Formalization of a sign determination algorithm in real algebraic geometry

Cyril Cohen

Formalization of a sign determination algorithm in real algebraic geometry

Cyril Cohen
cyril.cohen@inria.fr
Université Cote d'Azur, Inria
France

Abstract

One of the problems in real algebraic geometry is root counting. Given a polynomial, we want to count the number of roots that satisfies constraints expressed as polynomial inequalities, this is done through a process called "sign determination". A naive way is to compute an exponential number of time consuming quantities, called Tarski Queries. In this paper, we formalize the construction of the adapted matrix part of the "better sign determination" algorithm from "Algorithms in Real Algebraic Geometry" (Basu, Pollack, Roy). We prove in Coq that the matrix has the properties expected from the book: small size but still invertible.

Introduction

One of the main algorithms in real algebraic geometry is the cylindrical algebraic decomposition [5]. The purpose of such an algorithm is to give a precise representation of a partitioning of the space described by polynomial equations.

Many algorithms that are connected to the task of finding such a decomposition are described in a book by Basu, Pollack and Roy [1], which is a reference on algorithms in real algebraic geometry. In this paper, we focus on the formalization of one of the sign determination algorithms described in the book. The purpose of this algorithm is to count the number of roots of a given polynomial, that satisfy polynomial constraints. In this paper we formalize the "better sign determination algorithm" from the book.

In Section 1 we introduce the naive sign determination from the book. In Section 2 we summarize the better sign determination, with the strict minimal requirements for a formal proof, thus contributing to finding a minimal presentation of the solution. In Section 3 we provide the details of the formalization. As another contribution, our proof generalizes the standard one from the book and applies it to the particular matrix from Section 1, thus giving a slightly more abstract justification for the algorithm.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Quantifier elimination on the theory of real closed fields

1.1 Solving an existential statement by root counting

Given a formula using $0, 1, +, \times, \cdot^{-1}, =, \leq, \vee, \wedge, \neg, \exists, \forall$, we want to compute a quantifier free formula which is equivalent in the theory of real closed fields. The theory of real closed fields is the theory of ordered fields together with the intermediate value property for polynomials.

The problem reduces to eliminating one existential quantifier in a formula of the form:

$$\exists x, P(x) = 0 \wedge \bigwedge_i \operatorname{sign}(Q_i(x)) = \sigma_i$$

where P and Q_i are polynomials and $\sigma_i \in \{0, 1, -1\}$. An explanation of this reduction can be found in our main reference [1] as well as in formal developments in Coq, [2, 4] but it is irrelevant in this work.

Now solving the previous existential statements can be done by counting the number of roots of P, such that each Q_i satisfies the sign conditions $\sigma_i \in \{0, 1, -1\}$ and checking its positivity.

More formally, the quantity

$$\begin{aligned}\operatorname{cnt}\left(P, \vec{Q}, \vec{\sigma}\right) &= \#\{x \in \operatorname{roots}(P) \mid \forall i, \operatorname{sign}(Q_i(x)) = \sigma_i\} \\ &= \sum_{x \in \operatorname{roots}(P)} \prod_i \delta_{\operatorname{sign}(Q_i(x)),\, \sigma_i}\end{aligned}$$

is positive if and only iff the former statement is true.

1.2 Tarski queries and counting roots

Given a polynomial P which roots we consider, and a polynomial Q which signs we consider, their Tarski Query (denoted $\operatorname{TaQ}(P, Q)$) may be defined as follows:

$$\operatorname{TaQ}(P, Q) = \sum_{x \in \operatorname{roots}(P)} \operatorname{sign}(Q(x)).$$

There is an algorithmic way to compute this quantity using solely the coefficients of P and Q. It is described in the literature [1] and formalized in [2, 4], but it is again irrelevant

to the present work, and from now on we assume we can compute these Tarski Queries as a black-box. The purpose of the present work is to formalize a technique described in [1] to minimize the number of calls to this black-box, as well as the size of the polynomials Q that are fed to this black-box.

