore.

The parametric simplex algorithm solves linear problems of the form

$$\textbf{\textit{minimize the objective}} \sum_{i=1}^{i=q} \lambda_i \cdot c_i(x_1, \ldots, x_n) \ \textbf{\textit{s.t.}} \ A \cdot \boldsymbol{\lambda} = \mathbf{0}, \ \boldsymbol{\lambda} \geq 0$$

where c_i are now affine functions from parameters $(x_1, \ldots, x_n)$ to $\mathbb{Q}$. As in the standard simplex we seek for a pivot to decrease the objective value, *i.e.* a negative coefficient in the objective function. In general the sign of a parametric coefficient, say c_i, is unknown. The algorithm then explores two branches: one in which c_i is considered as nonnegative and we move to the next coefficient c_{i+1} ; and another branch in which c_i is assumed to be negative and we perform a pivot on the associated variable λ_i exactly as in the standard version. The exploration

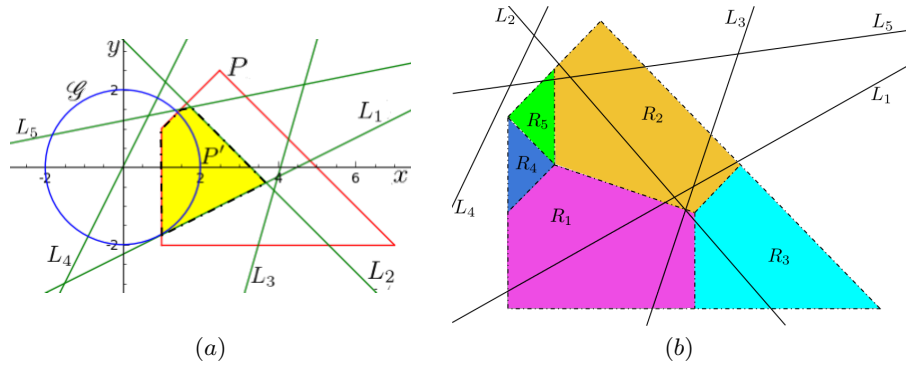
of a branch stops when the conjunction of the assumptions is unsatisfiable (the branch is then discarded); or when it implies that all the updated parametric coefficients are nonnegative, meaning that an optimum is reached. Both tests of unsatisfiability and implication are polyhedral operations performed by the VPL.

The result of the solver is a decision tree: the values of the decision variables $\boldsymbol{\lambda}$ at leaves give optima of the parametric objective ; the conjunction of the assumptions along a branch defines its *region of relevance*, it is a polyhedron in the parameter space. Our solver implements this algorithm in OCAML and works with rationals instead of floating points. It borrows a few optimizations from the PIP algorithm [19] which was developed for the dual case where parameters are in the right-hand side of the constraints, *i.e.* $A \cdot \boldsymbol{\lambda} = \boldsymbol{b}(x_1, \ldots, x_n)$.

Application to Handelman's Linearization. Back to our running example, we obtain the best polyhedral approximations of g by running our parametric simplex on H-PLOP where $(\lambda_\ell)_{\ell=1..q}$ are decision variables, $H^{\boldsymbol{I_\ell}}(x_1, \ldots, x_n)$ are parametric coefficients, x_i are parameters and the matrix A is made of the rows of $\mathcal{H}_g$ corresponding to monomials of degree > 1 (the last three rows of $\mathcal{H}_g$ in Example 2). We obtain a decision tree with 5 optimal solutions $\boldsymbol{\lambda}$ at leaves. Each of them is interpreted as constraint $\mathit{aff}(x_1, \ldots, x_n) \geq 0$ where $\mathit{aff}(\boldsymbol{x}) = g(\boldsymbol{x}) + \sum_{\ell=1}^{\ell=q} \lambda_\ell H^{\boldsymbol{I_\ell}}(\boldsymbol{x})$. These 5 constraints appear on Fig. 4(a) as the lines L_1 to L_5. Their conjunction with P forms the polyhedron P' which over-approximates $P \cap (g \geq 0)$.

