Formal Proofs for Nonlinear Optimization

VICTOR MAGRON
Circuits and Systems Group, Imperial College, London, UK
XAVIER ALLAMIGEON
INRIA and CMAP, École Polytechnique, CNRS, Palaiseau, France
STÉPHANE GAUBERT
INRIA and CMAP, École Polytechnique, CNRS, Palaiseau, France
and
BENJAMIN WERNER
LIX, École Polytechnique, CNRS, Palaiseau, France

We present a formally verified global optimization framework. Given a semialgebraic or transcendental function f and a compact semialgebraic domain K, we use the nonlinear maxplus template approximation algorithm to provide a certified lower bound of f over K. This method allows to bound in a modular way some of the constituents of f by suprema of quadratic forms with a well chosen curvature. Thus, we reduce the initial goal to a hierarchy of semialgebraic optimization problems, solved by sums of squares relaxations. Our implementation tool interleaves semialgebraic approximations with sums of squares witnesses to form certificates. It is interfaced with Coq and thus benefits from the trusted arithmetic available inside the proof assistant. This feature is used to produce, from the certificates, both valid under-approximations and lower bounds for each approximated constituent. The application range for such a tool is widespread; for instance Hales' proof of Kepler's conjecture yields thousands of multivariate transcendental inequalities. We illustrate the performance of our formal framework on some of these inequalities as well as on examples from the global optimization literature.

1. INTRODUCTION

1.1 Problems Involving Computer Assisted Proofs

This work is about one particular combination of secure formal proofs with fast mechanical computations: we want the computer to automatically determine precise numerical bounds of algebraic expressions, while retaining the safety of formal proofs - in our case, the Coq proof system.

On one hand, since their conception, computers have been used as fast calculators, whose speed allows the mathematician to access knowledge which would be out of reach without the machine. On the other hand, it is now common practice since a long while to use the computer as a rigorous *censor*, which checks the validity of a mathematical development down to its formal steps in a given logical formalism; this is the task of *proof systems* since the 1960s. Combining these two tasks, as in this work, is more recent.

The programming language provided inside the formalism of Coq can be used in sophisticated ways. In particular, it allows to build decision procedures or per-

The research leading to these results has received funding from the European Union's 7^{th} Framework Programme under grant agreement no. 243847 (ForMath).

form automatized reasoning, thus to prove classes of propositions in a systematic and efficient fashion. Because this involves formalizing a fragment of the logic in Coq itself, this technique is called *computational reflection* and was introduced in [BRB95] (see also [BM81] for details about reflection).

The fact that complex computations can also take place inside the proof-system was first used for some proof automation like the successive versions of Coq's ring tactic [Bou97, GM05] to check polynomial equalities (actually we happen to use the current version of this tactic in the present work). Other applications include verifying large numbers' primality [GTW06], checking witnesses from SAT/SMT solvers [AFG+11] or hardware verification [PM96].

Recently, proof-checkers embedded with computational features were particularly highlighted by allowing the formal checking of results whose proofs are fundamentally computational, like the four-color theorem [Gon08] or Kepler's conjecture.

Kepler's conjecture is one of the eventual motivations of the present work. It can be stated as follows:

Conjecture 1 (Kepler 1611). *The maximal density of sphere packings in three dimensional space is* $\pi/\sqrt{18}$.

This conjecture has been proved by Thomas Hales[1].

Theorem 1 (Hales [Hal94, Hal05]). *Kepler's conjecture is true.*

One of the chapters of [Hal05] is coauthored by Ferguson. The publication of the proof, one of the "most complicated [...] that has been ever produced", to quote his author[2], took several years and its verification required "unprecedented" efforts by a team of referees; the difficulty being made worse by the use of mechanical computations interwound with mathematical deductions. The degree of complexity of such a checking has motivated the effort to fully formalize them.

Like the four-color theorem's proof, Hales' proofs thus combines "conventional" mathematical deduction and non-trivial computations. The formalization of this development is an ambitious goal addressed by the Flyspeck project, launched by Hales himself [Hal06]. Note that other problems can be solved by proof assistants but do not rely on mechanical computation. As an example, one can mention the formal proof of the Feit-Thompson Odd Order Theorem [GAA+13]. Flyspeck also involves the formalization of many mathematical concepts and proofs; but we here do not deal with the "conventional" mathematical part of the project.

1.2 Nonlinear Inequalities

Computations are mandatory for at least three kind of tasks in Hales' proof: generation of planar graphs, use of linear programming, and bounding of non-linear expressions. Details about the two former issues are available in Solovyev's doctoral dissertation [Sol12].

We here focus on the last issue, namely the formal checking of the correctness of hundreds of nonlinear inequalities. Each of these cases boils down to the computation of a certified lower bound for a real-valued multivariate function $f : \mathbb{R}^n \to \mathbb{R}$

[1] `https://code.google.com/p/flyspeck/wiki/AnnouncingCompletion`
[2] `https://code.google.com/p/flyspeck/wiki/FlyspeckFactSheet`

over a compact semialgebraic set $K \subset \mathbb{R}^n$:

$$f^* := \inf_{\mathbf{x} \in K} f(\mathbf{x}) \ , \tag{1.1}$$

In some cases, f will be a multivariate polynomial (polynomial optimization problems (POP)); alternatively, f may belong to the algebra $\mathcal{A}$ of semialgebraic functions, which extends multivariate polynomials, obtained through arbitrary compositions of $(\cdot)^p$, $(\cdot)^{\frac{1}{p}} (p \in \mathbb{N}_{>0})$, $|\cdot|$, $+$, $-$, $\times$, $/$, $\sup(\cdot,\cdot)$, $\inf(\cdot,\cdot)$ (semialgebraic optimization problems); finally, in the most general case, f may, in addition, involve transcendental functions (sin, arctan, *etc*).

Our aim is twofold:

- The first is *automation*: we want to design a method that finds sufficiently precise lower bounds for all, or at least a majority of the functions f and domains K occurring in the proof.
- The second, as already stressed above, is *certification*; meaning that the correctness of each of these bounds must be, eventually, formally provable in a proof system such as Coq.

An additional crucial point is *precision*. Especially the inequalities of Hales' proof are essentially tight.

However, the application range for formal bounds reaches over many areas, way beyond the proof of Kepler Conjecture. Hence, we are also keen on tackling *scalability* issues, which arise when one wants to provide coarser lower bounds for optimization problems with a larger number of variables or polynomial inequalities of a higher degree.

1.3 Context

There have been a number of related efforts to obtain formal proofs for global optimization.

Lower bounds for POP can be obtained by solving sums of squares (SOS) programs using the numerical output of specialized semidefinite programming (SDP) software [HLL09]. Such techniques rely on hybrid symbolic-numeric certification methods, see Peyrl and Parrilo [PP08] and Kaltofen et al. [KLYZ12], which in turn allow to produce nonnegativity certificates which can be checked in proof assistants. Related formal frameworks include a decision procedure in Coq, described in [Bes07] as well as in Hol-light [Har07]. Both procedures include a proof-search step to find nonnegativity certificates, which relies on the same OCaml libraries. The procedure in [Bes07] is implemented as a tactic called `micromega`.

Alternative approaches to SOS are based on formalizing multivariate Bernstein polynomials. This research has been carried out in the thesis of R. Zumkeller [Zum08] and by Munõz and Narkawicz [MN13] in PVS [ORS92].

Such polynomial optimization methods can be extended to transcendental functions using multivariate polynomial approximation through a semialgebraic relaxation. This requires to be able to also certify the approximation error in order to conclude. MetiTarski [AP10] is a theorem prover that can handle nonlinear inequalities involving special functions such as ln, cos, etc. These univariate transcendental functions (as well as the square root) are approximated by a hierarchy

of approximations which are rational functions derived from Taylor or continued fractions expansions (for more details, see Cuyt et al. [CBBH08]).

The Flyspeck project also employed specific methods to verify nonlinear inequalities. Hales and Solovyev developed a nonlinear verification framework [Sol12, SH13], mixing OCaml and Hol-light [Har96] procedures to achieve formal Taylor interval approximations. A part of this procedure is informal and aims to provide useful hints such as an appropriate subdivision of the nonlinear inequality box K. The formal part of the procedure uses formalization results related to the multivariate Taylor Theorem (e.g. multivariate Taylor formula with second-order error terms) and formal interval arithmetic. Numerical computations with finite precision floating-point numbers are done in a formal setting within Hol-light, thanks to a careful representation of natural numerals over arbitrary bases (see [SH13] for more details). This formal framework is about 2000∼4000 times slower than an informal procedure (written in C++) performing the same verification.

In [CHJL11], the authors present a scheme amenable to formalization, which provides certified polynomial approximations of univariate transcendental functions. An upper bound of the approximation error is obtained by using a second approximation polynomial with bounded approximation, error relying on a non-negativity test performed by means of univariate sums of squares. The Flocq library [BM11] formalizes floating-point arithmetic inside Coq. The tactic `interval` [Mel12], built on top of Flocq, can simplify inequalities on expressions of real numbers. Our formal framework relies on this tactic for handling univariate transcendental functions. However, inequalities involving multivariate transcendental functions remain typically difficult to solve with interval arithmetic, in particular due to the correlation between arguments of unary functions (e.g. $\sin, \arctan$) or binary operations (e.g. $+, -, \times, /$).