The only relation that matters in the present work is the relation between the Tarski Queries and the previous counting $\mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right)$. Let us describe it for a single polynomial first, i.e. $\vec{Q} = (Q_0)$

1.2.1 Base case: $n = 1$.

Let us abbreviate

$$\mathrm{C}_{\sigma_0} = \mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right) = \sum_{x \in \mathrm{roots}(P)} \delta_{\mathrm{sign}(Q_0(x)),\ \sigma_0},$$

where $\sigma_0 \in \{0, 1, -1\}$, and

$$\mathrm{T}_{\alpha_0} = \mathrm{TaQ}\left(P, Q_0^{\alpha_0}\right) = \sum_{x \in \mathrm{roots}(P)} \mathrm{sign}\left(Q_0(x)\right),$$

where $\alpha_0 \in \{0, 1, 2\}$. We have,

$$\begin{array}{lcccccc} T_0 & = & \mathrm{C}_0 & + & \mathrm{C}_{+1} & + & \mathrm{C}_{-1} \\ T_1 & = & & & \mathrm{C}_{+1} & - & \mathrm{C}_{-1} \\ T_2 & = & & & \mathrm{C}_{+1} & + & \mathrm{C}_{-1} \end{array}$$

Otherwise said:

$$\begin{pmatrix} \mathrm{T}_0 & \mathrm{T}_1 & \mathrm{T}_2 \end{pmatrix} = \begin{pmatrix} \mathrm{C}_0 & \mathrm{C}_{+1} & \mathrm{C}_{-1} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}.$$

1.2.2 General case.

Using multiple calls of $\mathrm{TaQ}\left(P, \vec{Q}^{\vec{\alpha}}\right)$, with

$$\vec{Q}^{\vec{\alpha}} = \prod_i Q_i^{\alpha_i},$$

where $\alpha \in \{0, 1, 2\}^n$, there is already a formal proof [4], by induction on n, of the following equalities

$$\mathrm{TaQ}\left(P, Q^{\vec{\alpha}}\right) = \sum_{\vec{\sigma}} \prod_i \sigma_i^{\alpha_i} \mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right),$$

for all $\vec{\alpha} \in \{0, 1, 2\}^n$. Otherwise said:

$$\left(\mathrm{TaQ}\left(P, \vec{Q}^{\vec{\alpha}}\right)\right)_{\vec{\alpha} \in \{0,1,2\}^n} = \left(\mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right)\right)_{\vec{\sigma} \in \{0,1,-1\}^n} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}^{\otimes n}$$

with the appropriate ordering of $\{0, 1, 2\}^n$ and $\{0, 1, -1\}^n$, and where

$$A^{\otimes n} = \underbrace{A \otimes \ldots \otimes A}_{n \text{ times}}.$$

Now, computing naively $\mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right)$ amounts to inverting a matrix of size 3^n and computing all possible $\mathrm{TaQ}\left(P, \vec{Q}^{\vec{\alpha}}\right)$ for all $\vec{\alpha} \in \{0, 1, 2\}^n$, indeed

$$\mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right) = \left(\mathrm{TaQ}\left(P, \vec{Q}^{\vec{\alpha}}\right)\right)_{\vec{\alpha} \in \{0,1,2\}^n} \cdot \mathrm{col}_{\vec{\sigma}} \left(\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}^{\otimes n} \right)^{-1}$$

2 Minimizing the linear system

As noted in our reference [1], since

$$\mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right) = \#\{x \in \mathrm{roots}\,(P) \mid \forall i, \mathrm{sign}\,(Q_i(x)) = \sigma_i\},$$

we have

$$\sum_{\vec{\sigma} \in \{0,1,-1\}^n} \mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right) \leq \#(\mathrm{roots}\,(P)) \leq \deg P$$

Hence, **at most** $\deg P$ of the $\mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right)$ are nonzero, i.e. **at most** $\deg P$ sign conditions σ are "realizable", i.e. such that the set $\{x \in \mathrm{roots}\,(P) \mid \forall i, \mathrm{sign}\,(Q_i(x)) = \sigma_i\}$ is non empty.

