Counting Polynomial Roots in Isabelle/HOL: A Formal Proof of the Budan-Fourier Theorem

Wenda Li
Department of Computer Science and Technology
University of Cambridge
United Kingdom
wl302@cam.ac.uk

Lawrence C. Paulson
Department of Computer Science and Technology
University of Cambridge
United Kingdom
lp15@cam.ac.uk

Abstract

Many problems in computer algebra and numerical analysis can be reduced to counting or approximating the real roots of a polynomial within an interval. Existing verified root-counting procedures in major proof assistants are mainly based on the classical Sturm theorem, which only counts distinct roots.

In this paper, we have strengthened the root-counting ability in Isabelle/HOL by first formally proving the Budan-Fourier theorem. Subsequently, based on Descartes' rule of signs and Taylor shift, we have provided a verified procedure to efficiently over-approximate the number of real roots within an interval, counting multiplicity. For counting multiple roots exactly, we have extended our previous formalisation of Sturm's theorem. Finally, we combine verified components in the developments above to improve our previous certified complex-root-counting procedures based on Cauchy indices. We believe those verified routines will be crucial for certifying programs and building tactics.

CCS Concepts • **Theory of computation → Higher order logic**; • **Computing methodologies → Algebraic algorithms**; *Representation of exact numbers*;

Keywords formal verification, theorem proving, Isabelle, the Budan-Fourier theorem, Descartes' rule of signs, counting polynomial roots

ACM Reference Format:
Wenda Li and Lawrence C. Paulson. 2019. Counting Polynomial Roots in Isabelle/HOL: A Formal Proof of the Budan-Fourier Theorem. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '19), January 14–15, 2019, Cascais, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3293880.3294092

1 Introduction

Counting the real and complex roots of a univariate polynomial has always been a fundamental task in computer algebra and numerical analysis. For example, given a routine that counts the number of real roots of a polynomial within an interval, we can compute each real root to arbitrary precision through bisection, in which sense we have solved the polynomial equation. Another example would be the Routh-Hurwitz stability criterion [3, Section 23][23, Chapter 9], where the stability of a linear system can be tested by deciding the number of complex roots of the characteristic polynomial within the left half-plane (left of the imaginary axis).

Numerous methods have been invented in the symbolic and numerical computing community to efficiently count (or test) real and complex roots of a polynomial [10, 32, 33]. However, in the theorem proving community, where procedures are usually formally verified in a foundational proof assistant (e.g., Coq, Isabelle, HOL and PVS), our choices are typically limited to relying on Sturm's theorem to count distinct roots within an interval through signed remainder sequences.

In this paper, we aim to reinforce our root-counting ability in the Isabelle theorem prover [26]. In particular, our main contributions are the following:

- We have mechanised a proof of the Budan-Fourier theorem and a subsequent roots test based on Descartes' rule of signs and Taylor shift. This roots test efficiently over-approximates the number of real roots within an open interval, counting multiplicity.
- We have made a novel extension to our previous formalisation of Sturm's theorem to count real roots *with* multiplicity.
- Benefited from the developments above, we have extended our previous verified complex-root-counting procedures to more general cases: zeros on the border are allowed when counting roots in a half-plane; we can now additionally count roots within a ball.

All results of this paper have been formalised in Isabelle/HOL without using extra axioms, and the source code is available from the following URL:

https://bitbucket.org/liwenda1990/src-cpp-2019

To reuse the results in this paper, consult our entries in the Archive of Formal Proofs [16, 17], which we will keep updating.

This paper continues as follows: after introducing some frequently used notations (§2), we first present a formal proof of the Budan-Fourier theorem (§3), which eventually leads to the Descartes roots test. Next, we extend our previous formalisation of the classical Sturm theorem to count multiple real roots (§4). As an application, we apply those newly-formalised results to improve our previous complex-root-counting procedures (§5). After that, we discuss related work (§6) and some experiments (§7). Finally, we conclude the paper in §8.

2 Notations

Below, we will present definitions and proofs in both natural language and the formal language of Isabelle. Some common notations are as follows:

- we often use `p` and `q` in our proof scripts to denote polynomials. However, when presenting them in mathematical formulas, we will switch to capitalised letters—P and Q.
- `poly p a` in Isabelle means $P(a)$: the value of the univariate polynomial P evaluated at the point a.
- $\mu(a, P)$ denotes the multiplicity/order of a as a root of the polynomial P. In Isabelle, this becomes `order a p`.
- $\overline{\mathbb{R}}$ denotes the extended real numbers:
$$\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}.$$
- $\text{Num}_{\mathbb{R}}(P; S)$ and $\text{Num}_{\mathbb{C}}(P; S)$ denote the number of real and complex roots of P *counting multiplicity* within the set S, while in Isabelle they both correspond to `proots_count p s`—they are distinguished by the type of the polynomial `p`.
- Similarly, $\text{NumD}_{\mathbb{R}}(P; S)$ and $\text{NumD}_{\mathbb{C}}(P; S)$ denote the number of *distinct* real and complex roots of P within S. In Isabelle, they correspond to
`card (proots_within p s)`
differentiated by the type of the polynomial `p`.

Many of the theorems presented in this paper will involve sign variations, so we give a definition here:

Definition 2.1 (Sign variations). Given a list of real numbers $[a_0, a_1, ..., a_n]$, we use $\text{Var}([a_0, a_1, ..., a_n])$ to denote the number of *sign variations* after dropping zeros. Additionally, we abuse the notation by letting

$$\begin{aligned}\text{Var}([P_0, P_1, ..., P_n]; a) &= \text{Var}([P_0(a), P_1(a), ..., P_n(a)])\\ \text{Var}([P_0, P_1, ..., P_n]; a, b) &= \text{Var}([P_0, P_1, ..., P_n]; a)\\ &\quad - \text{Var}([P_0, P_1, ..., P_n]; b),\end{aligned}$$

where $[P_0, P_1, ..., P_n]$ is a sequence of univariate polynomials, and $\text{Var}([P_0, P_1, ..., P_n]; a)$ is interpreted as the number of sign variations of $[P_0, P_1, ..., P_n]$ evaluated at a. Finally, given $P(x) = a_0 + a_1x + \cdots + a_nx^n$, we also use $\text{Var}(P)$ to denote sign variations of the coefficient sequence of P:

$$\text{Var}(P) = \text{Var}([a_0, a_1, ..., a_n]).$$

Example 2.2. By Definition 2.1, we can have calculations like the following :

$$\begin{aligned}\text{Var}([1, -2, 0, 3]) &= \text{Var}([1, -2, 3]) = 2,\\ \text{Var}([x^2, x-2]; 0, 1) &= \text{Var}([x^2, x-2]; 0) - \text{Var}([x^2, x-2]; 1)\\ &= \text{Var}([0, -2]) - \text{Var}([1, -1])\\ &= 0 - 1 = -1,\\ \text{Var}(1 - x^2 + 2x^3) &= \text{Var}([1, 0, -1, 2]) = 2.\end{aligned}$$

3 From the Budan-Fourier Theorem to the Descartes Roots Test

In this section, we first formalise the proof of the Budan-Fourier theorem. We then apply this to derive Descartes' rule of signs, which effectively over-approximates the number of positive real roots (counting multiplicity) of a real polynomial by calculating the sign variations of its coefficient sequence. We then use this to show the *Descartes roots test*[1]: given a polynomial $P \in \mathbb{R}[x]$ of degree n and a bounded interval $I = (a, b)$, we can apply Descartes' rule of signs to a base-transformed polynomial

$$P_I(x) = (x+1)^n P\left(\frac{ax+b}{x+1}\right), \qquad (1)$$

to over-approximate the number of real roots of P over (a, b). Note that the base transformation (1) is commonly referred as *Taylor shift* in the literature [14].

Our formal proof of the Budan-Fourier theorem and Descartes' rule of signs roughly follows the textbook by Basu et al. [5], while that of the Descartes roots test is inspired by various sources [8, 13, 14].