Currently, it takes about 5000 CPU hours to verify all the nonlinear inequalities with formal Taylor interval approximations (developed by the Flyspeck project) in Hol-light. One motivation of the present work is to reduce this total verification time using alternative formal methods. In a previous work, the authors developed an informal framework, built on top of the certified –template based – global optimization method [MAGW, AGMW13a, AGMW13b]. The nonlinear template method is a certification framework, aiming at handling the approximation of transcendental functions and increasing the size of certifiable instances. It combines the ideas of maxplus approximations [FM00, AGL08] and linear templates [SSM05] to reduce the complexity of the semialgebraic approximations. Given a multivariate transcendental function f and a semialgebraic compact set K, one builds lower semialgebraic approximations of f using maxplus approximations (usually a supremum of quadratic forms) choosing a set of control points. In this way, the nonlinear template algorithm builds a hierarchy of semialgebraic relaxations that are solved with SDP.

1.4 Contributions

In this article, we present a formal framework, built on top of this informal method.

The correctness of the bounds for semialgebraic optimization problems can be verified using the interface of this algorithm with the Coq proof assistant. Thus, the certificate search and the proof checking inside Coq are separated, which is

common in the so-called *sceptical* approach [BB02]. There are some more practical difficulties however. When solving semialgebraic optimization problems (e.g. POP), the sums of squares certificates produced by existing tools do not exactly match with the system of polynomial inequalities defining K, because these external tools use limited precision floating point numbers and are thus prone to rounding errors. A certified upper bound of this error is obtained inside the proof assistant. Once the bounding of the error is obtained, the verification of the certificate is performed through an equality check in the ring of polynomials whose coefficients are arbitrary-size rationals. This means that we benefit from efficient arithmetic of these coefficients: the recent implementation of functional modular arithmetic allows to handle arbitrary-size natural numbers [GT06]. Spiwack [Spi06] has modified the virtual machine to handle 31-bits integers natively, so that arithmetic operations are delegated to the CPU. These recent developments made possible to deal with cpu-intensive tasks such as handling the proof checking of SAT traces [AGST10]. Here, it allows to check efficiently the correctness of SOS certificates. Furthermore, this verification for SDP relaxations is combined to deduce the correctness of semialgebraic optimization procedures, which requires in particular to assert that the semialgebraic functions are well-defined. It allows to handle more complex certificates for non-polynomial problems. Finally, the datatype structure of these certificates allows to reconstruct the steps of the nonlinear template optimization algorithm.

The present paper is a followup of [AGMW13a] in which the idea of maxplus approximation of transcendental functions was improved through the use of template abstractions. Here, we develop and detail the formal side of this approach. The present framework provides an automated decision procedure to obtain formal bounds for polynomial and semialgebraic functions over semialgebraic sets. This formalization is associated with the development of Coq libraries within the software package `NLCertify` (see [Mag14] as well as the software web-page[3]) and complements the existing libraries of the software, originally written in OCaml.

The paper is organized as follows. Section 2 is devoted to formal polynomial optimization. We recall some properties of SDP relaxations for polynomial problems (Section 2.1). In Section 2.2, we outline the conversion of the numerical SOS produced by the SDP solvers into an exact rational certificate. Section 2.3 describes the formal verification of this certificate inside Coq. Section 3 explains how to reduce semialgebraic problems to POP through the Lasserre-Putinar lifting. The structure of the interval enclosure certificates for semialgebraic functions is described in Section 3.1. We remind the principle of the nonlinear maxplus template method in Section 4.2. The interface between this algorithm and the formal framework is presented in Section 4.3. Finally, we demonstrate the scalability of our formal method by certifying bounds of non-linear problems from the global optimization literature as well as non trivial inequalities issued from the Flyspeck project.

[3] `http://nl-certify.forge.ocamlcore.org/`

2. FORMAL POLYNOMIAL OPTIMIZATION

We consider the general constrained polynomial optimization problem (POP):

$$f^*_{\mathrm{pop}} := \inf_{\mathbf{x}\in K_{\mathrm{pop}}} f_{\mathrm{pop}}(\mathbf{x}), \tag{2.1}$$

where $f_{\mathrm{pop}} : \mathbb{R}^n \to \mathbb{R}$ is a d-degree multivariate polynomial, K_{pop} is a semialgebraic compact set defined by inequality constraints $g_1(\mathbf{x}) \geqslant 0, \ldots, g_m(\mathbf{x}) \geqslant 0$, where $g_j(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is each time a real-valued polynomial. Recall that a d-degree multivariate polynomial p can be decomposed as $p(\mathbf{x}) = \sum_{|\boldsymbol{\alpha}|\leqslant d} p_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$, where each $\boldsymbol{\alpha}$ is a nonnegative integer vector $(\alpha_1, \ldots, \alpha_n)$, with $|\boldsymbol{\alpha}| := \sum_{i=1}^n \alpha_i$.

Since the domain K_{pop} is compact, we know that it is included in some box, say $[\mathbf{a}, \mathbf{b}] := [a_1, b_1] \times \cdots \times [a_n, b_n] \subset \mathbb{R}^n$. We can thus assume, without loss of generality, that the first constraints are precisely box constraints; more precisely, that $m \geqslant 2n$ and $g_1 := x_1 - a_1, g_2 := b_1 - x_1, \ldots,\ g_{2n-1} := x_n - a_n,\ g_{2n} := b_n - x_n$. In practice, such bounds $[\mathbf{a}, \mathbf{b}]$ are known in advance for all Flyspeck inequalities as well as for other global optimization problems which have been considered here.

Recall that the *set of feasible points* of an optimization problem is simply the domain over which the optimum is taken, i.e., here, K_{pop}.

2.1 Certified Polynomial Optimization using SDP Relaxations

Here, we remind how to cast a POP into an SOS program, which can be in turn written as an SDP. We define the set of polynomials which can be written as a sum of squares $\Sigma[\mathbf{x}] := \left\{ \sum_i q_i^2, \text{ with } q_i \in \mathbb{R}[\mathbf{x}] \right\}$. We set $g_0 := 1$ and take $k \geqslant k_0 := \max(\lceil d/2 \rceil, \lceil \deg g_1/2 \rceil, \ldots, \lceil \deg g_m/2 \rceil)$. Then, we consider the following hierarchy of SDP relaxations for Problem (2.1), consisting of the optimization problems Q_k over the variables $(\mu, \sigma_0, \ldots, \sigma_m)$:

$$Q_k : \begin{cases} \sup_{\mu,\sigma_0,\ldots,\sigma_m} & \mu \\ \text{s.t.} & f_{\mathrm{pop}}(\mathbf{x}) - \mu = \sum_{j=0}^m \sigma_j(\mathbf{x}) g_j(\mathbf{x}),\ \forall \mathbf{x} \in \mathbb{R}^n, \\ & \mu \in \mathbb{R},\ \sigma_j \in \Sigma[\mathbf{x}], \deg(\sigma_j g_j) \leqslant 2k,\ j = 0, \ldots, m\ . \end{cases}$$

The integer k is called the *SDP relaxation order* and $\sup(Q_k)$ is the optimal value of Q_k. A feasible point $(\mu_k, \sigma_0, \ldots, \sigma_m)$ of Problem Q_k is said to be an *SOS certificate*, showing the implication $g_1(\mathbf{x}) \geqslant 0, \ldots, g_m(\mathbf{x}) \geqslant 0 \implies f_{\mathrm{pop}}(\mathbf{x}) \geqslant \mu_k$. We also use the term *Putinar-type certificate* since its existence comes from the representation theorem of positive polynomials by Putinar [Put93].

The sequence of optimal values $(\sup(Q_k))_{k\geqslant k_0}$ is monotonically increasing. Lasserre showed [Las01] that it does converge to f^*_{pop} under an additional assumption on the polynomials $g_1, \ldots, g_m$ (see [Sch05] for more details). One way to ensure that this assumption is automatically satisfied is to normalize and index the box inequalities as follows (corresponding to the affine transformation $x_i \mapsto (x_i - a_i)/b_i, i = 1, \ldots, n$):

$$g_1(\mathbf{x}) := x_1,\ g_2(\mathbf{x}) := 1 - x_1, \ldots, g_{2n-1}(\mathbf{x}) := x_n,\ g_{2n}(\mathbf{x}) := 1 - x_n\ , \tag{2.2}$$

then to add the redundant constraint $n - \sum_{j=1}^n x_j^2 \geqslant 0$ to the set of constraints. For the sake of simplicity, we assume that the inequality constraints of Problem(2.1) satisfy both conditions.

Also note that our current implementation allows to compute lower bounds for POP more efficiently by using a sparse refinement of the hierarchy of SDP relaxations (Q_k) (see [WKKM06] for more details).

2.2 Hybrid Symbolic-Numeric Certification

The general scheme is thus quite clear: an external tool, here acting as an oracle, computes the certificate $(\mu_k, \sigma_0, \ldots, \sigma_m)$ and the formal proof essentially boils down to checking the equality

$$f_{\text{pop}}(\mathbf{x}) - \mu_k = \sum_{j=0}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x})$$

and Coq's ring tactic can typically verify such equalities.