Useless constraint detection. Fig. 4(a) reveals that L_3 and L_4 are useless since they do not intersect P'. This is not due to the parametric simplex: it happens when a constraint aff_j does not cross the plane $z = 0$ on its region of relevance R_j. Fig. 4(b) shows the region of relevance of each constraint. This remark leads us to a criterion to detect useless aff_i during exploration. It requires some explanations. Note that the output polyhedron $P' \triangleq P \sqcap (\bigsqcap_{j=1}^{j=k} \mathit{aff}_j \geq 0)$ is equal to the set $\bigcup_{j=1}^{j=k} \big(R_j \sqcap \mathit{aff}_j \geq 0\big)$. That can be proved by reasoning on sets, using (1) distributivity and (2) simplification, exploiting two consequences of the parametric simplex: (1) by construction of the exploration tree, the regions $(R_i)_{i=1}^{i=k}$ form a partition of P; (2) if $i \neq j$, $R_i \sqcap (\mathit{aff}_i \geq 0) \sqcap (\mathit{aff}_j \geq 0) = R_i \sqcap (\mathit{aff}_i \geq 0)$ since $\mathit{aff}_j \geq \mathit{aff}_i$ on R_i. Indeed, we asked the parametric simplex to seek for minimal affine forms.

Now, let us study the equality $P' = \bigcup_{j=1}^{j=k} \big(R_j \sqcap \mathit{aff}_j \geq 0\big)$: when the sign of aff_i is negative on its region of relevance R_i, then $(R_i \sqcap \mathit{aff}_i \geq 0) = \emptyset$ and this term vanishes from the union. Therefore, such an aff_i has no impact on P'. We draw upon this remark to design an algorithm that early detects useless exploration. The exploration of a new branch starts with the examination of the possible pivots. For minimization problems, the pivoting operation lowers the objective and the new region is a subpart of the previous one. Therefore, if all pivots give an objective that is negative on the current region, every optimum aff generated through this branch will be negative, thus useless; we simply cut this branch. Our experiments are conducted with the parametric simplex algorithm of Section 4 improved with this elimination criterion.



(a) (b)

Fig. 4: (a) The polyhedron $P' = P \sqcap \{ L_1 \geq 0, \ldots, L_5 \geq 0 \}$ is the over-approximation of $P \cap (g \geq 0)$ computed by our linearization without detection of useless constraints. P' is delimited by P and the constraints L_1, L_2, L_5 returned by the parametric simplex; L_3 and L_4 are useless: L_3 is detected by our criterion. The redundancy of L_4 cannot be detected before the intersection with P. (b) Each constraint L_i is the optimum associated to a region R_i of P. Our criterion eliminates L_3 since it is negative on R_3.

5 Heuristics and Certificates

We previously assumed a given set of Handelman products to be considered in H-PLOP; our implementation actually uses Schweighofer products ($S^{\boldsymbol{I}}$), which generalize Handelman's ones as shown by Theorem 2 below. We shall now describe the oracle that generates them together with a certificate of nonnegativity, then the heuristics it uses.

Theorem 2 (Schweighofer, 2001) *Let $P = \{ C_1 \geq 0, \ldots, C_p \geq 0 \}$ be a polytope where each C_i is an affine polynomial over $\boldsymbol{x} = (x_1, \ldots, x_n)$. Let $g_{p+1}, \ldots, g_q$ be polynomials. Then $g(\boldsymbol{x}) > 0$ on $P \cap \{ g_{p+1} \geq 0, \ldots, g_q \geq 0 \}$ if and only if*

$$g = \lambda_0 + \sum_{\boldsymbol{I} \in \mathbb{N}^q} \lambda_{\boldsymbol{I}} \cdot S^{\boldsymbol{I}}, \quad \lambda_0 \in \mathbb{R}^{*+},\ \lambda_{\boldsymbol{I}} \in \mathbb{R}^+$$
$$\text{where } S^{(i_1,\ldots,i_q)} = C_1^{i_1} \cdots C_p^{i_p} \cdot g_{p+1}^{i_{p+1}} \cdots g_q^{i_q}.$$