Let's call the set of realizable sign conditions Σ, we have

$$\Sigma = \left\{\vec{\sigma} \in \{0, 1, -1\}^n \;\middle|\; \mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right) \neq 0\right\} \subseteq \{0, 1, -1\}^n$$

and we showed we have $\#\Sigma \leq \deg P$.

Now, [1] essentially argues that for any such Σ one can find a set $\mathrm{Ada}\,(\Sigma)$, such that

- $\mathrm{Ada}\,(\Sigma)$ is a subset of $\{0, 1, 2\}^n$ which depends only on Σ and such that $\#\mathrm{Ada}\,(\Sigma) = \#\Sigma$,
- $\mathrm{Ada}\,(\Sigma)$ has "small products", i.e. for all $\vec{\alpha} \in \mathrm{Ada}\,(\Sigma)$,

  $$\#\{i \mid \alpha_i \neq 0\} \leq \log \#\Sigma,$$

  i.e. for all $\vec{\alpha} \in \mathrm{Ada}\,(\Sigma)$, $\vec{Q}^{\vec{\alpha}}$ is the product of at most $\log \#\Sigma$ polynomials Q_i or Q_i^2.
- If $\mathrm{mat}(\Sigma, A)$ is a sub-matrix of the tensor product, which depends only on Σ and any $A \subset \{0, 1, 2\}^n$, such that

  $$\mathrm{mat}(\Sigma, A)_{\vec{\sigma}, \vec{\alpha}} = \vec{\sigma}^{\vec{\alpha}} = \prod_i \sigma_i^{\alpha_i},$$

  abusing the notation to confuse $\vec{\sigma}, \vec{\alpha}$ and their indexes. Then, $\mathrm{mat}(\Sigma, \mathrm{Ada}\,(\Sigma))$ is invertible.

Assuming the existence of such a $\mathrm{Ada}\,(\Sigma)$, the system from 1.2 can be reduced to

$$\left(\mathrm{TaQ}\left(P, \vec{Q}^{\vec{\alpha}}\right)\right)_{\vec{\alpha} \in \mathrm{Ada}(\Sigma)} = \left(\mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right)\right)_{\vec{\sigma} \in \Sigma} \cdot \mathrm{mat}(\Sigma, \mathrm{Ada}\,(\Sigma))$$

hence

$$\mathrm{cnt}\left(P, \vec{Q}, \vec{\sigma}\right) = \left(\mathrm{TaQ}\left(P, \vec{Q}^{\vec{\alpha}}\right)\right)_{\vec{\alpha} \in \mathrm{Ada}(\Sigma)} \cdot \mathrm{col}_{\vec{\sigma}}\left(\mathrm{mat}(\Sigma, \mathrm{Ada}\,(\Sigma))\right)^{-1}$$

The size of the matrix to invert is smaller than $\deg P$, and the products $\vec{Q}^{\vec{\alpha}}$ are all relatively small, thus, the calls to $\mathrm{TaQ}\,(P, Q)$ are far less numerous than 3^n in many use-cases

and are guaranteed to be applied to relatively small values of Q.

The rest of the paper explains the constructive and formalized computation of $\mathrm{Ada}(\Sigma)$, when Σ is a subset of X^n for an arbitrary X, and gives a sufficient condition for an appropriate generalization of the matrix $\mathrm{mat}(\Sigma, \mathrm{Ada}(\Sigma))$ to be invertible.