There are practical difficulties however. In practice, we solve the relaxations Q_k using SDP solvers (e.g. SDPA [YFN+10]). Unfortunately, such solvers are implemented using floating-point arithmetic and the solution $(\mu_k, \sigma_0, \ldots, \sigma_m)$ satisfies only approximately the equality constraint in Q_k:

$$f_{\text{pop}}(\mathbf{x}) - \mu_k \simeq \sum_{j=0}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x}) \ .$$

More precisely, the optimization problems are formalized in Coq by using rational numbers for the coefficients. In any case, we need to deal with this approximation error.

An elaborate method would be to obtain exact certificates, for instance by the rationalization scheme (rounding and projection algorithm) developed by Peyrl and Parrilo [PP08], with an improvement of Kaltofen et al. [KLYZ12]. Let us note $\theta_k := \|f_{\text{pop}}(\mathbf{x}) - \mu_k - \sum_{j=0}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x})\|$ the error for the problem Q_k. The method of Kaltofen et al. [KLYZ12] consists in applying first Gauss-Newton iterations to refine the approximate SOS certificate, until θ_k is less than a given tolerance and then, to apply the algorithm of [PP08]. The number μ_k is approximated by a nearby rational number ${\mu_k}^{\mathbb{Q}} \lessapprox \mu_k$ and the approximate SOS certificate $(\sigma_0, \ldots, \sigma_m)$ is converted to a rational SOS (for more details, see [PP08]). Then the refined SOS is projected orthogonally to to the set of rational SOS certificates $({\mu_k}^{\mathbb{Q}}, {\sigma_0}^{\mathbb{Q}}, \ldots, {\sigma_m}^{\mathbb{Q}})$, which satisfy (exactly) the equality constraint in Q_k. This can be done by solving a least squares problem, see [PP08] for more information. Note that when the SOS formulation of the polynomial optimization problem is not strictly feasible, then the rounding and projection algorithm may fail. However, Monniaux and Corbineau proposed a partial workaround for this issue [MC11]. In this way, except in degenerate situations, we arrive at a candidate SOS certificate with rational coefficients, $({\mu_k}^{\mathbb{Q}}, {\sigma_0}^{\mathbb{Q}}, \ldots, {\sigma_m}^{\mathbb{Q}})$ from the floating point solution of (Q_k).

In our case, we do not use the rounding and projection algorithm of Peyrl and Parrilo; instead we rely on a simpler and cruder scheme. We perform a certain number of operations before handing over the certificate to Coq.

In practice, the SDP solvers solve an optimization problem (equivalent to Q_k) over symmetric matrix variables $Z_0, \ldots, Z_m$. From any floating point solution of this equivalent problem, one can extract the vectors $\mathbf{v}_{ij}$ of Z_j with the associated r_j coefficients $(\lambda_{ij})_{1 \leqslant i \leqslant r_j}$. Let v_{ij} be the polynomial with vector coefficient $\mathbf{v}_{ij}$.

Then, one has the following decomposition:

$$\sigma_j(\mathbf{x}) = \sum_{i=1}^{r_j} \lambda_{ij} v_{ij}^2(\mathbf{x}), j = 0, \ldots, m \ . \tag{2.3}$$

The extraction is done with the LACAML (Linear Algebra with OCAML) library, implementing the BLAS/LAPACK-interface. The floating-point numbers of the generated certificate are viewed as rationals through the straightforward mapping. Numerical SOS certificates are converted into rational SOS using the function `Q.of_float` of the ZARITH OCAML library, which implements arithmetic and logical operations over arbitrary-precision integers. The floating-point value μ_k is also converted into a rational.

Then, we compute the (exact) error polynomial:

$$\epsilon_{\text{pop}}(\mathbf{x}) := f_{\text{pop}}(\mathbf{x}) - \mu_k - \sum_{j=0}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x}) \ .$$

Now we explain how to provide another, hopefully small, bound for ϵ_{pop}. Note that this polynomial can be decomposed as $\epsilon_{\text{pop}}(\mathbf{x}) = \sum_{|\boldsymbol{\alpha}| \leqslant 2k} \epsilon_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$. Fortunately, the coefficients of this polynomial are generally small, which allows us to choose

$$\epsilon^*_{\text{pop}} := \sum_{\epsilon_{\boldsymbol{\alpha}} \leqslant 0} \epsilon_{\boldsymbol{\alpha}}.$$

Indeed, the box inequalities guaranty that for each $\mathbf{x} \in [0,1]^n$:

$$\epsilon_{\text{pop}}(\mathbf{x}) = \sum_{|\boldsymbol{\alpha}| \leqslant 2k} \epsilon_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}} \ \leqslant \ \sum_{\epsilon_{\boldsymbol{\alpha}} \leqslant 0} \epsilon_{\boldsymbol{\alpha}} = \epsilon^*_{\text{pop}} \ . \tag{2.4}$$

Finally, we compute the actual exact bound given by the certificate: $\mu_k^- := \mu_k + \epsilon^*_{\text{pop}}$. We see that $\mu_k^- := \mu_k + \epsilon^*_{\text{pop}}$ is a valid lower bound of f_{pop} over the domain K_{pop}.

Note that we could optimize the polynomial ϵ_{pop} over $[0,1]^n$, but it would be as hard as solving the initial POP. Moreover, one would have to consider again some residual polynomial after solving the corresponding SDP relaxation.

2.3 A Formal Checker for Polynomial Systems

Following the procedure described in Section 2.2, we extract a rational certificate $(\mu_k^-, \sigma_0, \ldots, \sigma_m, \epsilon_{\text{pop}})$.

By definition, this certificate satisfies the following, for all $\mathbf{x} \in [0,1]^n$:

$$f_{\text{pop}}(\mathbf{x}) - \mu_k^- = \sum_{j=0}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x}) + (\epsilon_{\text{pop}}(\mathbf{x}) - \epsilon^*_{\text{pop}}) \ . \tag{2.5}$$

The procedure which checks the equality (2.5) between polynomials and SOS inside COQ relies on computational reflexion. We use the reflexive `ring` tactic, by using a so-called "customized" polynomial ring [4]. Given two polynomials p and q, this tactic verifies the polynomial equality "$p = q$" in two steps. The first

[4] `http://coq.inria.fr/refman/Reference-Manual029.html`

step is a normalization of both p and q w.r.t. associativity, commutativity and distributivity, constant propagation and rewriting of monomials. The second step consists in comparing syntactically the results of this normalization.

Given a sequence of polynomial constraints `g` $:= [g_1, \ldots, g_m]$, a lower bound μ_k^-, an objective polynomial f_{pop} and a POP certificate `cert_pop` (build with a Putinar-type certificate and a polynomial remainder ϵ_{pop}), the fact that a successful check of the certificate entails nonnegativity of the polynomial is formalized by the following correctness lemma:

```
Lemma correct_pop env g f_pop cert_pop μ_k^- :
g_nonneg env g → checker_pop g f_pop μ_k^- cert_pop = true →
μ_k^- ⩽ [|f_pop|]_env .
```

The way this lemma is stated shows that in our development, we use an environment function `env` to bind positive integers to polynomial real variables. The function $[|\cdot|]_{\texttt{env}}$ maps a polynomial expression to the carrier type `R`. The function `g_nonneg` explicits the conditions by returning the conjunction of propositions $[|g_1|]_{\texttt{env}} \geqslant 0 \wedge \cdots \wedge [|g_m|]_{\texttt{env}} \geqslant 0$.

In the sequel of this section, we describe the data structure and the auxiliary lemmas that allow to define and prove `correct_pop`.

2.3.1 *Encoding Polynomials.* Checking ring equalities between polynomials requires to provide a type of coefficients. In our current setting, we choose `bigQ`, the type of arbitrary-size rationals. The ring morphism `IQR` injects rational coefficients into the carrier type `R` of Coq classical real numbers. For the sequel, we also note:

```
Notation "[ c ]" := IQR c.
```

We use two types of polynomials: `PExpr` is for uninterpreted ring expressions while `PolC` is for uninterpreted normalized polynomial expressions:

```
Inductive PExpr : Type :=
 | PEc   : bigQ → PExpr
 | PEX   : positive → PExpr
 | PEadd : PExpr → PExpr → PExpr
 | PEsub : PExpr → PExpr → PExpr
 | PEmul : PExpr → PExpr → PExpr
 | PEopp : PExpr → PExpr
 | PEpow : PExpr → N → PExpr.

Inductive PolC : Type :=
 | Pc   : bigQ → PolC
 | Pinj : positive → PolC → PolC
 | PX   : PolC → positive → PolC → PolC.
```

The three constructors `Pc`, `Pinj` and `PX` satisfy the following conditions:

(1) The polynomial `(Pc c)` is the constant polynomial that evaluates to `[c]`.
(2) The polynomial `(Pinj i p)` is obtained by shifting the index of `i` in the variables of `p`. In other words, when `p` is interpreted as the value of the $(n - i)$ variables polynomial $p(x_1, \ldots, x_{n-i})$, then one interprets `(Pinj i p)` as the value of $p(x_i, \ldots, x_n)$.

(3) Let `p` (resp. `q`) represents p (resp. $q(x_1, \ldots, x_{n-1})$). Then `(PX p j q)` evaluates to $px_1^j + q(x_2, \ldots, x_n)$.