3.1 The Budan-Fourier Theorem

Definition 3.1 (Fourier sequence). Let P be a univariate polynomial of degree n. The *Fourier sequence* of P is generated through polynomial derivatives:

$$\text{Der}(P) = [P, P', ..., P^{(n)}].$$

Theorem 3.2 (The Budan-Fourier theorem). *Let $P \in \mathbb{R}[x]$, a, b be two extended real numbers (i.e., $a, b \in \overline{\mathbb{R}}$) such that $a < b$. Through Fourier sequences and sign variations, the Budan-Fourier theorem over-approximates* $\text{Num}_{\mathbb{R}}(P; (a, b])$ *and the difference is an even number:*

- $\text{Var}(\text{Der}(P); a, b) \geq \text{Num}_{\mathbb{R}}(P; (a, b])$
- *and* $\text{Var}(\text{Der}(P); a, b) - \text{Num}_{\mathbb{R}}(P; (a, b])$ *is even.*

[1] There does not seem to be a uniform name for this test [1, 2, 9]—here we follow the one used in Arno Eigenwillig's PhD thesis [13] where he refers this test as "the Descartes test for roots".

To prove Theorem 3.2, the key idea is to examine sign variations near a root of P:

$$\mathrm{Var}(\mathrm{Der}(P); c-\epsilon, c) \quad \text{and} \quad \mathrm{Var}(\mathrm{Der}(P); c, c+\epsilon),$$

where c is a possible root of P and ϵ is a small real number.

Regarding $\mathrm{Var}(\mathrm{Der}(P); c-\epsilon, c)$, the property we have derived in Isabelle/HOL is the following:

Lemma 3.3 (budan_fourier_aux_left').

```
  fixes c d₁::real and p::"real poly"
  assumes "d₁ < c" and "p ≠ 0"
  assumes non_zero:"∀x. d₁ ≤ x ∧ x < c ⟶
            (∀q ∈ set (pders p). poly q x ≠ 0)"
  shows "changes_itv_der d₁ c p ≥ order c p"
        "even (changes_itv_der d₁ c p - order c p)"
```

where

- `order c p` is $\mu(c, P)$: the order/multiplicity of c as a root of the polynomial `p`, as we described in §2;
- `changes_itv_der d₁ c p` stands for $\mathrm{Var}(\mathrm{Der}(P); d_1, c)$;
- the assumption `non_zero` asserts that $Q(x) \neq 0$ (i.e., `poly q x ≠ 0`) for all $x \in [d_1, c)$ and $Q \in \mathrm{Der}(P)$. Here, `pders p` is the Fourier sequence (i.e., $\mathrm{Der}(P)$) and `set (pders p)` converts this sequence/list into a set of polynomials.

Essentially, d_1 in Lemma 3.3 can be considered as $c-\epsilon$, since d_1 is asserted to be closer to c from the left than any root of polynomials in $\mathrm{Der}(P)$. Therefore, Lemma 3.3 claims that $\mathrm{Var}(\mathrm{Der}(P); c-\epsilon, c)$ always exceeds $\mu(c, P)$ by an even number.

Proof of Lemma 3.3. By induction on the degree of P. For the base case (i.e., the degree of P is zero), the proof is trivial since both $\mathrm{Var}(\mathrm{Der}(P); d_1, c)$ and $\mu(c, P)$ are equal to 0.

For the inductive case, through the induction hypothesis, we have

$$\begin{aligned}&\mathrm{Var}(\mathrm{Der}(P'); d_1, c) \geq \mu(c, P') \\ &\qquad \wedge \operatorname{even}(\mathrm{Var}(\mathrm{Der}(P'); d_1, c) - \mu(c, P')).\end{aligned} \tag{2}$$

First, we consider the case when $P(c) = 0$. In this case, we can derive

$$\mu(c, P) = \mu(c, P') + 1, \tag{3}$$

$$\mathrm{Var}(\mathrm{Der}(P); d_1) = \mathrm{Var}(\mathrm{Der}(P'); d_1) + 1, \tag{4}$$

$$\mathrm{Var}(\mathrm{Der}(P); c) = \mathrm{Var}(\mathrm{Der}(P'); c). \tag{5}$$

Combining (2), (3), (4) and (5) yields

$$\begin{aligned}&\mathrm{Var}(\mathrm{Der}(P); d_1, c) \geq \mu(c, P) \\ &\qquad \wedge \operatorname{even}(\mathrm{Var}(\mathrm{Der}(P); d_1, c) - \mu(c, P)),\end{aligned} \tag{6}$$

which concludes the proof.

As for $P(c) \neq 0$, we can similarly have

$$\begin{aligned}&\mathrm{Var}(\mathrm{Der}(P); c) \\ &= \begin{cases} \mathrm{Var}(\mathrm{Der}(P'); c), & \text{if } P'(c+\epsilon) > 0 \\ & \quad \leftrightarrow P(c) > 0, \\ \mathrm{Var}(\mathrm{Der}(P'); c) + 1, & \text{otherwise,} \end{cases}\end{aligned} \tag{7}$$

$$\begin{aligned}&\mathrm{Var}(\mathrm{Der}(P); d_1) \\ &= \begin{cases} \mathrm{Var}(\mathrm{Der}(P'); d_1), & \text{if } \operatorname{even}(\mu(c, P')) \\ & \quad \leftrightarrow P'(x+\epsilon) > 0 \\ & \quad \leftrightarrow P(c) > 0, \\ \mathrm{Var}(\mathrm{Der}(P'); d_1) + 1, & \text{otherwise.} \end{cases}\end{aligned} \tag{8}$$

where $\leftrightarrow$ is the equivalence function in propositional logic. By putting together (3), (7) and (8), we can derive (6) through case analysis, and conclude the whole proof. □

Considering $\mathrm{Var}(\mathrm{Der}(P); c, c+\epsilon)$, we have an analogous proposition:

Lemma 3.4 (budan_fourier_aux_right).

```
  fixes c d₂::real and p::"real poly"
  assumes "c < d2" and "p ≠ 0"
  assumes "∀x. c < x ∧ x ≤ d₂ ⟶
            (∀q ∈ set (pders p). poly q x ≠ 0)"
  shows "changes_itv_der c d₂ p = 0"
```

which indicates that $\mathrm{Var}(\mathrm{Der}(P); c, c+\epsilon) = \mathrm{Var}(\mathrm{Der}(P); c, d_2) = 0$, since d_2 can be treated as $c+\epsilon$.

Proof of Lemma 3.4. Similar to that of Lemma 3.3: by induction on the degree of P and case analysis. □

With Lemma 3.4, we can generalise Lemma 3.3 a bit by allowing $P(d_1) = 0$ in the assumption:

Lemma 3.5 (budan_fourier_aux_left).

```
  fixes c d₁::real and p::"real poly"
  assumes "d₁ < c" and "p ≠ 0"
  assumes non_zero:"∀x. d₁ < x ∧ x < c ⟶
            (∀q ∈ set (pders p). poly q x ≠ 0)"
  shows "changes_itv_der d₁ c p ≥ order c p"
        "even (changes_itv_der d₁ c p - order c p)"
```

Proof. Let $d = (d_1 + c)/2$. Lemma 3.4 and 3.3 respectively yield

$$\mathrm{Var}(\mathrm{Der}(P); d_1, d) = 0, \tag{9}$$

$$\begin{aligned}&\mathrm{Var}(\mathrm{Der}(P); d, c) \geq \mu(c, P) \\ &\qquad \wedge \operatorname{even}(\mathrm{Var}(\mathrm{Der}(P); d, c) - \mu(c, P)).\end{aligned} \tag{10}$$

Moreover, by definition,

$$\begin{aligned}&\mathrm{Var}(\mathrm{Der}(P); d_1, c) \\ &\qquad = \mathrm{Var}(\mathrm{Der}(P); d_1, d) + \mathrm{Var}(\mathrm{Der}(P); d, c).\end{aligned} \tag{11}$$

From (9), (10) and (11), the conclusion follows. □

Finally, we come to our mechanised statement of the bounded interval case of Theorem 3.2:

Theorem 3.6 (budan_fourier_interval).

```
fixes a b::real and p::"real poly"
assumes "a < b" and "p ≠ 0"
shows "changes_itv_der a b p
          ≥ proots_count p {x. a < x ∧ x ≤ b} ∧
        even (changes_itv_der a b p
               - proots_count p {x. a < x ∧ x ≤ b})"
```

where `proots_count p {x. a < x ∧ x ≤ b})` denotes $\mathrm{Num}_{\mathbb{R}}(P;(a,b])$, which is the number of real roots of P (counting multiplicity) within the interval $(a,b]$.