3 Abstraction for the formal proof

There are three parts in the formal proof. The first part deals with the theory of extensions and restrictions of elements of X^n, for any finite type X. Then we develop the theory of adapted sub-matrices, generated by a square matrix M of size k, instantiating X with $[\![0, k-1]\!]$. And finally, we instantiate the former theory for $k = 3$ and specialize M to

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix}$$

3.1 Extensions and restrictions

We define five functions, ext : X -> X ^ n -> X ^ n.+1 adds an element at the beginning of a Cartesian product, while rem : X ^ n.+1 -> X removes one. We also define a function extset : X -> {set X ^ n} -> {set X ^ n.+1}, it takes the image of a set by ext. Now, let us consider a set S : {set X ^ n.+1} and a "tuple" r : X ^ n. The set compl S r : {set X} is the set of all possible completions of r so that they stay in S. Finally, Xi n S lists all possible restrictions of S in {set X ^ n} that have at least n.+1 completions in S.

```
Variables (n : nat) (X : finType).
Implicit Type (S : {set X ^ n.+1}).
Implicit Type (R : {set X ^ n}).
Implicit Types (s : X ^ n.+1) (r : X ^ n).
Implicit Types (x y : X) (m : nat).

(* Extension of s with one element x at the beginning *)
Definition ext x r : X ^ n.+1 :=
  [ffun i => oapp r x (unlift 0%R i)].

(* Removing the first element of s, i.e. inverse of ext *)
Definition rem s : X ^ n :=
  [ffun i => s (lift 0%R i)].

(* Extension of a set with an element x at the beginning. *)
Definition extset x R : {set X ^ n.+1} :=
  [set ext x s | s in R].

(* The set of possible completions of r by x in S *)
Definition compl S r : {set X} :=
  [set x | ext x r \in S].

(* The set of elements with at least m + 1 extensions in S *)
Definition Xi m S : {set X ^ n} :=
  [set r : X ^ n | m < #|compl S r|].
```

The main results are cancellation lemmas between ext and rem, as well as monotonicity lemmas for extset, compl and Xi:

```
Lemma subset_extset x (R R' : {set X ^ n} :
  (extset R \subset extset R') = (R \subset R').

Lemma subset_compl r (S S' : {set X ^ n.+1}) :
  S \subset S' -> compl S r \subset compl S' r.

Lemma subset_Xi m (S S' : {set X ^ n.+1}):
  S \subset S' -> Xi m S \subset Xi m S'.

Lemma leq_Xi (S : {set X ^ n.+1}) (m p : nat) :
  p <= m -> Xi m S \subset X p S.
```

3.2 Adapted families and matrices

Now, we specialize X := 'I_k (i.e. $X = [\![0, k-1]\!]$), with $k > 0$. (In Coq, we define k := k'.+1 to have a manifest positivity). We also consider a square matrix M of size k, and we define mat1 S for all S : {set X} to select a sub-matrix of M selecting the rows from S and taking the #|S| first columns: it is the left-most sub-matrix of M which selects rows from S.

```
Variables (F : fieldType) (k' : nat).
Let k := k'.+1.
Let X := 'I_k.
Variable M : 'M[F]_k.

Definition mat1 (S : {set X}) : 'M[F]_#|S| :=
  \matrix_(i, j) M (enum_val i) (inord j).
```

Now, the main contribution of this paper is to identify the sufficient condition that any left-most square sub-matrix is invertible.

```
Hypothesis row_free_mat1 :
  forall (S : {set X}), row_free (mat1 S).
```

We generalize the definition of the matrix mat as in Section 2, by making the product of all the corresponding entries (like in a tensor product).

```
Definition mat_coef n (i : X ^ n) (j : X ^ n) := \
    prod_k M (i k) (j k).

Definition mat n (S : {set X ^ n})
    (A : {set X ^ n}) : 'M_(#|S|, #|A|) :=
  \matrix_(i < #|S|, j < #|A|)
    mat_coef (enum_val i) (enum_val j).
```

From there, we can finally define what it means for a family A to be adapted to S, it is when the matrix is invertible, but invertibility is reserved to manifest square matrices, and there is no a priori reason why the cardinals #|A| and #|S| would be even provably equal. Hence we state the invertibility by asserting the two cardinals are equal and that the rows of the matrix form a free family. The latter condition is equivalent to invertibility when the matrix is square.

```
Definition adapted n
    (S : {set X ^ n}) (A : {set X ^ n}) :=
  (#|A| == #|S|) && row_free (mat S A).
```

Now, the only thing that remains is the construction of a concrete adapted family. We need to define $\mathrm{Ada}(S) \subset [\![0, k-1]\!]^n$, such that

- $\mathrm{Ada}(S) \subset [\![0, k-1]\!]^n$,
- $\#\mathrm{Ada}(S) = \#S$
- $\#\{i \mid \alpha_i \neq 0\} \leq \log \#S$
- $\mathrm{mat}(S, \mathrm{Ada}(S))$ is invertible.