Polynomial expressions can be normalized via the procedure `norm : PExpr → PolC`. We note $p_1 \equiv p_2$ the boolean equality test between two normal forms p_1 and p_2.

The procedure `checker_pop` for SOS certificates relies on the correctness lemma `norm_eval`:

```
Lemma norm_eval (p : PExpr) (q : PolC) :
norm(p) ≡ q ⟹ ∀ env, [|p|]env = eval_pol env q.
```

Here, the function `eval_pol` maps a sparse normal form to the carrier type `R`.

2.3.2 *Encoding SOS certificates.* We recall that an SOS can be decomposed as $\sigma_j := \sum_{i=1}^{r_j} \lambda_{ij} v_{ij}^2$. Each σ_j is encoded using a finite sequence of tuples composed of an arbitrary-size rational (of type `bigQ`) and a polynomial (of type `PolC`). Then, we build Putinar-type certificates $\sum_{j=0}^{m} \sigma_j(\mathbf{x}) g_j(\mathbf{x})$ of type `cert_putinar`, with a finite sequence of tuples, composed of an SOS σ_j and a polynomial g_j. Finally, we define POP certificates (object of type `cert_pop`) for Problem 2.1 using a Putinar-type certificate and a polynomial remainder ϵ_{pop}.

2.3.3 *Formal proofs for polynomial bounds.* The coarse lower bound ϵ_{pop}^* of a polynomial remainder ϵ_{pop} can be computed inside Coq with the following recursive procedure:

```
Fixpoint lower_bound_0_1 ϵpop := match ϵpop with
  | Pc c      ⟹ min c 0
  | Pinj _ p  ⟹ lower_bound_0_1 p
  | PX p _ q  ⟹ lower_bound_0_1 p + lower_bound_0_1 q
end.
```

The remainder inequality (2.4) can then be proved by structural induction.

Any certificate `c` of type `cert_putinar` can be mapped to a sparse Horner form with the function `toPolC` function, using the sequences `g`, λ and the environment `env`. Since each g_j is nonnegative by assumption and each σ_j is an SOS, one can verify easily the nonnegativity of a Putinar-type certificate by checking the nonnegativity of each λ element. The boolean function `checker_pop` verifies that:

(1) each element of λ is nonnegative

(2) `norm(`$f_{\text{pop}} - \mu_k^-$`)` ≡ `(toPolC g λ sos)` + ϵ_{pop} − `[lower_bound_0_1` ϵ_{pop}`]` (*i.e.* equality (2.5) is satisfied)

Then, we can prove Lemma `correct_pop`:

Proof. By assumption, one can apply `norm_eval` with $p = f_{\text{pop}} - \mu_k^-$ and q = `(toPolC g λ sos)` + ϵ_{pop} − `[lower_bound_0_1` ϵ_{pop}`]`. We first use the hypothesis `(g_nonneg env g)` as well as the nonnegativity of the rationals of the sequence λ to deduce that `eval_pol env (toPolC g λ sos)` $\geqslant$ `0`. The nonnegativity of `eval_pol env (`ϵ_{pop} − `[lower_bound_0_1` ϵ_{pop}`])` comes from inequality (2.4). □

We also define the type `cert_pop_itv` of formal interval bounds certificates for polynomials expressions:

```
Definition cert_pop_itv := (cert_pop * cert_pop).
```

Lemma `correct_pop_itv` relates interval enclosure of polynomials with certificates of type `cert_pop_itv`:

```
Lemma correct_pop_itv env g f_pop (i : itv) c:
g_nonneg env g → checker_pop_itv g f_pop i c = true →
    [|f_pop|]_env ∈ i.
```

Here, `itv` refers to the type of intervals, encoded using two rational coefficients:

```
Inductive itv : Type := Itv : bigQ → bigQ → itv.
Definition itv01 := Itv 0 1. (* the interval [0,1] *)
```

Moreover, we denote the lower (resp. upper) bound of an interval i by $\underline{i}$ (resp. $\overline{i}$). We note $[|p|]_{\text{env}} \in i$ to state that $\underline{i} \leqslant [|p|]_{\text{env}} \leqslant \overline{i}$.

2.4 Experimental Results

We first recall some Flyspeck related definitions [Hal03]:

$$\begin{aligned}\Delta\mathbf{x} &:= x_1x_4(x_2-x_1+x_3-x_4+x_5+x_6)+x_2x_5(x_1-x_2+x_3+x_4-x_5+x_6)\\ &\quad +x_3x_6(x_1+x_2-x_3+x_4+x_5-x_6)-x_3(x_2x_4+x_1x_5)-x_6(x_1x_2+x_4x_5)\ ,\\ \partial_4\Delta\mathbf{x} &:= x_1(x_2-x_1+x_3-2x_4+x_5+x_6)+x_2x_5+x_3x_6-x_3x_2-x_6x_5\ .\end{aligned}$$

We tested our formal verification procedure on the following polynomial problems, occurring as sub-problems of Flyspeck nonlinear inequalities:

$$\begin{aligned}&\textit{POP1}: (4\leqslant x_1,x_2,x_3,x_5,x_6\leqslant 2.52^2\wedge 2.52^2\leqslant x_4\leqslant 8) \implies \partial_4\Delta\mathbf{x}\in[-40.33,40.33]\,.\\ &\textit{POP2}: (4\leqslant x_1,x_2,x_3,x_5,x_6\leqslant 2.52^2\wedge 2.52^2\leqslant x_4\leqslant 8) \implies 4x_1\Delta\mathbf{x}\in[2047,14262]\,.\end{aligned}$$

A preliminary phase consists in scaling the POP to apply the correctness Lemma `correct_itv`. Table I shows some comparison results with the `micromega` tactic, available inside Coq. While performing the proof-search step, the tactic relies on the external SDP solver Csdp. This solver is used to solve another SDP relaxation that is more general than Q_k (Stengle Positivstellensatz [Ste74]) to find witnesses of unfeasibility of a set of polynomial constraints (see [Bes07] for more details). To deal with the numerical errors of Csdp, the proof-search (OCaml libraries) also includes a projection algorithm, which is is performed in such a way that $\epsilon_{\text{pop}} = 0$. Thus, the procedure returns a rational SOS certificate that matches exactly $f_{\text{pop}} - \tilde{\mu}_k$, so that the proof-checking consists only in verifying a polynomial equality.

Numerical experiments are performed using the Coq proof scripts of our formalization (available in the `NLCertify` [5] software package), on an Intel Core i5 CPU (2.40 GHz). For the timings related to `NLCertify`, the column "t_i" refers to the time spent to find the SOS certificates "externally" (while solving SDP relaxations and extracting SOS certificates in OCaml) and the column "t_f" refers to the total verification time (while compiling Coq proof scripts)[6]. Notice that for *POP2*, we

[5] `http://nl-certify.forge.ocamlcore.org/`

[6] Note that obtaining t_i while using `micromega` is possible in practice but would require to modify the OCaml libraries of the tactic.

consider the projection of $\Delta\mathbf{x}$ with respect to the first n coordinates on the box K (fixing the other variables to 6.3504).

Table I indicates that our tool outperforms the `micromega` decision procedure, thanks to the sparse variant of relaxation Q_k and a simpler projection method. The symbol "–" means that the inequality could not be checked by `micromega` within one hour of computation. Problems occur while performing the proof-search step of `micromega`, as either the projection algorithm fails or the computational cost of the SDP relaxation is too demanding.

Table I. Comparing our formal POP checker with `micromega`,

Problem	n	`NLCertify` t_i	`NLCertify` t_f	`micromega` $t_i + t_f$
POP1	6	0.04 *s*	0.16 *s*	18 *s*
POP2	2	0.06 *s*	0.18 *s*	0.72 *s*
	3	0.14 *s*	0.78 *s*	–
	6	4.8 *s*	26.4 *s*	–

3. FORMAL SEMIALGEBRAIC OPTIMIZATION

We can now build on the work of the previous section in order to extend the framework to obtain also formal bounds for semialgebraic optimization problems:

$$f^*_{\mathrm{sa}} := \inf_{\mathbf{x}\in K} f_{\mathrm{sa}}(\mathbf{x}) \ , \tag{3.1}$$

where $f_{\mathrm{sa}} \in \mathcal{A}$ and $K := \{\mathbf{x} \in \mathbb{R}^n : g_1(\mathbf{x}) \geqslant 0, \ldots, g_m(\mathbf{x}) \geqslant 0\}$ is a basic semialgebraic set such that the constraints (g_j) satisfy (2.2). For the sake of clarity, we explicit only the subset of $\mathcal{A}$ consisting of arbitrary composition of polynomials with $\sqrt{\cdot}, +, -, \times, /$, whenever these operations are well-defined (neither division by zero nor square root of negative value occur). However, we can deal with the case $f_{\mathrm{sa}} = \max(f_1, f_2)$ by using the identity: $2\max(f_1, f_2) = f_1 + f_2 + \sqrt{(f_1 - f_2)^2}$. Similar identities exist to handle operations such as $\min(\cdot,\cdot), |\cdot|$.