Schweighofer products are products of polyhedral constraints of P and polynomials $(g_i)_{i=p+1}^{i=q}$. They are obviously nonnegative on the set $P \cap \{ g_{p+1} \geq 0, \ldots, g_q \geq 0 \}$. From a certification viewpoint, the key property of the polynomials resulting from Handelman or Schweighofer products is their nonnegativity on the input polyhedron. Therefore, heuristics must attach to each product a nonnegativity certificate as its representation in the OCAML/COQ type nonNegCert given below. The COQ checker contains the proof that this type only yields nonnegative polynomials by construction.

```
type nonNegCert = C of ℕ
  | Square of polynomial
  | Power of ℕ * nonNegCert
  | Product of nonNegCert list
```

with

$$[\![C(i)]\!] = C_i \geq 0 \ \textit{of } P$$
$$[\![\mathsf{Square}\,(p)]\!] = p^2 \geq 0 \ \ \forall p \in \mathbb{Q}[\boldsymbol{x}]$$
$$[\![\mathsf{Power}\,(n, S)]\!] = S^n \ \textit{with } S \geq 0$$
$$[\![\mathsf{Product}\,(L)]\!] = \Pi_{S \in L}\ S \geq 0$$

Design of the oracle. The oracle treats the input polynomial g as the set $\mathscr{M}$ of its monomials and maintains a set $\mathscr{M}_C$ of already-canceled monomials. Each heuristic looks for a monomial m in $\mathscr{M}$ it can apply to, checks that it doesn't belong to $\mathscr{M}_C$ and generates a product S or H for it. Monomial m is then added to $\mathscr{M}_C$ and the monomials of S that are different from m are added to $\mathscr{M}$. The oracle finally returns a list of couples formed of a product H or S and its certificate of nonnegativity. The heuristics are applied according to their priority. The most basic of them consists in taking every Handelman product whose degree is smaller than or equal to that of g. If solving H-PLOP fails with these products, we increase the maximum degree up to which all the products are considered. Theorem 1 ensures eventual success. However, the number of products quickly becomes so large that this heuristic is used as a last resort.

Targeted heuristics. The following heuristics aim at finding either Handelman products $H^{\boldsymbol{I}}$ or Schweighofer products $S^{\boldsymbol{I}}$ which cancel a given nonlinear monomial m. Besides a monomial canceling m, a product may contain nonlinear monomials which need to be eliminated. The heuristics guarantee that these monomials are of smaller degree than m when the polyhedron is bounded, thereby ensuring termination. Otherwise, they try to limit the degree of these additional monomials as much as possible, so as to make them easier to cancel. As before, we consider an input polyhedron $\{C_1 \geq 0, \ldots, C_p \geq 0\}$ with $C_i = \sum_{j=1}^{n} a_{ij}x_j + a_{i0}$, where the x_j's are program variables and the a_{ij}'s are constants in $\mathbb{Q}$. We wish to cancel monomial $m \triangleq c_m \times x_1^{e_1} \cdots x_n^{e_n}$, with $c_m \in \mathbb{Q}$.

Extraction of even powers. This heuristic builds on squares being always nonnegative to apply Schweighofer's theorem in an attempt to simplify the problem. The idea is to rewrite m into $m = m' \times (x_1^{\epsilon_1} \ldots x_n^{\epsilon_n})^2$ where $m' \triangleq c_m \times x_1^{\delta_1} \ldots x_n^{\delta_n}$, with $\delta_j \in \{0, 1\}$. The heuristic recursively calls the oracle in order to find a product S canceling m'. Then, $S \times (x_1^{\epsilon_1} \ldots x_n^{\epsilon_n})^2$ cancels the monomial m. If W_S is the nonnegativity certificate for S, then $\mathsf{Product}\,[W_S; \mathsf{Square}\,(x_1^{\epsilon_1} \ldots x_n^{\epsilon_n})]$ is that of the product.

Simple products. Consider a monomial $m = c_m \times x_1 \cdots x_n$ where $c_m \in \mathbb{Q}$, as can be produced by the previous heuristic. We aim at finding a Schweighofer product S that cancels m, and such that every other monomial of S has a degree smaller than that of m. We propose an analysis based on intervals, expressing S as a product of *variable bounds*, *i.e.* $x_j \in [l_j, u_j]$ where $l_j,\ u_j \in \mathbb{Q}$. For each variable x_j, we may choose either constraint $x_j + l_j \geq 0$ or $-x_j + u_j \geq 0$, so that the product of the chosen constraints contains $x_1 \cdots x_n$ with the appropriate sign. Moreover, other monomials of this product are ensured to have a degree smaller than that of m. The construction of a product of bounds is guided by the following concerns.