Proof of Theorem 3.6. By induction on the number of roots of polynomials in $\mathrm{Der}(P)$ within the interval (a,b). For the base case, we have

$$Q(x) \neq 0, \qquad \text{for all } x \in (a,b) \text{ and } Q \in \mathrm{Der}(P), \tag{12}$$

and then, by Lemma 3.5,

$$\mathrm{Var}(\mathrm{Der}(P);a,b) \geq \mu(b,P) \\ \wedge \mathrm{even}(\mathrm{Var}(\mathrm{Der}(P);a,b) - \mu(b,P)). \tag{13}$$

In addition, (12) also leads to

$$\mathrm{Num}_{\mathbb{R}}(P;(a,b]) = \mu(b,P). \tag{14}$$

We finish the base case by combining (13) and (14).

Regrading the inductive case, let b' be the largest root within the interval (a,b) of the polynomials from $\mathrm{Der}(P)$:

$$b' = \max\{x \in (a,b) \mid \exists Q \in \mathrm{Der}(P).\ Q(x) = 0\}. \tag{15}$$

With the induction hypothesis, we have

$$\mathrm{Num}_{\mathbb{R}}(P;(a,b']) \leq \mathrm{Var}(\mathrm{Der}(P);a,b') \\ \wedge \mathrm{even}(\mathrm{Num}_{\mathbb{R}}(P;(a,b']) - \mathrm{Var}(\mathrm{Der}(P);a,b')). \tag{16}$$

Also, considering there is no root of P within (b',b) (otherwise it will be larger than b', contradicting (15)), we have

$$\mathrm{Num}_{\mathbb{R}}(P;(a,b]) = \mathrm{Num}_{\mathbb{R}}(P;(a,b']) + \mu(b,P). \tag{17}$$

Finally, Lemma 3.5 yields

$$\mathrm{Var}(\mathrm{Der}(P);b',b) \geq \mu(b,P) \\ \wedge \mathrm{even}(\mathrm{Var}(\mathrm{Der}(P);b',b) - \mu(b,P)). \tag{18}$$

Putting together (16), (17), and (18) finishes the proof. □

Note that Theorem 3.6 only corresponds to the bounded interval case of Theorem 3.2. In the formal development, we also have versions for $a = -\infty$, $b = +\infty$ or both.

An interesting corollary of the Budan-Fourier theorem is that when all roots are real, the over-approxiamation (i.e., $\mathrm{Var}(\mathrm{Der}(P);a,b)$) becomes exact:

Corollary 3.7 (budan_fourier_real).

```
fixes a b::real and p::"real poly"
assumes "all_roots_real p"
shows
  "proots_count p {x. x ≤ a} = changes_le_der a p"
  "a < b ⟶ proots_count p {x. a < x ∧ x ≤ b}
                         = changes_itv_der a b p"
  "proots_count p {x. b < x} = changes_gt_der b p"
```

where

- `all_roots_real p` is formally defined as every *complex* root of P having a zero imaginary part,
- `changes_le_der a p` encodes $\mathrm{Var}(\mathrm{Der}(P);-\infty,a)$,
- `changes_itv_der a b p` encodes $\mathrm{Var}(\mathrm{Der}(P);a,b)$,
- `changes_gt_der b p` encodes $\mathrm{Var}(\mathrm{Der}(P);b,+\infty)$.

Proof of Corollary 3.7. Let

$$t_1 = \mathrm{Var}(\mathrm{Der}(P);-\infty,a) - \mathrm{Num}_{\mathbb{R}}(P;(-\infty,a])$$
$$t_2 = \mathrm{Var}(\mathrm{Der}(P);a,b) - \mathrm{Num}_{\mathbb{R}}(P;(a,b])$$
$$t_3 = \mathrm{Var}(\mathrm{Der}(P);a,b) - \mathrm{Num}_{\mathbb{R}}(P;(b,+\infty))$$

As a result of Theorem 3.2, we have

$$t_1 \geq 0 \wedge t_2 \geq 0 \wedge t_3 \geq 0. \tag{19}$$

Additionally, by the definition of Var we derive

$$\mathrm{Var}(\mathrm{Der}(P);-\infty,a) + \mathrm{Var}(\mathrm{Der}(P);a,b) \\ + \mathrm{Var}(\mathrm{Der}(P);a,+\infty) = \deg(P), \tag{20}$$

and the assumption (i.e., all roots are real) brings us

$$\mathrm{Num}_{\mathbb{R}}(P;(-\infty,a]) + \mathrm{Num}_{\mathbb{R}}(P;(a,b]) \\ + \mathrm{Num}_{\mathbb{R}}(P;(b,+\infty)) = \deg(P). \tag{21}$$

Joining (20) with (21) yields

$$t_1 + t_2 + t_3 = 0. \tag{22}$$

Finally, putting (19) and (22) together concludes the proof. □

3.2 Descartes' Rule of Signs

Given $a, b \in \overline{\mathbb{R}}$, $a < b$ and a polynomial $P \in \mathbb{R}[x]$, the Budan-Fourier theorem (Theorem 3.2) in the previous section grants us an effective way to over-approximate $\mathrm{Num}_{\mathbb{R}}(P;(a,b])$ (by an even number) through calculating $\mathrm{Var}(\mathrm{Der}(P);a,b)$.

Nevertheless, the approximation $\mathrm{Var}(\mathrm{Der}(P);a,b)$ still requires calculating a Fourier sequence ($\mathrm{Der}(P)$) and a series of polynomial evaluations. When $a = 0$ and $b = +\infty$, the approximation can be refined to counting the number of sign variations of the coefficient sequence of P, which requires almost no calculation! Approximating $\mathrm{Num}_{\mathbb{R}}(P;(0,+\infty))$ using $\mathrm{Var}(P)$ (rather than $\mathrm{Var}(\mathrm{Der}(P);0,+\infty)$) is the celebrated Descartes' rule of signs:

Theorem 3.8 (descartes_sign).

```
fixes p::"real poly"
assumes "p ≠ 0"
shows "changes (coeffs p)
```

```
        ≥ proots_count p {x. 0 < x} ∧
      even (changes (coeffs p)
           - proots_count p {x. 0 < x})"
```

where `changes (coeffs p)` encodes $\mathrm{Var}(P)$—sign variations of the coefficient sequence of P.

Proof. Let $P = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} + a_nx^n$. $\mathrm{Der}(P)$ is as follows:

$$\begin{aligned}[&a_0 + a_1x + a_2x^2 + \cdots + x_{n-1}x^{n-1} + a_nx^n,\\ &a_1 + 2a_2x + \cdots + (n-1)a_{n-1}x^{n-1} + na_nx^{n-1},\\ &\vdots\\ &(n-1)!a_{n-1} + n!a_nx\\ &n!a_n\\ ]\end{aligned} \tag{23}$$

where $n!$ is the factorial of n. From (23), it can be derived that $\mathrm{Der}(P)$ has no sign variation when evaluated at $+\infty$:

$$\mathrm{Var}(\mathrm{Der}(P); +\infty) = 0. \tag{24}$$

Also, evaluating $\mathrm{Der}(P)$ at 0 gives $[a_0, a_1, ..., (n-1)!a_{n-1}, n!a_n]$, hence its sign variations should equal $[a_0, a_1, ..., a_{n-1}, a_n]$:

$$\mathrm{Var}(\mathrm{Der}(P); 0) = \mathrm{Var}(P). \tag{25}$$

Joining (24) and (25) gives $\mathrm{Var}(\mathrm{Der}(P); 0, +\infty) = \mathrm{Var}(P)$, with which we apply Theorem 3.2 to finish the proof. □

3.3 Base Transformation and Taylor Shift

Given $P \in \mathbb{R}[x]$, with Descartes' rule of signs (Theorem 3.8) in the previous section, we can efficiently approximate $\mathrm{Num}_{\mathbb{R}}(P; (0, +\infty))$. However, in many cases, we are interested in roots within a bounded interval: $\mathrm{Num}_{\mathbb{R}}(P; I)$, where $I = (a, b)$ and $a, b \in \mathbb{R}$. Can we still exploit the efficiency from Descartes' rule of signs? The answer is yes, via an operation to transform P into $P_I \in \mathbb{R}[x]$ such that

$$\mathrm{Num}_{\mathbb{R}}(P; I) = \mathrm{Num}_{\mathbb{R}}(P_I; (0, +\infty)). \tag{26}$$

In order to develop this transformation operation, we first define the composition of a univariate polynomial and a rational function (i.e., a function of the form $f(x) = P(x)/Q(x)$, where P and Q are polynomials) in Isabelle/HOL:

```
definition fcompose::
   "'a ::field poly ⇒ 'a poly ⇒ 'a poly ⇒ 'a poly"
  where
   "fcompose p q₁ q₂ =
       fst (fold_coeffs (λa (r₁,r₂).
            (r₂ * [:a:] + q₁ * r₁,q₂ * r₂)) p (0,1))"
```

where

- q₁ and q₂ are respectively the numerator and denominator of a rational function,
- `fst` gives the first part of a pair,
- `[:a:]` is a constant polynomial lifted from the value a.