The function f_{sa} has a basic semialgebraic lifting; this means that one adds new "lifting variables" in order to get rid of the non-polynomial functions in f_{sa} thus reducing the problem to a POP (for more details, see e.g. [LP10]). More precisely, we can add auxiliary variables $x_{n+1}, \ldots, x_{n+p}$ (lifting variables), and construct polynomials $h_1, \ldots, h_s \in \mathbb{R}[x_1, \ldots, x_{n+p}]$ defining the semialgebraic set:

$$K_{\mathrm{pop}} := \{(x_1, \ldots, x_{n+p}) \in \mathbb{R}^{n+p} : \mathbf{x} \in K, h_l(x_1, \ldots, x_{n+p}) \geqslant 0, \ l = 1, \ldots, s\} \ ,$$

such that $f^*_{\mathrm{pop}} := \inf\{x_{n+p} : (x_1, \ldots, x_{n+p}) \in K_{\mathrm{pop}}\}$ is a lower bound of f^*_{sa}.

Now, we explain how to implement this procedure in a formal setting.

3.1 Data Structure for Semialgebraic Certificates

The inductive type `cert_sa` represents interval bounds certificates for semialgebraic optimization problems:

```
Inductive cert_sa : Type :=
| Poly  : PExpr → itv → cert_pop_itv → cert_sa
| Fadd  : cert_sa → cert_sa → itv → cert_pop_itv → cert_sa
```

```
| Fmul  : cert_sa → cert_sa → itv → cert_pop_itv → cert_sa
| Fdiv  : cert_sa → cert_sa → itv → cert_pop_itv → cert_sa
| Fopp  : cert_sa → itv → cert_sa
| Fsqrt : cert_sa → itv → cert_sa.
```

Note that the constructor `Poly` takes a type `PExpr` object as argument to represent the polynomial components of the function f_{sa}. The other constructors correspond to the various ways of building elements of $\mathcal{A}$. Even though this certificate data-structure may look heavy-weighted, all constructors are required for verification purpose: for instance `Fadd` is mandatory to check the nonnegativity of a function such as $\sqrt{p}+q$, for polynomials p and q. Each constructor takes a formal interval bound i and a certificate `c` of type `cert_pop_itv` (as defined in section 2) as arguments. In the sequel, we explain how to ensure that i is a valid interval enclosure of f_{sa} by checking the correctness of `c`.

Example 2 (FROM LEMMA9922699028 FLYSPECK).
From the two multivariate polynomials $p(\mathbf{x}) := \partial_4\Delta\mathbf{x}$ and $q(\mathbf{x}) := 4x_1\Delta\mathbf{x}$, we define the semialgebraic function $r(\mathbf{x}) := p(\mathbf{x})/\sqrt{q(\mathbf{x})}$ over $K := [4, 2.52^2]^3 \times [2.52^2, 8] \times [4, 2.52^2]$. Using the procedure described in Section 2, one obtains a formal interval i_p (resp. i_q) enclosing the range of p (resp. q), certified by an SOS certificate c_p (resp. c_q). We also derive an interval enclosure $i_\surd := [m_7, M_7]$ for $\sqrt{q}$ and then build the following terms:

```
Definition p := Poly p i_p c_p. Definition q := Poly q i_q c_q.
Definition sqrtq := Fsqrt q i_√.
Definition r := Fdiv p sqrtq i_r c_r.
```

The SOS certificate c_r allows to prove that i_r is a correct interval enclosure of r.

The interpretation of `cert_sa` objects is straightforward, using the evaluation procedure for polynomial expressions. Thus, we also note $[\![f]\!]_{\texttt{env}}$ the interpretation of the semialgebraic certificate f, which returns the expression of the semialgebraic function f_{sa}. A procedure $\texttt{n}_{\texttt{lifting}}$ returns the index `v` of the lifting variable which represents f_{sa}. When f_{sa} is a polynomial, $\texttt{n}_{\texttt{lifting}}$ returns the number of variables involved in f_{sa}. The value is incremented when f_{sa} is either a division or a square root.

```
Fixpoint n_lifting f v :=
  match f with
  | Poly p _ _ ⟹ v | Fopp a _ ⟹ n_lifting q v
  | Fdiv f1 f2 _ _ ⟹ n_lifting f2 (n_lifting f1 v) + 1
  | Fsqrt q _ ⟹ n_lifting q v + 1
  | Fadd f1 f2 _ _ | Fsub f1 f2 _ _ | Fmul f1 f2 _ _ ⟹
     n_lifting f2 (n_lifting f1 v)
  end.
```

Then, two procedures are mandatory to reduce Problem (3.1) into a polynomial optimization problem $\min_{\mathbf{x}\in K_{\text{pop}}} f_{\text{pop}}$. The function `obj` derives the objective polynomial f_{pop}, while `cstr` returns a list of polynomials (h_l) defining K_{pop}.

```
Fixpoint obj f v : PExpr :=
```

```
  match f with
   | Poly p _ _ ⟹ p
   | Fopp q _ ⟹ - (obj q v)
   | Fadd f1 f2 _ _ ⟹ (obj f1 v) + (obj f2 (n_lifting f1 v))
   | Fmul f1 f2 _ _ ⟹ (obj f1 v) * (obj f2 (n_lifting f1 v))
   | Fsub f1 f2 _ _ ⟹ (obj f1 v) - (obj f2 (n_lifting f1 v))
   | Fdiv f1 f2 i _ ⟹
             scale_obj (PEX (n_lifting f2 (n_lifting f1 v) + 1)) i
   | Fsqrt q i ⟹ scale_obj (PEX (n_lifting q v + 1)) i
  end.
```

Given a polynomial p and an interval $i := [m, M]$, the result of (`scale_obj` p i) is $(M - m)p + m$.

```
Fixpoint cstr f v : seq PExpr :=
  match f with
   | Poly p _ _ ⟹ [::]
   | Fopp q _ ⟹ cstr q v
   | Fadd f1 f2 _ _ | Fsub f1 f2 _ _ | Fmul f1 f2 _ _ =>
      cstr f1 v ++ cstr f2 (var f1 v)
   | Fdiv f1 f2 _ _ ⟹ cstr f1 v ++ cstr f2 (var f1 v) ++
      [::PEsub (PEmul (obj f2 (var f1 v)) (obj f v)) (obj
      f1 v); PEsub (obj f1 v) (PEmul (obj f2 (var f1 v)) (
      obj f v))]
   | Fsqrt q _ ⟹ cstr q v ++ [::PEsub (PEmul (obj f v)
      (obj f v)) (obj q v) ; PEsub (obj q v) (PEmul (obj
      f v) (obj f v))]
  end.
```

Example 3. Applying the function $\mathtt{n}_{\mathtt{lifting}}$ to the six dimensional functions q and r of Example 2 yields $\mathtt{n}_{\mathtt{lifting}}$ `sqrtq` $= 7$ and $\mathtt{n}_{\mathtt{lifting}}$ `r` $= 8$. Then, `obj sqrtq` returns the polynomial $(M_7 - m_7)\, x_7 + m_7$ and `obj r` returns $(M_8 - m_8)\, x_8 + m_8$. The interval $[m_8, M_8]$ is obtained using formal interval arithmetic division $i_p \,\hat{/}\, i_{\surd}$. This scaling procedure allows to use the function `lower_bound_0_1` since one can prove that $x_7, x_8 \in [0, 1]$. Here `cstr sqrtq` returns the finite sequence of polynomials $l_{\surd} := [h_1; h_2; h_3; h_4]$, with $h_1 := [(M_7 - m_7)\, x_7 + m_7]^2 - q$, $h_2 := -h_1$, $h_3 := x_7$ and $h_4 := 1 - x_7$. Next, `cstr r` returns the concatenation of $l_{\surd}$ with $l_r := [h_5; h_6; h_7; h_8]$, where $h_5 := [(M_8 - m_8)\, x_8 + m_8][(M_7 - m_7)\, x_7 + m_7] - p$, $h_6 := -h_5$, $h_7 := x_8$ and $h_8 := 1 - x_8$.

3.2 Formal Interval Bounds for Semialgebraic Functions

Now, we introduce the function `checker_sa` built on top of `checker_pop_itv`, which checks recursively the correctness of certificates for semialgebraic functions. For the sake of simplicity and to stay consistent with Example 3, we only present the result of the procedure for the constructors `Poly`, `Fdiv` and `Fsqrt`. We use the inclusion relation: $[a, b] \subseteq [a', b']$ whenever $a' \leqslant a$ and $b \leqslant b'$, the formal interval arithmetic square: $\hat{\mathrm{sq}}\,[a, b] := [(\max\{0, a, -b\})^2, (\max\{-a, b\})^2]$, as well as the interval positivity: $[a, b] \,\hat{>}\, 0$ whenever $a > 0$.

```
Fixpoint checker_sa g f : bool :=
  match f with
  | Poly p i c ⟹ checker_pop_itv g p i c
  | Fdiv f1 f2 i c ⟹
      checker_sa g f1 && checker_sa g f2 && 0 ∉ (itv f2)
   && checker_pop_itv (g ++ cstr f) (obj f) i c
  | Fsqrt q i ⟹ checker_sa g q &&
    itv q ≽ 0 && i < ī && (itv q) ⊆ sq i
  ...
  end.
```