- The sign of the canceling monomial is to be opposite to that of m.
- The bounds that are available in the input constraints are used in priority. It is possible to call the VPL to deduce additional bounds on any variable

from the input constraints. However, finding a new bound requires solving a linear problem.
- The selected bounds should exist, which is not necessarily the case if the input polyhedron is not a polytope. If too many bounds don't exist, the heuristic fails.

Thanks to Farkas' lemma [12, Th. 2.14], each implied bound on a variable ($x_j + l_j$ or $-x_j + u_j$) can be expressed as a nonnegative linear combination of the input constraints, *i.e.* $\sum_{i=1}^{p} \beta_{ij} C_i$ for some $\beta_{ij} \geq 0$ solutions of a linear problem. The combination reduces to C_i if C_i is already a constraint of the input polyhedron P. The resulting product of bounds can then be expressed as follows.

$$\prod_{j \in L} (x_j + l_j) \times \prod_{j \in U} (-x_j + u_j) = \prod_{j \in L \cup U = \{1,\ldots,n\}} \big(\sum_{i=1}^{p} \beta_{ij} \cdot C_i \big), \qquad \beta_{ij} \geq 0$$

The right-hand side expression is then refactorised with the C_i's kept symbolic, so that the Handelman products appear. This case is illustrated in Example 4.

Example 4. We illustrate the behavior of the oracle and the satisfiability test on the polynomial $g = y^2 - x^2y + xy - 85$ and still the same polytope $P = \{ (C_1)\ x - 1 \geq 0, (C_2)\ y + 2 \geq 0, (C_3)\ x - y \geq 0,\ (C_4)\ 5 - x - y \geq 0 \}$. The oracle starts with $\mathscr{M} = \{ xy, -x^2y, y^2 \}$ and processes the monomials in order.

(xy) For eliminating xy, the simple product heuristic uses constraint $(C_1)\ x - 1 \geq 0$ and the combination $(C_1) + (C_4) = (x - 1) + (-x - y + 5)$ which entails $-y + 4 \geq 0$. Their product $(x-1)(-y+4) = -xy + 4x + y - 4$ cancels xy and the development $C_1 \cdot (C_1 + C_4) = C_1^2 + C_1C_4$ reveals the useful Handelman products: $H_1 \triangleq C_1^2 = x^2 - 2x + 1$ and $H_2 \triangleq C_1C_4 = -x^2 - xy + 6x + y - 5$. They are returned with their certificates of nonnegativity: $\mathsf{Power}\,(2, C_1)$ and $\mathsf{Product}\,[C_1; C_4]$. Then, xy is added to $\mathscr{M}_C$ as well as the new monomials x^2 and $-x^2$: They are not placed in $\mathscr{M}$ since opposite monomials cancel each other.

($-x^2y$) The heuristic for squares splits the term $-x^2y$ into $m' \times x^2$ and lets the oracle deal with $m' \triangleq -y$. The simple product heuristic reacts by looking for a constraint with the term $+y$ and as few variables as possible: $(C_2)\ y + 2 \geq 0$ fulfills these criteria. The calling heuristic builds the Schweighofer product $S_3 \triangleq x^2 \cdot C_2 = x^2y + 2x^2$ that cancels $-x^2y$, and returns S_3 with its certificate of nonnegativity $\mathsf{Product}\,[\mathsf{Square}\,(x); C_2]$. Then, the oracle removes x^2y from the working set and places it into the set of cancelled monomials.

(y^2) The heuristic on squares cannot produce $y^2 \times (-1)$ with a certificate of nonnegativity for -1. The last heuristic is then triggered and finds two Handelman's products that generate $(-y^2)$: $H_4 \triangleq C_2C_3 = (y+2)(x-y) = xy - y^2 + 2x - 2y$ and $H_5 \triangleq C_2C_4 = (y+2)(5-x-y) = 5y - xy - y^2 + 10 - 2x - 2y$. H_4 is prefered since it does not introduce a new monomial – indeed $xy \in \mathscr{M}_C$ – whereas H_5 would add $-y^2$ to the working set $\mathscr{M}$.