Also, `fold_coeffs` is the classical foldr operation on a coefficient sequence:

```
definition fold_coeffs ::
   "('a::zero ⇒ 'b ⇒ 'b) ⇒ 'a poly ⇒ 'b ⇒ 'b"
  where "fold_coeffs f p = foldr f (coeffs p)"
```

Essentially, let P, Q_1, and Q_2 be three univariate polynomials over some field such that P is of degree n. Our composition operation over these three polynomials (i.e., `fcompose p q₁ q₂`) gives the following polynomial:

$$(Q_2(x))^n P\left(\frac{Q_1(x)}{Q_2(x)}\right). \tag{27}$$

The idea of (27) can be illustrated by the following mechanised lemma:

Lemma 3.9 (`poly_fcompose`).

```
  fixes p q₁ q₂::"'a::field poly"
  assumes "poly q₂ x ≠ 0"
  shows "poly (fcompose p q₁ q₂) x =
                   poly p (poly q₁ x / poly q₂ x)
                       * (poly q₂ x) ^ (degree p)"
```

where `poly p x` gives the value of the polynomial `p` when evaluated at `x`.

When $Q_1(x) = a + bx$ and $Q_2(x) = 1 + x$, (27) yields a transformation (i.e., Taylor shift):

$$P_I(x) = (x+1)^n P\left(\frac{ax+b}{x+1}\right), \tag{28}$$

with which we have achieved (26):

Lemma 3.10 (`proots_count_pos_interval`).

```
  fixes a b::real and p::"real poly"
  assumes "p ≠ 0" and "a < b"
  shows "proots_count p {x. a < x ∧ x < b} =
          proots_count (fcompose p [:b,a:] [:1,1:])
                                              {x. 0 < x}"
```

where

- `[:b,a:]` encodes the polynomial $b + ax$,
- `[:1,1:]` stands for the polynomial $1 + x$.

3.4 The Descartes Roots Test

Finally, we come to the Descartes roots test. Given $P \in \mathbb{R}[x]$, $a, b \in \mathbb{R}$ and $I = (a, b)$, the Descartes roots test is $\mathrm{Var}(P_I)$: the number of sign variations on the coefficient sequence of the Taylor-shifted polynomial P_I:

```
definition descartes_roots_test::
   "real ⇒ real ⇒ real poly ⇒ nat"
  where
   "descartes_roots_test a b p = nat (changes
           (coeffs (fcompose p [:b,a:] [:1,1:])))"
```

where

- `fcompose p [:b,a:] [:1,1:]` encodes Taylor shift as in (28),
- `coeffs` converts a polynomial into its coefficient sequence,
- `changes` calculates the number of sign variations (i.e., Var),

- `nat` converts an integer into a natural number.

Just like $\mathrm{Var}(\mathrm{Der}(P); a, b)$, whose root approximation property has been reflected in Theorem 3.6, $\mathrm{Var}(P_I)$ has a similar theorem related to $\mathrm{Num}_{\mathbb{R}}(P; (a, b))$:

Theorem 3.11 (`descartes_roots_test`)**.**

```
  fixes a b::real and p::"real poly"
  assumes "p ≠ 0" and "a < b"
  shows "proots_count p {x. a < x ∧ x < b}
                   ≤ descartes_roots_test a b p ∧
          even (descartes_roots_test a b p
                 - proots_count p {x. a < x ∧ x < b})"
```

which claims that $\mathrm{Var}(P_I)$ always exceeds $\mathrm{Num}_{\mathbb{R}}(P; (a, b))$ by an even number.

Proof of Theorem 3.11. Lemma 3.10 yields

$$\mathrm{Num}_{\mathbb{R}}(P; (a, b)) = \mathrm{Num}_{\mathbb{R}}(P_I; (0, +\infty)),$$

with which we apply Theorem 3.8 to conclude the proof. □

As an approximation, it is natural to ask when the Descartes roots test ($\mathrm{Var}(P_I)$) is exact. From Theorem 3.11, it is easy to see that it would be exact as least when $\mathrm{Var}(P_I) = 0$ and $\mathrm{Var}(P_I) = 1$. Also, it is analogous to Corollary 3.7 that $\mathrm{Var}(P_I)$ is exact when all roots are real:

Corollary 3.12 (`descartes_roots_test_real`)**.**

```
  fixes a b::real and p::"real poly"
  assumes "all_roots_real p" and "a < b"
  shows "proots_count p {x. a < x ∧ x < b}
                         = descartes_roots_test a b p"
```

3.5 Remarks

Ever since the seminal paper by Collins and Akritas [8], the Descartes roots test has been closely linked to modern real root isolation [13, 14, 28], where an effective method is needed for testing if an interval has zero or exactly one root. Although Sturm's theorem (which has already been formalised in Isabelle [11, 15, 19]) is also up to the task of root testing, it is considered too slow in modern computer algebra. Our mechanised version of the Descartes roots test is, by no means, state of the art; it is probably the most straightforward and naive implementation. Improvements over our current implementation are mainly about avoiding exact arithmetic, and the approaches include partial Taylor shift [14] and bitstream arithmetic [13, Chapter 3].

4 Extending Sturm's Theorem to Exactly Count Multiple Roots

With the Descartes roots test we obtained from the previous section, we have an effective method to over-approximate the number of roots (with multiplicity) within an interval. However, we may sometimes want to know the exact number, as we will describe below (§5). For now, we only have the classical Sturm theorem available (in Isabelle/HOL), which only counts *distinct* real roots. In this section, we extend our previous formalisation of Sturm's theorem so that we will be able to count roots *with multiplicity* and exactly.

Our mechanised proof follows Rahman and Schmeisser [27, Theorem 10.5.6].

Theorem 4.1 (Sturm's theorem)**.** *Let $P \in \mathbb{R}[x]$, $a, b \in \overline{\mathbb{R}}$ such that $a < b$, $P(a) \neq 0$, and $P(b) \neq 0$. Sturm's theorem claims*

$$\mathrm{NumD}_{\mathbb{R}}(P; (a, b)) = \mathrm{Var}(\mathrm{SRemS}(P, P'); a, b)$$

where

- *$\mathrm{NumD}_{\mathbb{R}}(P; (a, b))$ is the number of distinct roots of the polynomial P within the interval (a, b),*
- *P' is the first derivative of P,*
- *Var is as in Definition 2.1,*
- *$\mathrm{SRemS}(P, P')$ is the signed remainder sequence:*
$$[P_1, P_2, \ldots, P_n], \tag{29}$$
such that $P_1 = P$, $P_2 = P'$, $P_i = -(P_{i-1} \bmod P_{i-2})$ $(3 \leq i \leq n)$, and $P_n \bmod P_{n-1} = 0$.

The core idea of our extended Sturm's theorem is to extend the remainder sequence (SRemS):

Definition 4.2 (Extended signed remainder sequence)**.** Let $P, Q \in \mathbb{R}[x]$. The extended signed remainder sequence

$$\mathrm{SRemSE}(P, Q) = [P_1, P_2, \ldots, P_m]$$

is defined as $P_1 = P$, $P_2 = Q$, and for $i \geq 3$:

$$P_i = \begin{cases} -(P_{i-1} \bmod P_{i-2}), & \text{if } P_{i-1} \bmod P_{i-2} \neq 0 \\ P'_{i-1}, & \text{otherwise,} \end{cases} \tag{30}$$

until P_m such that $P_{m+1} = 0$ by (30).

In Isabelle/HOL, SRemS and SRemSE are respectively mechanised as `smods` and `smods_ext`:

```
function  smods::
    "real poly ⇒ real poly ⇒ real poly list"
  where
    "smods p q = (if p = 0 then []
                  else p # (smods q (- (p mod q)))
                  )"
function  smods_ext::
    "real poly ⇒ real poly ⇒ real poly list"
  where
    "smods_ext p q =
          (if p = 0 then
              []
           else if p mod q ≠ 0 then
              p # (smods_ext q (- (p mod q)))
           else
              p # (smods_ext q (pderiv q))
          )"
```

where `[]` is an empty list and `#` is the Cons operation on lists—adding one element at the start of a list.