Recall that our goal is to prove the correctness of a lower bound μ^- of $\min_{\mathbf{x}\in K} f_{\rm sa}(\mathbf{x})$ (resp. an upper bound μ^+ of $\max_{\mathbf{x}\in K} f_{\rm sa}(\mathbf{x})$). Then, one can apply the following correctness lemma to state that the interval $i := [\mu^-, \mu^+]$ (returned by $itv\ f$) is a valid enclosure of $f_{\rm sa}$ over K whenever one succeeds to check the certificate f.

```
Lemma correct_fsa env g f v : g_nonneg env g →
checker_sa g f = true → [|f|]env ∈ (itv f).
```

Proof. By induction over the structure of semialgebraic expressions. □

Example 4 (Formal bounds for the function of Example 2). Continuing Example 3, one considers the POP:

$$\min_{\mathbf{x},x_7,x_8} \{(M_8 - m_8)\ x_8 + m_8) : \mathbf{x} \in K, h_1(\mathbf{x}, x_7, x_8) \geqslant 0, \ldots, h_8(\mathbf{x}, x_7, x_8) \geqslant 0\},$$

to bound from below the function $r(\mathbf{x}) := p(\mathbf{x})/\sqrt{q(\mathbf{x})}$. Solving this POP using the second order SDP relaxation Q_2 yields the lower bound $\mu_2^- = -0.618$. Similarly, one obtains the upper bound $\mu_2^+ = 0.892$. The procedure `checker_sa` calls the function `checker_pop_itv` to prove the correctness of the interval bounds i_p, i_q (as detailed in Section 2.4) and $i_r := [\mu_2^-, \mu_2^+]$. The total running time of this formal verification in Coq is about $200s$. Adding the bit-size of all rational coefficients involved in this certificate yields a total of about 667 kbit. About 90% of the CPU time is spent verifying the correctness of SOS certificates, that is checking polynomial equalities with the `ring` tactic.

The corresponding proof script is available in the `NLCertify` package [7].

4. CERTIFIED BOUNDS FOR MULTIVARIATE TRANSCENDENTAL FUNCTIONS

We now consider an instance of Problem (1.1). We identify the objective function f with its abstract syntax tree t, whose leaves are semialgebraic functions (see Section 3) and other nodes are either basic binary operations ($+$, $\times$, $-$, $/$) or belong to the set $\mathcal{D}$ of unary transcendental functions (sin, *etc*). We first recall how to handle these unary functions using maxplus approximations.

4.1 Maxplus Approximations for Univariate Semiconvex Transcendental Functions

We consider transcendental functions which are twice differentiable. Thus, the restriction of $r \in \mathcal{D}$ to any closed interval I is γ-semiconvex for a sufficiently large

[7] the file `coq/fsacertif.v` in the archive at `https://forge.ocamlcore.org/frs/?group_id=351`

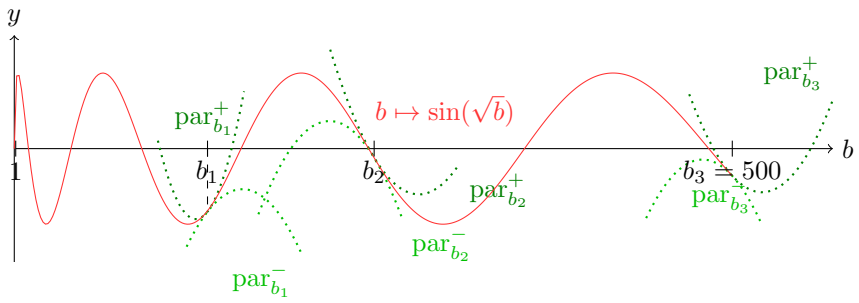

Fig. 1. Template Maxplus Semialgebraic Approximations for $b \mapsto \sin(\sqrt{b})$: $\max_{j\in\{1,2,3\}}\{\mathrm{par}^-_{b_j}(x_i)\} \leqslant \sin\sqrt{x_i} \leqslant \min_{j\in\{1,2,3\}}\{\mathrm{par}^+_{b_j}(x_i)\}$

γ, *i.e.* the univariate function $g := r + \frac{\gamma}{2}|\cdot|^2$ is convex on I (for more details on maxplus approximation, we refer the interested reader to [AGK05, McE06]). Using the convexity of g, one can always find a constant $\gamma \leqslant \sup_{b\in I} -r''(b)$ such that for all $b_i \in I$:

$$\forall b \in I, \quad r(b) \geqslant \mathrm{par}^-_{b_i}(b) := -\frac{\gamma}{2}(b-b_i)^2 + r'(b_i)(b-b_i) + r(b_i) \ . \tag{4.1}$$

Note that the choice $\gamma = \sup_{b\in I} -r''(b)$ is always valid. By selecting a finite subset of control points $B \subset I$, one can bound r from below using a maxplus under-approximation:

$$\forall b \in I, \quad r(b) \geqslant \max_{b_i\in B} \mathrm{par}^-_{b_i}(b) \ . \tag{4.2}$$

Example 5. Consider the function $f := \sum_{i=1}^n x_i \sin(\sqrt{x_i})$ defined over $[1, 500]^n$ (Modified Schwefel Problem [AKZ05]) and the finite set $\{b_1, b_2, b_3\}$ of control points, with $b_1 := 135$, $b_2 := 251$, $b_3 := 500$. For each $i = 1, \ldots, n$, consider the subtree $\sin(\sqrt{x_i})$. First, we get the equations of $\mathrm{par}^-_{b_1}$, $\mathrm{par}^-_{b_2}$ and $\mathrm{par}^-_{b_3}$, which are three under-approximations of the function $b \mapsto \sin(\sqrt{b})$ on the real interval $I := [1, \sqrt{500}]$. Similarly we obtain three over-approximations $\mathrm{par}^+_{b_1}$, $\mathrm{par}^+_{b_2}$ and $\mathrm{par}^+_{b_3}$ (see Figure 1). Then, we obtain the under-approximation $t_i^- := \max_{j\in\{1,2,3\}}\{\mathrm{par}^-_{b_j}(x_i)\}$ and the over-approximation $t_i^+ := \min_{j\in\{1,2,3\}}\{\mathrm{par}^+_{b_j}(x_i)\}$.

4.2 The Nonlinear Maxplus Template Method

Our main algorithm `template_approx` (Figure 2) is based on a previous method of the authors [MAGW], in which the objective function is bounded by means of semialgebraic functions. For the sake of completeness, we first recall the basic principles of this method.

Given a function represented by an abstract tree t, semialgebraic lower and upper approximations t^- and t^+ are computed by induction. If the tree is reduced to a leaf, *i.e.* $t \in \mathcal{A}$, we set $t^- = t^+ := t$. If the root of the tree corresponds to a binary operation `bop` with children c_1 and c_2, then the semialgebraic approximations c_1^-, c_1^+ and c_2^-, c_2^+ are composed using a function `compose_bop` to provide bounding approximations of t. Finally, if t corresponds to the composition of a transcendental

(unary) function r with a child c, we first bound c with semialgebraic functions c^+ and c^-. We compute a lower bound c_m of c^- as well as an upper bound c_M of c^+ to obtain an interval $I := [c_m, c_M]$ enclosing c. Then, we bound r from below and above by computing parabola at given control points with a function called `build_par`, thanks to the semiconvexity properties of r on the interval I (e.g. the functions $r^- := \max_{j\in\{1,2,3\}}\{\mathrm{par}^-_{b_j}\}$ and $r^+ := \min_{j\in\{1,2,3\}}\{\mathrm{par}^+_{b_j}\}$ from Example 5). These parabola are composed with c^+ and c^-, thanks to a function denoted by `compose_approx` (Line 9).

At the end (Line 11), we call the function `min_sa` (resp. `max_sa`) which determines lower (resp. upper) bounds of the approximation t^- (resp. t^+) using techniques presented in Section 3.

Input: tree t, semialgebraic set K, finite sequence of control points s
Output: lower bound m, upper bound M, lower semialgebraic approximation t^-, upper semialgebraic approximation t^+
1: **if** $t \in \mathcal{A}$ **then** $t^- := t$, $t^+ := t$
2: **else if** `bop` $:=$ `root`(t) is a binary operation with children c_1 and c_2 **then**
3: $\quad m_i, M_i, c_i^-, c_i^+ :=$ `template_approx`(c_i, K, s) for $i \in \{1, 2\}$
4: $\quad t^-, t^+ :=$ `compose_bop`$(c_1^-, c_1^+, c_2^-, c_2^+,$ `bop`$)$
5: **else if** $r :=$ `root`(t) is a univariate transcendental function with a child c **then**
6: $\quad m_c$, M_c, c^-, $c^+ :=$ `template_approx`(c, K, s)
7: $\quad I := [m_c, M_c]$
8: $\quad r^-, r^+ :=$ `build_par`(r, I, c, s)
9: $\quad t^-, t^+ :=$ `compose_approx`$(r, r^-, r^+, I, c^-, c^+)$
10: **end**
11: **return** `min_sa`(t^-, K), `max_sa`(t^+, K), t^-, t^+

Fig. 2. `template_approx`

Example 6. We illustrate the nonlinear maxplus template method with the function f of Example 5. We approximate f with maxplus approximations built with 3 control points (Figure 1), which allows to reduce the modified Schwefel problem to the following POP:

$$\begin{cases} \min\limits_{\mathbf{x},\mathbf{z}} \; -\sum_{i=1}^n x_i z_i \\ \text{s.t. } z_i \leqslant \mathrm{par}^+_{b_j}(x_i)\,, \quad j \in \{1,2,3\}\,, \quad i = 1,\ldots,n\ , \\ \qquad \mathbf{x} \in [1,500]^n\,, \quad \mathbf{z} \in [-1,1]^n\ . \end{cases}$$

4.3 Formal Verification of Semialgebraic Relaxations

The correctness of the semialgebraic maxplus approximations for univariate functions (computed by the `build_par` procedure, see Figure 2 at Line 8) is ensured with the `interval` [8] tactic [Mel12], available inside Coq. As detailed in Section 3.2, the procedure `checker_sa` validates the interval bounds for semialgebraic problems obtained with the functions `min_sa` and `max_sa` (see Figure 2 at Line 11).