Finally the oracle returns the four polynomials with their certificates. The expanded forms of H_1, H_2, S_3, H_4 are installed in the matrix $\mathcal{H}_g$ and are each

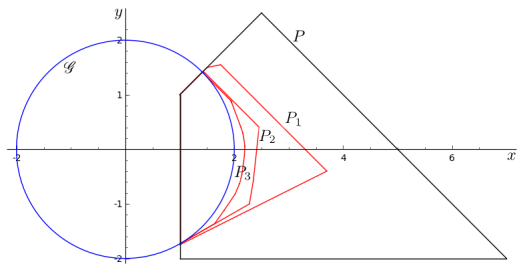


Fig. 5: The polytopes resulting of 3 iterations of Handelman's linearization: $P_0 = P$, $P_i = \mathsf{HL}\,(P_{i-1}, 4 - x^2 - y^2 \geq 0)$. *$P_1$, P_2 and P_3 are respectively composed of 5, 9 and 36 constraints.*

associated with a decision variable $\lambda_1, \ldots, \lambda_4$. The parametric simplex computes all the positive, minimal, affine constraints *aff* of the form $1 \cdot g + \lambda_1 \cdot H_1 + \lambda_2 \cdot H_2 + \lambda_3 \cdot S_3 + \lambda_4 \cdot H_4$. With such few products, it returns only one affine constraint $\mathit{aff} = g + 2H_2 + H_3 + H_4 = 13x + y - 95$ from which we build a polyhedral over-approximation of the set $P \cap (g \geq 0)$ as $P \sqcap \mathit{aff} \geq 0$. The VPL reveals that this polyhedron is empty, meaning that $P \wedge (g \geq 0)$ is unsatisfiable.

6 Implementation and Experiments

We implemented our linearization as part of the VPL. The linearization process has two parts: an OCAML oracle, defined in Section 5, uses heuristics to select the most promising Handelman-Schweighofer products $S_1, \ldots, S_q$, then it runs the parametric simplex to find coefficients $\lambda_1, \ldots, \lambda_q$ such that $g + \sum \lambda_i S_i$ is affine. The result is fed into a checker implemented and proved correct in COQ. It guarantees in three steps that *aff* is an affine form and dominates g on P: (1) it verifies that *aff* is affine; (2) the proof of $\sum \lambda_i S_i \geq 0$ boils down to "sums and products of nonnegative reals are nonnegative" using the nonnegativity certificates W_i provided by the oracle; (3) it checks that the two polynomials *aff* and $g + \sum \lambda_i S_i$ are equal in expanded form using the internals of the `ring` tactic. We pay some care to efficiency by caching translations of polynomials from certificates to the expanded form to reduce the overhead of certificate checking. The architecture of the checker is detailed in a previous work [33].

Increasing precision. We show on Fig. 5 the results of Handelman's linearization on the running example. We chose the subset $\{H^{\boldsymbol{I_1}}, \ldots, H^{\boldsymbol{I_{15}}}\}$ from Example 1, meaning that we are faced with a 15-variable linear problem. Precision can be increased without degree elevation by iterating Handelman's linearization (HL): $P_0 = P$, $P_{i+1} = \mathsf{HL}\,(P_i, g \geq 0)$. The linearization operator of the VPL computes this sequence until reaching a fixpoint, *i.e.* $P_{k+1} = P_k$, or a time limit. The sequence is decreasing with respect to inclusion since $\mathsf{HL}\,(P_i, g \geq 0) = P_i \sqcap \bigwedge_i \mathit{aff}_i \geq 0$ is by construction included in P_i.

Showing emptiness of nonlinear sets. A SMT-solver for nonlinear real arithmetic using the DPLL(T) architecture enumerates conjunctions of nonlinear inequalities, each of which having to be tested for satisfiability. We show the unfeasibility of the conjunction of $C_1 \geq 0, \ldots, C_p \geq 0$ and nonlinear ones $g_1 \geq 0, \ldots, g_q \geq 0$ by computing the sequence of approximations: $P_0 = \{C_1 \geq 0, \ldots, C_q \geq 0\}$, $P_{i+1} = \mathsf{HL}\,(P_i, g_i \geq 0)$. The polynomials are added one after the other, meaning that g_{i+1} is linearized with respect to the previous polyhedral approximation P_i. If at some point $P_k = \emptyset$, it means that the conjunction is unsatisfiable, as our approximation is sound. Otherwise, as it is not complete, we cannot conclude. Such a procedure can thus be used to soundly prune branches in DPLL(T) search. Furthermore, the subset of constraints appearing in the products used in the emptiness proof is unsatisfiable, and thus the negation of its conjunction may be used as a learned clause.