As SRemSE extends SRemS (from the back), it is natural to consider SRemS as a prefix of SRemSE:

Lemma 4.3 (smods_ext_prefix).

```
  fixes p q::"real poly"
  defines "r ≡ last (smods p q)"
  assumes "p ≠ 0" and "q ≠ 0"
  shows "smods_ext p q = smods p q
              @ tl (smods_ext r (pderiv r))"
```

where

- last gives the last element of a list,
- @ concatenates two lists,
- tl removes the head of a list,
- pderiv returns the first derivative of a polynomial.

Moreover, we may need to realise that the last element of SRemS(P, Q) is actually the greatest common divisor (gcd) of P and Q up to some scalar:

Lemma 4.4 (last_smods_gcd).

```
  fixes p q::"real poly"
  defines "r ≡ last (smods p q)"
  assumes "p ≠ 0"
  shows "r = smult (lead_coeff r) (gcd p q)"
```

where

- smult multiplies a polynomial with a scalar,
- lead_coeff gives the lead coefficient of a polynomial.

Finally, we can state (the bounded version of) our extended Sturm's theorem:

Theorem 4.5 (sturm_ext_interval).

```
  fixes a b::real and p::"real poly"
  assumes "a < b" and "poly p a ≠ 0"
    and "poly p b ≠ 0"
  shows "proots_count p {x. a < x ∧ x < b}
          = changes_itv_smods_ext a b p (pderiv p)"
```

where changes_itv_smods_ext a b p (pderiv p) encodes $\mathrm{Var}(\mathrm{SRemSE}(P, P'); a, b)$. Essentially, Theorem 4.5 claims that under some conditions

$$\mathrm{Num}_{\mathbb{R}}(P; (a, b)) = \mathrm{Var}(\mathrm{SRemSE}(P, P'); a, b).$$

Proof of Theorem 4.5. By induction on the length of SRemSE(P, P'), and case analysis on whether $P' = 0$. When $P' = 0$, the proof is trivial since both $\mathrm{Num}_{\mathbb{R}}(P; (a, b)) = 0$ and $\mathrm{Var}(\mathrm{SRemSE}(P, P'); a, b) = 0$ provided $P \neq 0$.

When $P' \neq 0$, we let R be the last element of SRemS(P, P'), and Lemma 4.4 gives us

$$R = \mathrm{lc}(R) \gcd(P, P'), \tag{31}$$

where $\mathrm{lc}(R)$ is the leading coefficient of R.

The essential part of the proof is to relate $\mathrm{Num}(P; (a, b))$ and $\mathrm{Num}(R; (a, b))$:

$$\begin{aligned}
&\mathrm{Num}_{\mathbb{R}}(P; (a, b))\\
&= \sum_{x:P(x)=0 \wedge x \in (a,b)} \mu(x, P) &(32)\\
&= \sum_{x:P(x)=0 \wedge x \in (a,b)} (1 + \mu(x, R)) &(33)\\
&= \mathrm{NumD}_{\mathbb{R}}(P; (a, b)) + \sum_{x:P(x)=0 \wedge x \in (a,b)} \mu(x, R) &(34)\\
&= \mathrm{NumD}_{\mathbb{R}}(P; (a, b)) + \sum_{x:R(x)=0 \wedge x \in (a,b)} \mu(x, R) &(35)\\
&= \mathrm{NumD}_{\mathbb{R}}(P; (a, b)) + \mathrm{Num}_{\mathbb{R}}(R; (a, b)). &(36)
\end{aligned}$$

In particular, (33) has been derived by

$$\begin{aligned}
\mu(x, P) = 1 + \mu(x, P') &= 1 + \min(\mu(x, P'), \mu(x, P))\\
&= 1 + \mu(x, \gcd(P, P')) = 1 + \mu(x, R),
\end{aligned}$$

provided $P(x) = 0$ and (31). Also, (35) is because $\{x \mid R(x) = 0\} \subseteq \{x \mid P(x) = 0\}$ and $\mu(y, R) = 0$ for all $y \in (a, b)$ such that $P(y) = 0$ and $R(y) \neq 0$. With (32) - (36), we have

$$\mathrm{Num}_{\mathbb{R}}(P; (a, b)) = \mathrm{NumD}_{\mathbb{R}}(P; (a, b)) + \mathrm{Num}_{\mathbb{R}}(R; (a, b)). \tag{37}$$

Moreover, the induction hypothesis yields

$$\mathrm{Num}_{\mathbb{R}}(R; (a, b)) = \mathrm{Var}(\mathrm{SRemSE}(R, R'); a, b), \tag{38}$$

and the classical Sturm theorem (Theorem 4.1) yields

$$\mathrm{NumD}_{\mathbb{R}}(P; (a, b)) = \mathrm{Var}(\mathrm{SRemS}(P, P'); a, b). \tag{39}$$

Also, by joining Lemma 4.3 and definition of Var, we may have

$$\begin{aligned}
\mathrm{Var}(\mathrm{SRemSE}(P, P'); a, b) &= \mathrm{Var}(\mathrm{SRemS}(P, P'); a, b)\\
&\quad + \mathrm{Var}(\mathrm{SRemSE}(R, R'); a, b). &(40)
\end{aligned}$$

Finally, putting together (37), (38), (39), and (40) yields

$$\mathrm{Num}_{\mathbb{R}}(P; (a, b)) = \mathrm{Var}(\mathrm{SRemSE}(P, P'); a, b),$$

concluding the proof. □

Be aware that Lemma 4.5 only corresponds to the bounded version of the extended Sturm's theorem. Our formal development also contains unbounded versions (i.e., when $a = -\infty$ or $b = +\infty$).

5 Applications to Counting Complex Roots

In the previous sections (§3 and §4), we have demonstrated our enhancements for counting real roots in Isabelle/HOL. In this section, we will further apply those enhancements to improve existing complex-root-counting procedures [21].

In particular, we will first review the idea of counting complex roots through Cauchy indices in §5.1. After that, we will apply the extended Sturm's theorem (§4) to remove the constraint of forbidding roots on the border when counting complex roots in the upper half-plane (§5.2). In §5.3, we will combine the improved counting procedure (for roots in the upper half-plane) and the base transformation in §3.3 to build a verified procedure to count complex roots within a ball. Finally, we give some remarks about counting complex roots (§5.4).

5.1 Number of Complex Roots and the Cauchy Index

In this section we will briefly review the idea of counting complex roots through Cauchy indices, For a more detailed explanation, the reader can refer to our previous work [21].

Thanks to the argument principle, the number of complex roots can be counted by evaluating a contour integral:

$$\frac{1}{2\pi i}\oint_{\gamma}\frac{P'(x)}{P(x)}dx = N, \tag{41}$$

where $P \in \mathbb{C}[x]$, $P'(x)$ is the first derivative of P and N is the number of complex roots of p (counting multiplicity) inside the loop γ. Also, by the definition of winding numbers, we have

$$n(P\circ\gamma, 0) = \frac{1}{2\pi i}\oint_{\gamma}\frac{P'(x)}{P(x)}dx, \tag{42}$$

where $\circ$ is function composition and $n(P\circ\gamma, 0)$ is the winding number of the path $P\circ\gamma$ around 0. Combining (41) and (42) enables us to count (complex) roots by evaluating a winding number:

$$N = n(P\circ\gamma, 0). \tag{43}$$

Now the question becomes how to evaluate the winding number $n(P\circ\gamma, 0)$. One of the solutions is to utilise the Cauchy index.

To define the Cauchy index, we need to first introduce the concept of *jumps*:

Definition 5.1 (Jump). For $f : \mathbb{R} \to \mathbb{R}$ and $x \in \mathbb{R}$, we define

$$\mathrm{jump}_+(f,x) = \begin{cases} \frac{1}{2} & \text{if } \lim_{u\to x^+} f(u) = +\infty,\\ -\frac{1}{2} & \text{if } \lim_{u\to x^+} f(u) = -\infty,\\ 0 & \text{otherwise.}\end{cases}$$

$$\mathrm{jump}_-(f,x) = \begin{cases} \frac{1}{2} & \text{if } \lim_{u\to x^-} f(u) = +\infty,\\ -\frac{1}{2} & \text{if } \lim_{u\to x^-} f(u) = -\infty,\\ 0 & \text{otherwise.}\end{cases}$$

We can now proceed to define Cauchy indices by summing up these jumps over an interval and along a path.