Table II presents the results obtained when proving the correctness of lower bounds for semialgebraic relaxations of two 6-variables Flyspeck inequalities. When the bounds obtained with the algorithm `template_approx` are too coarse to certify

[8] `https://www.lri.fr/~melquion/soft/coq-interval/`

a given inequality, we perform a branch and bound procedure over the domain K. We refer to #boxes as the total number of domain cuts that are mandatory to prove the inequality. As for Table I, the time t_i refers to the informal verification time, required to construct the certificates for semialgebraic functions, while using the optimization algorithm without any call to the Coq libraries. The total verification time t_f is then compared with t_i.

Table II. Formal Bounds Computation Results for Semialgebraic Relaxations of Flyspeck Inequalities

Inequality	#boxes	t_i	t_f	$\frac{t_i+t_f}{t_i}$
9922699028	39	295 s	2218 s	8.5
3318775219	338	2285 s	19136 s	9.4

Here, the formal verification of SOS certificates is the bottleneck of the computational certification task. Indeed, it is 8.5 (resp. 9.4) times slower to prove the correctness of semialgebraic lower bounds for the first (resp. second) inequality. For both inequalities, it takes about 7% of the total time to compute bounds with SDP. Note that half this time is spent to compute negative bounds which are not formally verified afterwards. Such non trivial inequalities are also used as test cases for the formal techniques employed by the Flyspeck project (see the row corresponding to the inequality 3318775219 in Table 2 of [SH13]) and it takes about the same amount of CPU time to verify them with both methods. For comparison purpose, notice that this ratio between formal and informal verification does not exceed 10 in our case, while it is about 2000∼4000 in [SH13]. Also, as mentioned in [MAGW], the number of subdivisions is much smaller than for methods using interval Taylor approximation (9370 for the first inequality and 25994 for the second one[9]), due to the precision of SOS-based methods.

Table III presents the results obtained for examples issued from the global optimization literature (see Appendix B in [AKZ05] for more details). For each problem, we indicate the number of subdivisions #boxes that are performed to obtain the lower bound m with our method.

Example 7. We recall the definition of Problem(MC):
$\min_{\mathbf{x}\in[-1.5,4]\times[-3,3]} \sin(x_1+x_2)+(x_1-x_2)^2-\frac{3}{2}x_1+\frac{5}{2}x_2+1$.

The package `NLCertify` contains an example of proof obligations for Problem MC [10] on the box $[-\frac{3}{2},-\frac{1}{8}]\times[-3,-\frac{9}{4}]$, allowing to assert the following:

Lemma 8. $\forall x_1,x_2\in[0,1], -1.92 \leqslant \sin(\frac{11}{8}x_1-\frac{3}{2}+\frac{3}{4}x_2-3)+(\frac{11}{8}x_1-\frac{3}{2}+\frac{3}{4}x_2-3)^2-\frac{3}{2}(\frac{11}{8}x_1-\frac{3}{2})+\frac{5}{2}(\frac{3}{4}x_2-3)+1$.

Proof. Using the interval tactic, one proves that

[9]The data come from the benchmarks file available at `http://code.google.com/p/flyspeck/source/browse/trunk/informal_code/interval_code/qed_log_2012.txt`. The file indicates the number of subdivisions (denoted by "cells") for each inequality while running informal verification in C++ with interval arithmetic and directed rounding.
[10]the file `coq/mccertif.v` in the archive at `https://forge.ocamlcore.org/frs/?group_id=351`

Table III. Formal Bounds Computation Results for Semialgebraic Relaxations of Global Optimization Problems

Problem	n	m	#boxes	t_i	t_f	$\frac{t_i+t_f}{t_i}$
MC	2	−1.92	17	1.8 *s*	1.9 *s*	2.1
SWF	5	−2150	78	270 *s*	477 *s*	2.8

(i) $\forall z \in [-\frac{9}{2}, -\frac{19}{8}], r^-(z) \leqslant \sin z$, where the parabola r^- is defined as follows:

$$r^-(z) := -\frac{68787566775937}{140737488355328}z^2 - \frac{104740403727667521893}{23346660468288651264}z - \frac{145294742556168586619925337}{154917232470538711235297 28}.$$

(ii) By substitution, it follows that $\forall x_1, x_2 \in [0,1], r^-(\frac{11}{8}x_1 - \frac{3}{2} + \frac{3}{4}x_2 - 3) \leqslant \sin(\frac{11}{8}x_1 - \frac{3}{2} + \frac{3}{4}x_2 - 3)$.

(iii) Using a Putinar-type certificate, one checks the nonnegativity of the left-hand side polynomial with the procedure `correct_pop`, which yields the desired result.

□

5. CONCLUSION

This framework allows to prove formal bounds for nonlinear optimization problems. The SOS certificates checker benefits from a careful implementation of informal and formal libraries. The informal certification tool exploits the system properties of the problems to derive semialgebraic relaxations involving less SOS variables, thus more concise certificates. Our simple projection procedure yields SOS polynomials with arbitrary-size rational coefficients, that are efficiently checked on the Coq side, thanks to the machine modular arithmetic. The formal libraries can currently verify medium size semialgebraic certificates for global optimization problems and inequalities arising in the proof of Kepler's Conjecture. The implementation of polynomial arithmetic still needs some streamlining, as checking ring equalities in Coq remains the bottleneck of our verification procedure. A possible workaround to handle larger size problems is to use polynomials with interval coefficients as in [BJMD+12], so that one could obtain formal bounds without computing the exact polynomial remainder ϵ_{pop}. We plan to complete the formal verification procedure by additionally automatizing in Coq the proof of correctness of the maxplus semialgebraic approximations. A topic of further investigation is to evaluate the resulting improved methodology on all Flyspeck inequalities as well as on the sample of global optimization problems informally solved in [MAGW].

References

[AFG+11] Michael Armand, Germain Faure, Benjamin Grégoire, Chantal Keller, Laurent Théry, and Benjamin Werner. A Modular Integration of SAT/SMT Solvers to Coq through Proof Witnesses. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs*, volume 7086 of *Lecture Notes in Computer Science*, pages 135–150. Springer Berlin Heidelberg, 2011.

[AGK05] Marianne Akian, Stéphane Gaubert, and Vassili Kolokoltsov. Set coverings and invertibility of Functional Galois Connections. In G.L. Litvinov and V.P. Maslov, editors, *Idempotent Mathematics and Mathematical Physics*, volume 377 of *Contemporary Mathematics*, pages 19–51. American Mathematical Society, 2005. Also ESI Preprint 1447, http://arXiv.org/abs/math.FA/0403441.

[AGL08] M. Akian, S. Gaubert, and A. Lakhoua. The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. *SIAM J. Control Optim.*, 47(2):817–848, 2008.

[AGMW13a] Xavier Allamigeon, Stéphane Gaubert, Victor Magron, and Benjamin Werner. Certification of bounds of non-linear functions : the templates method, 2013. Proceedings of Conferences on Intelligent Computer Mathematics, CICM 2013 Calculemus, Bath.

[AGMW13b] Xavier Allamigeon, Stéphane Gaubert, Victor Magron, and Benjamin Werner. Certification of inequalities involving transcendental functions: combining SDP and max-plus approximation, 2013. Proceedings of the European Control Conference, ECC'13, Zurich.

[AGST10] Michaël Armand, Benjamin Grégoire, Arnaud Spiwack, and Laurent Théry. Extending Coq with Imperative Features and Its Application to SAT Verification. In Matt Kaufmann and LawrenceC. Paulson, editors, *Interactive Theorem Proving*, volume 6172 of *Lecture Notes in Computer Science*, pages 83–98. Springer Berlin Heidelberg, 2010.

[AKZ05] M. Montaz Ali, Charoenchai Khompatraporn, and Zelda B. Zabinsky. A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. *J. of Global Optimization*, 31(4):635–672, April 2005.

[AP10] Behzad Akbarpour and Lawrence Charles Paulson. MetiTarski: An Automatic Theorem Prover for Real-Valued Special Functions. *J. Autom. Reason.*, 44(3):175–205, March 2010.

[BB02] Henk Barendregt and Erik Barendsen. Autarkic computations in formal proofs. *J. Autom. Reason.*, 28(3):321–336, April 2002.

[Bes07] Frédéric Besson. Fast Reflexive Arithmetic Tactics: the linear case and beyond. In *Proceedings of the 2006 international conference on Types for proofs and programs*, TYPES'06, pages 48–62, Berlin, Heidelberg, 2007. Springer-Verlag.