3.3 Effective construction of the adapted family

The adapted family $\mathrm{Ada}(S)$ is defined recursively as the disjoint union of $0\Xi_0(S)$, $1\Xi_1(S), \ldots, (k-1)\Xi_{k-1}(S)$.

```
Fixpoint adapt n : {set X ^ n} -> {set X ^ n} :=
  if n isn't _.+1 then id else fun S =>
    \bigcup_(i < k) extset i (adapt (Xi i S)).
```

The intuition is the following: for elements of S whose restriction have i completions, we need to add all of $0, \ldots, i-1$ to the beginning of the adapted family, in order to have enough information to invert the matrix, hence when there are at least one completion, we add at least 0, when there are at least two completions, we also add $1, \ldots$, when there are at least i completions, we need to also add $i-1$ and so on.

With this definition, adapt is monotonous: giving it more elements gives a bigger family.

```
Lemma subset_adapt n (S S' : {set X ^ n}) :
  S \subset S' -> adapt S \subset adapt S'.
```

Moreover it is downward closed, indeed if it contains a "tuple" a, then it contains any tuple that has smaller element. This is obvious via the construction above.

```
Lemma adapt_down_closed n S (a b : X ^ n) :
    (forall i, b i <= a i) ->
  a \in adapt S -> b \in adapt S.
```

Now the main results are quite technical. In particular the inductions in mat_adapt_free is non trivial: there are two inductions, one on n, and the other one on #|compl S (rem t)|, where t is a well selected row in S (see the formal proof for more details).

```
Theorem mat_adapt_free n (S : {set X ^ n}) :
  row_free (mat S (adapt S)).

Lemma card_adapt n S : #|adapt S| = #|S|.
```

Finally, we reach the fact that adapt S is always adapted to S, and that the adapted family has less than $\log \#S$ products.

```
Theorem adapt_adapted n (S : {set T^n}) : adapted
    S (adapt S).

Proposition prop_10_84 n S (a : T^n) :
    a \in adapt S ->
  #|[set i | a i != 0]| <= trunc_log 2 #|S|.
```

3.4 Instantiation of the abstract sign determination

Finally getting back the result amounts to providing the right matrix, and proving that all its left-most sub-matrices are invertible. In order to do that we encode the bijection sign between 'I_3 and $\{0, 1, -1\}$, and we define the matrix ctmat3 in a concise way:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix} = \left((\text{if } i \geq 2 \text{ then} (-1) \text{ else } i)^j\right)_{(i<3, j<3)}$$

In Coq we write:

```
Definition sign (k : 'I_3) : rat :=
  if k >= 2 then -1 else k%:Q.

Definition ctmat3 :=
  \matrix_(i < 3, j < 3) sign i ^ j.
```

Now, in order to apply the theorem adapt_adapted from above, the only thing we need is the invertibility of left-most sub-matrices. This is done by computing the determinant of all possible left-most sub-matrices and checking they are non zero. In the current work, we need to do it by hand (in 65 lines), but ultimately, these determinants should be automatically computed, e.g. using a refinement mechanism à la CoqEAL. [3, 6]

Finally we get the expected results.

```
Lemma row_free_ctmat (S : {set 'I_3}) :
  row_free (mat1 ctmat3 S).

Theorem ctmat_adapted n (S : {set 'I_3 ^ n}) :
  adapted ctmat3 S (adapt S).
```

Related Work

This work is a potential "spare part" for the preexisting Coq development[2, 4] already relating Tarski Queries with root counting. It could be substituted directly inside the old code, or substituted during a refinement phase.