Definition 5.2 (Cauchy index). For $f : \mathbb{R} \to \mathbb{R}$ and $a, b \in \overline{\mathbb{R}}$, the Cauchy index of f over the interval $[a, b]$ is defined as

$$\mathrm{Ind}_a^b(f) = \sum_{x\in[a,b)} \mathrm{jump}_+(f,x) - \sum_{x\in(a,b]} \mathrm{jump}_-(f,x).$$

Definition 5.3 (Cauchy index along a path). Given a path $\gamma : [0, 1] \to \mathbb{C}$ and a point $z_0 \in \mathbb{C}$, the Cauchy index along γ about z_0 is defined as

$$\mathrm{Indp}(\gamma, z_0) = \mathrm{Ind}_0^1(f),$$

where

$$f(t) = \frac{\mathrm{Im}(\gamma(t) - z_0)}{\mathrm{Re}(\gamma(t) - z_0).}$$

As the Cauchy index $\mathrm{Indp}(\gamma, z_0)$ captures the way that γ crosses the line $\{z \mid \mathrm{Re}(z) = \mathrm{Re}(z_0)\}$, we can evaluate the winding number through the Cauchy index:

Theorem 5.4. *Given a valid path $\gamma : [0, 1] \to \mathbb{C}$ and a point $z_0 \in \mathbb{C}$, such that γ is a loop and z_0 is not on the image of γ, we have*

$$n(\gamma, z_0) = -\frac{\mathrm{Indp}(\gamma, z_0)}{2}.$$

Combining Theorem 5.4 and (43) gives us a way to count complex polynomial roots:

$$N = -\frac{\mathrm{Indp}(P\circ\gamma, z_0)}{2}. \tag{44}$$

What is more interesting is that $\mathrm{Indp}(P\circ\gamma, z_0)$ (or $\mathrm{Ind}_a^b(f, z_0)$) can be calculated through remainder sequences and sign variations when $P\circ\gamma$ (or f) is a rational function. That is, the right-hand side of (44) becomes executable, and we have a procedure to count N.

5.2 Resolving the Root-on-the-Border Issue when Counting Roots within a Half-Plane

Fundamentally, the complex-root-counting procedure in the previous section relies on the winding number and the argument principle, both of which disallow roots of P on the border γ. As a result, both mechanised procedures — counting roots within a rectangle and within a half-plane — in our previous work will fail whenever there is a root on the border. In this section, we will utilise our newly mechanised extended Sturm's theorem to resolve the root-on-the-border issue when counting roots within a half-plane. Note that the root-on-the-border issue for the rectangular case, unfortunately, remains: we leave this issue for future work.

Considering that any half-plane can be transformed into the upper half-plane through a linear-transformation, we only need to focus on the upper-half-plane case:

```
definition  proots_upper :: "complex poly ⇒ nat"
  where
 "proots_upper p = proots_count p {z. Im z > 0}"
```

where `proots_upper p` encodes $\mathrm{Num}_{\mathbb{C}}(P; \{z \mid \mathrm{Im}(z) > 0\})$ — the number of complex roots of P within the upper half-plane $\{z \mid \mathrm{Im}(z) > 0\}$.

Previously, we relied on the following lemma to count $\mathrm{Num}_{\mathbb{C}}(P; \{z \mid \mathrm{Im}(z) > 0\})$:

Lemma 5.5 (`proots_upper_cindex_eq`).

```
  fixes p::"complex poly"
  assumes "lead_coeff p = 1"
      and no_real_roots: "∀x∈proots p. Im x ≠ 0"
    shows "proots_upper p = (degree p -
              cindex_poly_ubd (map_poly Im p)
                              (map_poly Re p)) / 2"
```

where

- lead_coeff p = 1 asserts the polynomial P to be monic,
- the assumption no_real_roots asserts that P does not have any root on the real axis (i.e., the border). This assumption is, as mentioned earlier, because the argument principle disallows roots on the border.
- cindex_poly_ubd (map_poly Im p) (map_poly Re p) encodes the Cauchy index

  $$\mathrm{Ind}_{-\infty}^{+\infty}\left(\lambda x.\frac{\mathrm{Im}(P(x))}{\mathrm{Re}(P(x))}\right),$$

  which can be computed (through remainder sequences and sign variations) due to $\lambda x.\,\mathrm{Im}(P(x))/\,\mathrm{Re}(P(x))$ being a rational function.

To solve the root-on-the-border issue in Lemma 5.5, we observe the effect of removing roots from a horizontal border:

Lemma 5.6 (cindexE_roots_on_horizontal_border).

```
  fixes p q r ::"complex poly" and s_t::complex
    and a b s::real
  defines "γ≡linepath s_t (s_t + of_real s)"
  assumes "p = q * r" and "lead_coeff r = 1"
      and "∀x∈proots r. Im x = Im s_t"
    shows "cindexE a b (λt. Im ((poly p ∘ γ) t)
                      / Re ((poly p ∘ γ) t)) =
          cindexE a b (λt. Im ((poly q ∘ γ) t)
                      / Re ((poly q ∘ γ) t))"
```

where the polynomial Q is the result of P after removing some roots on the horizontal border γ. Lemma 5.6 claims that

$$\mathrm{Ind}_b^a\left(\lambda x.\frac{\mathrm{Im}(P(\gamma(x)))}{\mathrm{Re}(P(\gamma(x)))}\right)=\mathrm{Ind}_b^a\left(\lambda x.\frac{\mathrm{Im}(Q(\gamma(x)))}{\mathrm{Re}(Q(\gamma(x)))}\right).$$

That is, the Cauchy index will remain the same if we only drop roots on a horizontal border.

We can now refine Lemma 5.5 by dropping the no_real_roots assumption:

Lemma 5.7 (proots_upper_cindex_eq').

```
  fixes p::"complex poly"
  assumes "lead_coeff p = 1"
  shows "proots_upper p =
            (degree p - proots_count p {x. Im x=0}
            - cindex_poly_ubd (map_poly Im p)
                              (map_poly Re p)) /2"
```

To compare Lemma 5.7 with Lemma 5.5, we may note there is an extra term proots_count p {x. Im x=0} in the conclusion. This term encodes $\mathrm{Num}_{\mathbb{C}}(P;\{z \mid \mathrm{Im}(z)=0\})$, and we can have

$$\begin{aligned}\mathrm{Num}_{\mathbb{C}}&(P;\{z \mid \mathrm{Im}(z)=0\})\\ &=\mathrm{Num}_{\mathbb{R}}(\gcd(\mathrm{Re}(P),\mathrm{Im}(P));(-\infty,+\infty)),\end{aligned}\tag{45}$$

where $\mathrm{Re}(P),\mathrm{Im}(P)\in\mathbb{R}[x]$ are respectively the real and complex part of a complex polynomial P such that $P(x)=\mathrm{Re}(P)(x)+i\,\mathrm{Im}(P)(x)$. The rationale behind (45) is that each root of P on the real axis ($\{z \mid \mathrm{Im}(z)=0\}$) is actually real and is also a root of both $\mathrm{Re}(P)$ and $\mathrm{Im}(P)$. More importantly, the right-hand side of (45) is where we will apply our extended Sturm's theorem in §4.