Although our contribution applies to both static analysis and SMT solving, we felt that performing our experimental evaluation with SMT-solvers was better suited: the SMT community has a standard set of nonlinear benchmarks from SMT-LIB, which the static analysis community is missing. Therefore, we experimented with conjunctions arising from deciding formulas from the Quantifier-Free Nonlinear Real Arithmetic (QF_NRA) benchmark, from SMT-LIB 2014 [2]. These conjunctions, that we know to be unsatisfiable, are mostly coming from approximations of transcendental functions as polynomial expressions. We added our linearization algorithm as a theory solver for the SMT-solver CVC4 [15]. The calls to our linearization follow a factorization step, where for instance polynomial guards such as $x^2 - y^2 \geq 0$ are split into two cases ($x+y \geq 0 \wedge x-y \geq 0$ and $x+y \leq 0 \wedge x-y \leq 0$), in order to give more constraints to the input polyhedron.

The comparison of our contribution with the state of the art SMT-solvers Z3 [13], Yices2 [16], SMT-RAT [10] and raSat [28] was done on the online infrastructure StarExec [39]. Fig. 6 is a cactus plot showing the number of benchmarks proved unsatisfiable depending on time. It illustrates that linearization based on Handelman's representation, implemented as a non-optimized prototype, gives fast answers and that its results are precise enough in many cases. Note that our approach also provides an easy-to-verify certificate, as opposed to the cylindrical algebraic decomposition implemented in Z3 for example. Indeed, if the answer of the VPL is that the final polyhedral approximation is empty, then the nonzero coefficients in the solution $\boldsymbol{\lambda}$ of the parametric problem H-PLOP give a list of sufficient Schweighofer products. Together with the nonlinear guards, the conjunction of the original constraints involved in these products are actually sufficient for emptiness. As mentioned above, in a SMT-solver the negation of this conjunction may be used as a *learned theory lemma*. However, due to engineering issues we have not been able to fully integrate this procedure into CVC4 by sending back minimized learned lemmas. Over a total of 4898 benchmarks, adding our method (represented in the figure as curve *cvc4+vpl*) allows CVC4 to show the unsatisfiability of 1030 more problems. Failure in showing emptiness may come from strict constraints since up to now, our solver considers each inequality as nonstrict.

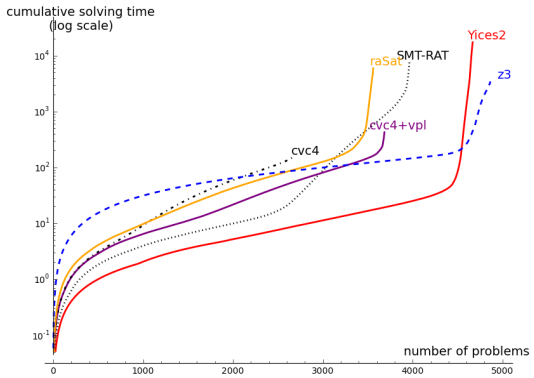


Fig. 6: Comparison between CVC4+VPL and other SMT*-solvers on Quantifier-Free Nonlinear Real Arithmetic benchmarks.*

7 Conclusions and Future Work

We presented a new approach to the linear approximation of multivariate polynomials, based on Handelman's and Schweighofer's theorems, and implemented it in the Verimag Verified Polyhedra Library (VPL) as an operator of the abstract domain of polyhedra. A verifier implemented and proved correct in COQ can optionally check its results.

The approach is directly usable in static analysis by abstract interpretation: besides linear expressions, the VPL now accepts polynomials as well. Apart from handmade examples [33], we actually did not find programs manipulating integers where the linearization improves the global analysis result: non-linearity is too sparse in such programs. We believe that it could have an impact on the analysis of floating-point computations where polynomials appear more naturally in programs for approximating transcendental functions and in the analysis of the round-off errors [3]. Work in that direction is planned for the very near future but supporting this claim still requires some work on the integration of the VPL into a mature analyzer for floating-point programs, the treatment of round-off errors and some certification effort. The VPL can already deal with strict inequalities over the rationals but the algorithms are not yet certified (the enlargement of any strict inequality $< n$ over the integers to $\leq n-1$, is not valid for polyhedra over the rational field).

Our approach already proved to be useful in satisfiability modulo theory solving. A simple coupling of our prototype, implemented in OCAML, with the competitive SMT-solver CVC4 improved notably the performance of that solver on nonlinear arithmetic.