To the best of our knowledge, a related work in the proof assistant Isabelle/HOL [7] follows very closely the presentation from Coq [2, 4], hence the problem we address here is not dealt with. Another work in PVS [8] relies on the naive version, akin to the one in Section 1.2.2, except they iterate the tensor product of a square matrix of size 6 instead of 3.

In comparison our work is the only one to tackle "better variants" of the sign determination, even though for now we do not provide the full procedure, nor an executable version, it still is constructive and an effective program could in principle be derived (semi-)automatically.

Conclusion

The use of a proof assistant such as Coq in the formalization of these algorithms led us to write algorithms and conduct our proofs in a slightly different way than the one described in our reference [1]. The authors of the book then adopted some of our reformulations to modify their book, in order to make the description more precise or concise and to provide more detailed justifications.

The following difficulties were easily avoided thanks to the Mathematical Components Library [9].

- Re-indexing of big operators, and establishing ad-hoc bijections on-the-fly.
- Finding the induction for adapt_adapted (which was formerly left implicit in the book).
- Using matrices with judgmentally different but propositionally identical sizes.
- Set extensionality problems, thanks to finite sets.

Compared to the "pen and paper"/ "printed" literature, we bring several improvements.

The new formal proof of prop_10_84 and the intermediate lemmas were taken as a source of inspiration for the future revision of the book [1].

The new formal proof of adapt_adapted is more abstract, using an explicit induction, and reducing the invertibility to a simple property on a 3×3-matrix.

The Coq code contains less than 400 lines of specific development (definitions and proofs), which makes the *de Bruijn factor* remarkably smaller than 1 according to our visual estimation, and the reason looks sensible to us: cautiously explaining how to cut and reassemble matrices looks easier when programmed than when written and drawn.

Acknowledgments

I would like to thank Matthieu Kohli for his contribution to the early development of this proof. I thank Marie-Françoise Roy for the numerous discussions we had together that enlightened us with her expertise of the subject.

References

[1] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. 2006. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Algorithms and Computation in Mathematics, Vol. 10. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[2] Cyril Cohen. 2012. *Formalized algebraic numbers: construction and first order theory*. Ph.D. Dissertation. École polytechnique.

[3] Cyril Cohen, Maxime Dénès, and Anders Mörtberg. 2013. Refinements for Free!. In *Certified Programs and Proofs*. Melbourne, Australia, 147 – 162. https://doi.org/10.1007/978-3-319-03545-1_10

[4] Cyril Cohen and Assia Mahboubi. 2012. Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination. *Logical Methods in Computer Science* 8, 1:02 (Feb. 2012), 1–40. https://doi.org/10.2168/LMCS-8(1:02)2012

[5] George E. Collins. 1974. Quantifier elimination for real closed fields by cylindrical algebraic decomposition–preliminary report. *SIGSAM Bull.* 8 (August 1974), 80–90. Issue 3. https://doi.org/10.1145/1086837.1086852

[6] Maxime Dénès, Anders Mörtberg, and Vincent Siles. 2012. A Refinement-Based Approach to Computational Algebra in Coq. In *Interactive Theorem Proving*, Lennart Beringer and Amy Felty (Eds.). Lecture Notes in Computer Science, Vol. 7406. Springer Berlin Heidelberg, 83–98. https://doi.org/10.1007/978-3-642-32347-8_7

[7] Wenda Li, Grant Olney Passmore, and Lawrence C. Paulson. 2015. A Complete Decision Procedure for Univariate Polynomial Problems in Isabelle/HOL. *CoRR* abs/1506.08238 (2015). arXiv:1506.08238 http://arxiv.org/abs/1506.08238

[8] Anthony Narkawicz, César Muñoz, and Aaron Dutle. 2015. Formally-Verified Decision Procedures for Univariate Polynomial Computation Based on Sturm's and Tarski's Theorems. *Journal of Automated Reasoning* 54, 4 (2015), 285–326. https://doi.org/10.1007/s10817-015-9320-x

[9] The Mathematical Components Team. 2007 – 2020. The Mathematical Components Library. https://github.com/math-comp/math-comp.