Proof of Lemma 5.7. Let Q be the polynomial P after removing all the roots on the border (i.e., the real axis) such that

$$\mathrm{Im}(z)\neq 0 \quad \text{whenever } z \text{ is a complex root of } Q. \tag{46}$$

By the definition of Q and (46), we can apply Lemma 5.5 to derive

$$\begin{aligned}\mathrm{Num}_{\mathbb{C}}&(P;\{z \mid \mathrm{Im}(z)>0\})\\ &=\mathrm{Num}_{\mathbb{C}}(Q;\{z \mid \mathrm{Im}(z)>0\})\\ &=\deg(Q)-\mathrm{Ind}_{-\infty}^{+\infty}\left(\lambda x.\frac{\mathrm{Im}(Q(x))}{\mathrm{Re}(Q(x))}\right).\end{aligned}\tag{47}$$

Moreover, $\deg(P)$ and $\deg(Q)$ are related by the fundamental theorem of algebra:

$$\deg(Q)=\deg(P)-\mathrm{Num}_{\mathbb{C}}(P;\{z \mid \mathrm{Im}(z)=0\}),\tag{48}$$

and Lemma 5.6 brings us the equivalence between two Cauchy indices:

$$\mathrm{Ind}_{-\infty}^{+\infty}\left(\lambda x.\frac{\mathrm{Im}(Q(x))}{\mathrm{Re}(Q(x))}\right)=\mathrm{Ind}_{-\infty}^{+\infty}\left(\lambda x.\frac{\mathrm{Im}(P(x))}{\mathrm{Re}(P(x))}\right).\tag{49}$$

Putting (47), (48), and (49) together yields

$$\begin{aligned}\mathrm{Num}_{\mathbb{C}}&(P;\{z \mid \mathrm{Im}(z)>0\})\\ &=\deg(P)-\mathrm{Num}_{\mathbb{C}}(P;\{z \mid \mathrm{Im}(z)=0\})\\ &\qquad-\mathrm{Ind}_{-\infty}^{+\infty}\left(\lambda x.\frac{\mathrm{Im}(P(x))}{\mathrm{Re}(P(x))}\right),\end{aligned}\tag{50}$$

which concludes the proof. □

Finally, we can have a *code equation* (i.e., executable procedure) from the refined Lemma 5.7 to compute $\mathrm{Num}_{\mathbb{C}}(P;\{z \mid \mathrm{Im}(z)>0\})$:

Lemma 5.8 (proots_upper_code1[code]).

```
  "proots_upper p =
    (if p ≠ 0 then
       (let p_m = smult (inverse (lead_coeff p)) p;
            p_I = map_poly Im p_m;
            p_R = map_poly Re p_m;
            g = gcd p_I p_R
        in
            nat ((degree p
                    - changes_R_smods_ext g (pderiv g)
                    - changes_R_smods p_R p_I) div 2
               )
```

```
      )
    else
     Code.abort (STR ''proots_upper fails when p=0.'')
                                          (λ_. proots_upper p))"
```

where

- p_m is a monic polynomial produced by P divided by its leading coefficient, and this monic polynomial is required by the assumption `lead_coeff p = 1` in Lemma 5.7.
- `changes_R_smods_ext g (pderiv g)` encodes

$$\mathrm{Var}(\mathrm{SRemSE}(G, G'); -\infty, +\infty)$$
$$\text{where } G = \gcd(\mathrm{Re}(P), \mathrm{Im}(P)),$$

  which computes $\mathrm{Num}_{\mathbb{C}}(P; \{z \mid \mathrm{Im}(z) = 0\})$ in Lemma 5.7. Note that this part is where our extended Sturm's theorem in §4 has been utilised.
- `changes_R_smods` p_R p_I stands for

$$\mathrm{Var}(\mathrm{SRemS}(P_R, P_I); -\infty, +\infty),$$

  which computes $\mathrm{Ind}_{-\infty}^{+\infty}\left(\lambda x. \frac{\mathrm{Im}(P(x))}{\mathrm{Re}(P(x))}\right)$ in Lemma 5.7.
- The command `Code.abort` raises an exception in the case of $P = 0$.

Overall, Lemma 5.8 asserts that

$$\mathrm{Num}_{\mathbb{C}}(P; \{z \mid \mathrm{Im}(z) > 0\})$$

is equivalent to an executable expression: the right-hand side of Lemma 5.8. Because it is declared as a code equation, it implements a verified procedure to compute $\mathrm{Num}_{\mathbb{C}}(P; \{z \mid \mathrm{Im}(z) > 0\})$. We can now type the following command in Isabelle/HOL:

value `"proots_upper [:1+i, -2-i, 1:]"`

to compute $\mathrm{Num}_{\mathbb{C}}((1 + i) + (-2 - i)x + x^2; \{z \mid \mathrm{Im}(z) > 0\})$, which was not possible in our previous work [21] since the polynomial $(1 + i) + (-2 - i)x + x^2 = (x - 1)(x - 1 - i)$ has a root on the border (i.e., the real axis).

5.3 Counting Roots within a Ball

In this section, we will introduce a verified procedure to count $\mathrm{Num}_{\mathbb{C}}(P; \{z \mid |z - z_0| < r\})$, the number of complex roots of a polynomial P within the ball $\{z \mid |z - z_0| < r\}$. The core idea is to use the base transformation in §3.3 to convert the current case to the one of counting roots within the upper half-plane, and then make use of the procedure in §5.2 to finish counting.

Let `proots_ball` denote $\mathrm{Num}_{\mathbb{C}}(P; \{z \mid |z - z_0| < r\})$:

```
definition  proots_ball::
    "complex poly ⇒ complex ⇒ real ⇒ nat"
  where
    "proots_ball p z₀ r = proots_count p (ball z₀ r)"
```

With the transformation operation (`fcompose`) we developed in §3.3, we can derive the following equivalence relation in the number of roots:

Lemma 5.9 (`proots_ball_plane_eq`).
```
  fixes p::"complex poly"
  assumes "p ≠ 0"
  shows "proots_count p (ball 0 1)
            = proots_count (fcompose p [:i,-1:] [:i,1:])
                                              {z. 0 < Im z}"
```

That is,

$$\mathrm{Num}_{\mathbb{C}}(P; \{z \mid |z| < 1\}) = \mathrm{Num}_{\mathbb{C}}\left((i + x)^n P\left(\frac{i - x}{i + x}\right); \{z \mid \mathrm{Im}(z) > 0\}\right), \quad (51)$$

where n is the degree of P.

Moreover, we can relate roots between different balls using normal polynomial composition:

Lemma 5.10 (`proots_uball_eq`).
```
  fixes p::"complex poly" and z₀::complex and r::real
  assumes "p ≠ 0" and "r > 0"
  shows "proots_count p (ball z₀ r)
            = proots_count (p ∘ₚ [:z₀, of_real r:])
                                              (ball 0 1)"
```

where $\circ_p$ encodes the composition operation between two polynomials. Overall, Lemma 5.10 claims

$$\mathrm{Num}_{\mathbb{C}}(P; \{z \mid |z - z_0| < r\}) = \mathrm{Num}_{\mathbb{C}}\left(P(rx - z_0); \{z \mid |z| < 1\}\right). \quad (52)$$

Finally, we can derive a code equation for `proots_ball`:

Lemma 5.11 (`proots_ball_code1[code]`).
```
  "proots_ball p z₀ r =
      ( if r ≤ 0 then
          0
        else if p ≠ 0 then
          proots_upper (fcompose
              (p ∘ₚ [:z₀, of_real r:]) [:i,-1:] [:i,1:])
        else
          Code.abort (STR ''proots_ball fails
                  when p=0.'') (λ_. proots_ball p z₀ r)
      )"
```

The idea behind of Lemma 5.11 is to combine (51) and (52):

$$\mathrm{Num}_{\mathbb{C}}(P; \{z \mid |z - z_0| < r\}) = \mathrm{Num}_{\mathbb{C}}\left((rx + i - z_0)^n P\left(\frac{-rx + i + z_0}{rx + i - z_0}\right); \{z \mid \mathrm{Im}(z) > 0\}\right), \quad (53)$$

so that we can apply Lemma 5.8 to count roots within the upper half-plane instead.

Because Lemma 5.11 is declared as a code equation, we can execute it. For example, we can now type the following command in Isabelle/HOL:

value `"proots_ball [:i,- 1 - i, 1:] 0 1"`

to check that the polynomial $i + (-1 - i)x + x^2$ has no roots within the ball $\{z \mid |z| < 1\}$.

5.4 Remarks

Generally, most complex-root-counting procedures boil down to applying the argument principle and approximating some winding number. The Cauchy index on the complex plane elegantly approximates the winding number, and it can be effectively computed by remainder sequences and sign variations as in the application of Sturm's theorem. As this approach is moderately efficient, in 1978 Wilf [32] used this counting mechanism for his (complex) root isolation algorithm.

However, as we mentioned earlier in §3.5, remainder sequences are generally considered too slow for modern computer algebra systems. As a result, in 1992 Collins and Krandick [10] proposed an approach to directly approximate the winding number through efficient real root isolation based on Descartes' rule of signs, and this approach is actually the one that has been widely implemented in modern systems like Mathematica and SymPy.

In the future, we hope to implement Collins and Krandick's approach. Luckily, the Descartes roots test we mechanised in §3.4 should serve as the first step.