In contrast to cylindrical algebraic decomposition, which is a complete approach, our method may fail to prove a true property. However, it provides easy-to-check certificates for its results.

From a polynomial guard $g \geq 0$ and an input polyhedron P, our algorithm operates in two phases. The first selects products of constraints of P which are likely to cancel nonlinear monomials from g. The second phase uses parametric programming to explore the linear combinations of these products yielding an affine form which bounds g. Both phases offer room for improvement.

(1) Blindly including all products of degree n is exponential in n and many of them may be useless. This is why we developed an oracle procedure using selection heuristics to obtain good precision at reasonable cost. In a future refinement of this work, an incremental approach could grow the set of products, using feedback from the solver about missing monomials in cancellations.

(2) Our parametric linear solver currently relies on the parametric variant of the simplex algorithm. The latter subdivides regions of relevance, leading to multiple copies of each solution, which makes the exploration more expensive than it should be. We are now working on more efficient exploration algorithms, following previous work by Jones *et al.* [25].

References

1. R. Bagnara, P. M. Hill, and E. Zaffanella. 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. Tool available at `www.cs.unipr.it/ppl/`.
2. C. Barrett, A. Stump, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2010.
3. D. Boland and G. A. Constantinides. Bounding Variable Values and Round-Off Effects Using Handelman Representations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(11):1691–1704, 2011.
4. S. Boulmé and A. Maréchal. Refinement to certify abstract interpretations, illustrated on linearization for polyhedra. In *Interactive Theorem Proving (ITP)*, volume 9236 of *LNCS*, pages 100–116. Springer, August 2015.
5. L. Chen, A. Miné, J. Wang, and P. Cousot. Interval polyhedra: An abstract domain to infer interval linear relationships. In *Static Analysis Symposium (SAS)*, volume 5673 of *LNCS*, pages 309–325. Springer, December 2009.
6. S. Chevillard, M. Joldeş, and C. Lauter. Sollya: An environment for the development of numerical codes. In K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 28–31, Heidelberg, Germany, September 2010. Springer.
7. S. Chevillard, M. Joldeş, and C. Lauter. Certified and fast computation of supremum norms of approximation errors. In *Computer Arithmetic (ARITH)*, pages 169–176. IEEE Computer Society, June 2009.
8. V. Chvatal. *Linear Programming*. Series of books in the Mathematical Sciences. W. H. Freeman, 1983.
9. P. Clauss, F. J. Fernandez, D. Gabervetsky, and S. Verdoolaege. Symbolic polynomial maximization over convex sets and its application to memory requirement estimation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(8):983–996, August 2009.