[BJMD+12] Nicolas Brisebarre, Maria Joldes, Mioara, Érik Martin-Dorel, Micaela Mayero, Jean-Michel Muller, Ioana Pasca, Laurence Rideau, and Laurent Théry. Rigorous Polynomial Approximation using Taylor Models in Coq. In Alwyn Goodloe and Suzette Person, editors, *NASA Formal Methods 4th International Symposium, NFM 2012*, Lecture Notes in Computer Science, page 15, Norfolk, Virginia, États-Unis, April 2012. NASA, Springer.

[BM81] R. S. Boyer and J. S. Moore. Metafunctions: Proving Them Correct and Using Them Efficiently as New Proof Procedures. In *The*

Correctness Problem in Computer Science, pages 103–84. Academic Press, New York, 1981.

[BM11] S. Boldo and G. Melquiond. Flocq: A Unified Library for Proving Floating-Point Algorithms in Coq. In *Computer Arithmetic (ARITH), 2011 20th IEEE Symposium on*, pages 243–252, July 2011.

[Bou97] Samuel Boutin. Using Reflection to Build Efficient and Certified Decision Procedures. In *TACS'97. Springer-Verlag LNCS 1281*, pages 515–529. Springer-Verlag, 1997.

[BRB95] Gilles Barthe, Mark Ruys, and Henk Barendregt. A Two-Level Approach Towards Lean Proof-Checking. In Stefano Berardi and Mario Coppo, editors, *TYPES*, volume 1158 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 1995.

[CBBH08] A. Cuyt, F. Backeljauw, and C. Bonan-Hamada. *Handbook of Continued Fractions for Special Functions*. SpringerLink: Springer e-Books. Springer, 2008.

[CHJL11] S. Chevillard, J. Harrison, M. Joldeş, and Ch. Lauter. Efficient and accurate computation of upper bounds of approximation errors. *Theoretical Computer Science*, 412(16):1523 – 1543, 2011. Symbolic and Numerical Algorithms.

[FM00] W. H. Fleming and W. M. McEneaney. A Max-Plus-Based Algorithm for a Hamilton-Jacobi-Bellman Equation of Nonlinear Filtering. *SIAM J. Control Optim.*, 38(3):683–710, 2000.

[GAA+13] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A Machine-Checked Proof of the Odd Order Theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving*, volume 7998 of *Lecture Notes in Computer Science*, pages 163–179. Springer Berlin Heidelberg, 2013.

[GM05] Benjamin Grégoire and Assia Mahboubi. Proving Equalities in a Commutative Ring Done Right in Coq. In Joe Hurd and Thomas F. Melham, editors, *TPHOLs*, volume 3603 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2005.

[Gon08] Georges Gonthier. Computer mathematics. chapter The Four Colour Theorem: Engineering of a Formal Proof, pages 333–333. Springer-Verlag, Berlin, Heidelberg, 2008.

[GT06] B. Grégoire and L. Thery. A purely functional library for modular arithmetic and its application for certifying large prime numbers. In U. Furbach and N. Shankar, editors, *Proceedings of IJCAR'06*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 423–437. Springer-Verlag, 2006.

[GTW06] Benjamin Grégoire, Laurent Théry, and Benjamin Werner. A computational approach to pocklington certificates in type theory. In Masami Hagiya and Philip Wadler, editors, *Functional and Logic*

Programming, volume 3945 of *Lecture Notes in Computer Science*, pages 97–113. Springer Berlin Heidelberg, 2006.

[Hal94] Thomas C. Hales. A proof of the Kepler conjecture. *Math. Intelligencer*, 16:47–58, 1994.

[Hal03] Thomas C. Hales. Some Algorithms Arising in the Proof of the Kepler Conjecture. In Boris Aronov, Saugata Basu, János Pach, and Micha Sharir, editors, *Discrete and Computational Geometry*, volume 25 of *Algorithms and Combinatorics*, pages 489–507. Springer Berlin Heidelberg, 2003.

[Hal05] Thomas C. Hales. A proof of the Kepler conjecture. *Ann. of Math. (2)*, 162(3):1065–1185, 2005.

[Hal06] Thomas C. Hales. Introduction to the Flyspeck Project. In Thierry Coquand, Henri Lombardi, and Marie-Françoise Roy, editors, *Mathematics, Algorithms and Proofs*, number 05021 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[Har96] John Harrison. HOL Light: A Tutorial Introduction. In Mandayam K. Srivas and Albert John Camilleri, editors, *FMCAD*, volume 1166 of *Lecture Notes in Computer Science*, pages 265–269. Springer, 1996.

[Har07] John Harrison. Verifying nonlinear real formulas via sums of squares. In Klaus Schneider and Jens Brandt, editors, *Proceedings of the 20th International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2007*, volume 4732 of *Lecture Notes in Computer Science*, pages 102–118, Kaiserslautern, Germany, 2007. Springer-Verlag.

[HLL09] Didier Henrion, Jean-Bernard Lasserre, and Johan Lofberg. GloptiPoly 3: moments, optimization and semidefinite programming. *Optimization Methods and Software*, 24(4-5):pp. 761–779, August 2009.

[KLYZ12] Erich L. Kaltofen, Bin Li, Zhengfeng Yang, and Lihong Zhi. Exact Certification in Global Polynomial Optimization Via Sums-Of-Squares of Rational Functions with Rational Coefficients. *JSC*, 47(1):1–15, jan 2012. In memory of Wenda Wu (1929–2009).

[Las01] Jean B. Lasserre. Global Optimization with Polynomials and the Problem of Moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[LP10] Jean B. Lasserre and Mihai Putinar. Positivity and Optimization for Semi-Algebraic Functions. *SIAM Journal on Optimization*, 20(6):3364–3383, 2010.

[Mag14] Victor Magron. Nlcertify: A tool for formal nonlinear optimization. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, volume 8592 of *Lecture Notes in Computer Science*, pages 315–320. Springer Berlin Heidelberg, 2014.

[MAGW] Victor Magron, Xavier Allamigeon, Stéphane Gaubert, and Benjamin Werner. Certification of Real Inequalities – Templates and Sums of

Squares. Accepted for publication in Mathematical Programming Ser. B, special issue on Polynomial Optimization. arxiv:1403.5899, October 2014.

[MC11] David Monniaux and Pierre Corbineau. On the Generation of Positivstellensatz Witnesses in Degenerate Cases. In Marko Van Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk, editors, *Interactive Theorem Proving (ITP)*, volume 6898 of *Lecture Notes in Computer Science*, pages 249–264. Springer Verlag, August 2011.

[McE06] W. M. McEneaney. *Max-plus methods for nonlinear control and estimation*. Systems & Control: Foundations & Applications. Birkhäuser Boston Inc., Boston, MA, 2006.

[Mel12] Guillaume Melquiond. Floating-point arithmetic in the Coq system. *Information and Computation*, 216(0):14 – 23, 2012. Special Issue: 8th Conference on Real Numbers and Computers.

[MN13] César Muñoz and Anthony Narkawicz. Formalization of Bernstein Polynomials and Applications to Global Optimization. *Journal of Automated Reasoning*, 51(2):151–196, 2013.

[ORS92] S. Owre, J. M. Rushby, , and N. Shankar. PVS: A Prototype Verification System. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, jun 1992. Springer-Verlag.

[PM96] C. Paulin-Mohring. Circuits as streams in Coq : Verification of a sequential multiplier. In S. Berardi and M. Coppo, editors, *Types for Proofs and Programs, TYPES'95*, volume 1158 of *Lecture Notes in Computer Science*, 1996.

[PP08] Helfried Peyrl and Pablo A. Parrilo. Computing sum of squares decompositions with rational coefficients. *Theor. Comput. Sci.*, 409(2):269–281, 2008.

[Put93] M. Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

[Sch05] Markus Schweighofer. Optimization of Polynomials on Compact Semialgebraic Sets. *SIAM Journal on Optimization*, 15(3):805–825, 2005.

[SH13] Alexey Solovyev and Thomas C. Hales. Formal verification of nonlinear inequalities with Taylor Interval Approximations. *CoRR*, abs/1301.1702, 2013.

[Sol12] Alexey Solovyev. *Formal Computations and Methods*. PhD thesis, University of Pittsburgh, 2012.

[Spi06] Arnaud Spiwack. Ajouter des entiers machine à Coq. Master's thesis, 2006.

[SSM05] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Scalable Analysis of Linear Systems using Mathematical Programming. In Radhia Cousot, editor, *Proc. of Verification, Model Checking and*

Abstract Interpretation (VMCAI), volume 3385, pages 21–47, Paris, France, January 2005. Springer Verlag.

[Ste74] Gilbert Stengle. A nullstellensatz and a positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207(2):87–97, 1974.

[WKKM06] Hayato Waki, Sunyoung Kim, Masakazu Kojima, and Masakazu Muramatsu. Sums of Squares and Semidefinite Programming Relaxations for Polynomial Optimization Problems with Structured Sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006.

[YFN+10] Makoto Yamashita, Katsuki Fujisawa, Kazuhide Nakata, Maho Nakata, Mituhiro Fukuda, Kazuhiro Kobayashi, and Kazushige Goto. A high-performance software package for semidefinite programs : SDPA7. Technical report, Dept. of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 2010.

[Zum08] Roland Zumkeller. *Rigorous Global Optimization*. PhD thesis, École Polytechnique, 2008.