10. F. Corzilius, U. Loup, S. Junges, and E. Ábrahám. SMT-RAT: An SMT-Compliant Nonlinear Real Arithmetic Toolbox. In A. Cimatti and R. Sebastiani, editors, *Theory and Applications of Satisfiability Testing (SAT)*, volume 7317 of *Lecture Notes in Computer Science*, pages 442–448. Springer Berlin Heidelberg, 2012.
11. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *ACM Principles of Programming Languages (POPL)*, pages 84–97. ACM Press, January 1978.
12. G. B. Dantzig and M. N. Thapa. *Linear Programming 2: Theory and Extensions*. Operations Research. Springer, 2003.
13. L. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In C. Ramakrishnan and J. Rehof, editors, *Tools and Algorihtms for the Construction and Analysis of Systems (TACAS)*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer Berlin Heidelberg, 2008.
14. L. M. de Moura and D. Jovanovic. A model-constructing satisfiability calculus. In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *VMCAI*, volume 7737 of *LNCS*, pages 1–12. Springer, 2013.
15. M. Deters, A. Reynolds, T. King, C. Barrett, and C. Tinelli. A tour of CVC4: How it works, and how to use it. In *Proceedings of the 14th Conference on Formal Methods in Computer-Aided Design*, FMCAD '14, pages 4:7–4:7, Austin, TX, 2014. FMCAD Inc.
16. B. Dutertre. Yices 2.2. In A. Biere and R. Bloem, editors, *Computer Aided Verification (CAV)*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer, July 2014.
17. B. Dutertre and L. de Moura. A fast linear-arithmetic solver for DPLL(T). In *Computer-aided verification (CAV)*, volume 4144 of *LNCS*, pages 81–94. Springer, 2006.
18. B. Dutertre and L. De Moura. Integrating simplex with DPLL(T). Technical Report SRI-CSL-06-01, SRI International, computer science laboratory, 2006.
19. P. Feautrier. Parametric integer programming. *RAIRO Recherche opérationnelle*, 22(3):243–268, September 1988.
20. A. Fouilhé, D. Monniaux, and M. Périn. Efficient generation of correctness certificates for the abstract domain of polyhedra. In *Static Analysis Symposium (SAS)*, volume 7935 of *LNCS*, pages 345–365. Springer, 2013.
21. H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. DPLL(T): Fast decision procedures. In *Computer Aided Verification (CAV)*, volume 3114 of *LNCS*, pages 175–188. Springer, 2004.
22. N. Halbwachs. *Détermination automatique de relations linéaires vérifiées par les variables d'un programme*. Thèse de doctorat de troisième cycle, Université de Grenoble, March 1979.
23. D. Handelman. Representing polynomials by positive linear functions on compact convex polyhedra. *Pacific Journal of Mathematics*, 132(1):35–62, 1988.
24. B. Jeannet and A. Miné. APRON: A library of numerical abstract domains for static analysis. In *Computer Aided Verification (CAV)*, volume 5643 of *LNCS*, pages 661–667, 2009. Tool available at `apron.cri.ensmp.fr/library/`.
25. C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski. Lexicographic perturbation for multiparametric linear programming with applications to control. *Automatica*, 2007.
26. J.-H. Jourdan, V. Laporte, S. Blazy, X. Leroy, and D. Pichardie. A formally-verified C static analyzer. In *ACM Principles of Programming Languages (POPL)*, pages 247–259. ACM Press, January 2015.

27. D. Jovanovic and L. M. de Moura. Solving non-linear arithmetic. In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning (IJCAR)*, volume 7364 of *LNCS*, pages 339–354. Springer, 2012.
28. T. V. Khanh, X. Vu, and M. Ogawa. rasat: SMT for polynomial inequality. In *Proceedings of the 12th International Workshop on Satisfiability Modulo Theories, SMT 2014, affiliated with the 26th International Conference on Computer Aided Verification (CAV 2014), the 7th International Joint Conference on Automated Reasoning (IJCAR 2014), and the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT 2014), Vienna, Austria, July 17-18, 2014.*, page 67, 2014.
29. J.-L. Krivine. Anneaux préordonnés. *Journal d'analyse mathématique*, 12:307–326, 1964.
30. J. B. Lasserre. *Moments, Positive Polynomials and Their Applications*, volume 1 of *Imperial College Optimization Series*. Imperial College Press, 2010.
31. V. Loechner and D. K. Wilde. Parameterized polyhedra and their vertices. *International Journal of Parallel Programming*, 2(6):525–549, December 1997. Tool available at `icps.u-strasbg.fr/polylib/`.
32. A. Maréchal and M. Périn. Three linearization techniques for multivariate polynomials in static analysis using convex polyhedra. Technical Report 7, Verimag, July 2014.
33. A. Maréchal and M. Périn. A linearization technique for multivariate polynomials using convex polyhedra based on Handelman's theorem. In *Journées Francophones des Langages Applicatifs (JFLA)*, January 2015.
34. A. Miné. Symbolic methods to enhance the precision of numerical abstract domains. In *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 3855 of *LNCS*, pages 348–363. Springer, January 2006.
35. C. Muñoz and A. Narkawicz. Formalization of a representation of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, 51(2):151–196, August 2013.
36. P. Néron. *A Quest for Exactness: Program Transformation for Reliable Real Numbers*. PhD thesis, École Polytechnique, Palaiseau, France, 2013.
37. A. Prestel and C. N. Delzell. *Positive Polynomials: From Hilbert's 17th Problem to Real Algebra*. Springer-Verlag, June 2001.
38. M. Schweighofer. An algorithmic approach to Schmüdgen's Positivstellensatz. *Journal of Pure and Applied Algebra*, 166(3):307–319, January 2002.
39. A. Stump, G. Sutcliffe, and C. Tinelli. Starexec: a cross-community infrastructure for logic solving. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Proceedings of the 7th International Joint Conference on Automated Reasoning*, Lecture Notes in Artificial Intelligence. Springer, 2014.