6 Related Work

Counting *distinct* real roots with Sturm's theorem has been widely implemented among major proof assistants including PVS [25], Coq [22], HOL Light [24] and Isabelle [11, 15, 18]. In contrast, our previous complex-root-counting procedure [21] seems to be the only one that counts complex roots, since counting complex roots usually requires a formal proof of the argument principle in complex analysis, which (to the best of our knowledge) is only available in Isabelle/HOL [20].

According to the website of *Formalizing 100 Theorems*[2], Descartes' rule of signs has been independently formalised in Isabelle/HOL [12], HOL Light, and ProofPower, and all three versions seem to follow an informal inductive proof by Arthan [4]. In Coq, Bertot et al. [6] have investigated real root isolation through Bernstein coefficients, and during the investigation they have proved a corollary of Descartes' rule of sign: the polynomial has exactly one positive root if there is only one sign change in its coefficient sequence. In comparison, we have formalised a more general result (i.e., the Budan-Fourier theorem), and derive Descartes' rule of signs as an almost trivial consequence. As a benefit of this more general result, we can additionally derive the corollary that the roots approximation through both Descartes' rule of signs and the Descartes roots test will be exact when all roots are real; it is not clear how to deduce this without the Budan-Fourier theorem.

Since Thiemann and Yamada have formalised Yun's algorithm in Isabelle/HOL [30, 31], there could be an alternative procedure to count multiple real roots exactly. Given $P \in \mathbb{R}[x]$, with Yun's algorithm we can have a square-free factorisation of P:

$$P = Q_1 Q_2^2 Q_3^3 \cdots Q_n^n,$$

such that polynomials from $\{Q_i\}$ $(1 \leq i \leq n)$ are pairwise coprime and square-free. We can then obtain a procedure to count multiple roots by applying Sturm's theorem to each Q_i, multiplying the result by i, and summing them together. We believe our extended Sturm's theorem will be more efficient than the sketch above, but that is never for sure until we perform a side-by-side comparison.

Potential applications of our work include various formalisations of algebraic numbers in Coq [7] and Isabelle/HOL [19, 29]. A real algebraic number is usually encoded as a polynomial P and an isolation interval, and this interval is frequently tested (or refined) to guarantee that exactly one root of P lies within it. At present, the testing and refining process relies on Sturm's theorem, which can be replaced by our new Descartes roots test for better efficiency. Furthermore, when encoding complex algebraic numbers, we may need to deal with an isolation box or ball in the complex plane, where our complex-root-counting procedures should be of help.

7 Experiments

In this section, we briefly benchmark our root-counting procedures over some randomly generated polynomials, to convey an idea about their scalability. All the experiments are run on a Intel Core i7 CPU (quad core @ 2.66 GHz) and 16 gigabytes RAM. When benchmarking verified operations, the expression to evaluate is first defined in Isabelle/HOL, and then extracted and evaluated in Poly/ML. The reason for this is that when invoking **value** in Isabelle/HOL to evaluate an expression, a significant and unpredictable amount of time is spent generating code, so we evaluate an extracted expression to obtain more precise results.

First, we compare using the classical Sturm theorem, the extended Sturm theorem, the Budan-Fourier theorem, and the Descartes roots test to count/approximate the number of real roots of various polynomials over the interval $(0, 1)$ or $(0, 1]$. As illustrated in Table 1 and Figure 1, procedures based on remainder sequences (i.e., Sturm and Ex_Sturm) are much slower than the others, and their performance degrades rapidly as the bit size of the coefficients grows. In the meantime, the difference in performance between the Budan-Fourier theorem and the Descartes roots test is, surprisingly, marginal. We believe this is due to our naive implementation of Taylor shift, which usually contributes most to the running time of the roots test.

In addition, we also apply our complex-root-counting procedures to count roots within the upper half-plane and the ball $\{z \mid |z| < 1\}$. The result is illustrated in Table 2: both methods have shown moderate performance, but due to their method of computing remainder sequences the performance deteriorates quickly as the coefficient bit-size increases.

[2] http://www.cs.ru.nl/~freek/100/index.html

$$\begin{aligned}
\mathrm{P}_1(\mathrm{x}) = \quad & -\frac{85}{68}+\frac{70}{5}x+\frac{88}{79}x^2+\frac{29}{75}x^3+\frac{80}{51}x^4-\frac{66}{52}x^5+\frac{9}{71}x^6-\frac{14}{61}x^7-\frac{27}{64}x^8-\frac{100}{83}x^9+\frac{1}{53}x^{10}-\frac{23}{85}x^{11}+\frac{83}{98}x^{12}+\frac{48}{16}x^{13}\\
& -\frac{89}{25}x^{14}-\frac{100}{5}x^{15}+\frac{36}{28}x^{16}+\frac{1}{1}x^{17}+\frac{43}{99}x^{18}-\frac{29}{32}x^{19}+\frac{74}{97}x^{20}+\frac{9}{5}x^{21}+\frac{20}{70}x^{22}-\frac{89}{27}x^{23}-\frac{33}{48}x^{24}+\frac{16}{33}x^{25}+\frac{84}{63}x^{26}\\
& +\frac{96}{89}x^{27}+\frac{22}{69}x^{28}+\frac{95}{97}x^{29}\\
\mathrm{P}_2(\mathrm{x}) = \quad & -34-28x+5x^2-39x^3+83x^4-89x^5-49x^6+94x^7-66x^8+18x^9+75x^{10}+84x^{11}-98x^{12}-68x^{13}+12x^{14}\\
& +46x^{15}-43x^{16}+98x^{17}+24x^{18}-30x^{19}+10x^{20}-88x^{21}+54x^{22}+79x^{23}-29x^{24}+12x^{25}-55x^{26}-46x^{27}\\
& -18x^{28}+50x^{29}\\
\mathrm{P}_3(\mathrm{x}) = \quad & \frac{9}{30}-\frac{65}{82}x-\frac{94}{68}x^2+\frac{9}{33}x^3-\frac{56}{83}x^4-\frac{22}{35}x^5+\frac{73}{31}x^6+\frac{69}{2}x^7-\frac{58}{43}x^8+\frac{71}{22}x^9-\frac{75}{44}x^{10}+\frac{2}{49}x^{11}+\frac{24}{40}x^{12}+\frac{33}{62}x^{13}\\
& -\frac{17}{2}x^{14}-\frac{39}{82}x^{15}-\frac{55}{43}x^{16}-\frac{26}{47}x^{17}+\frac{46}{4}x^{18}-\frac{48}{26}x^{19}+\frac{35}{83}x^{20}-\frac{50}{100}x^{21}-\frac{60}{65}x^{22}+\frac{66}{36}x^{23}-\frac{43}{76}x^{24}+\frac{30}{24}x^{25}\\
& +\frac{18}{28}x^{26}-\frac{96}{51}x^{27}+\frac{49}{42}x^{28}-\frac{41}{89}x^{29}+\frac{81}{90}x^{30}-\frac{65}{57}x^{31}-\frac{70}{64}x^{32}-\frac{50}{26}x^{33}+\frac{91}{40}x^{34}+\frac{52}{68}x^{35}-\frac{91}{99}x^{36}-\frac{79}{59}x^{37}+\frac{15}{93}x^{38}\\
& -\frac{56}{42}x^{39}-\frac{20}{59}x^{40}+\frac{50}{62}x^{41}-\frac{27}{77}x^{42}+\frac{28}{53}x^{43}-\frac{36}{75}x^{44}\\
\mathrm{P}_4(\mathrm{x}) = \quad & -20-6x-50x^2-95x^3+35x^4-64x^5+77x^6-56x^7+18x^8-94x^9-74x^{10}-69x^{11}-62x^{12}-93x^{13}-4x^{14}\\
& -41x^{15}-47x^{16}-48x^{17}-95x^{18}-41x^{19}+29x^{20}+76x^{21}+70x^{22}-67x^{23}-91x^{24}-93x^{25}-55x^{26}-34x^{27}\\
& -67x^{28}-61x^{29}-8x^{30}+32x^{31}+8x^{32}-33x^{33}-27x^{34}-8x^{35}+88x^{36}+53x^{37}-28x^{38}-66x^{39}-72x^{40}-46x^{41}\\
& +15x^{42}-19x^{43}+29x^{44}\\
\mathrm{P}_5(\mathrm{x}) = \quad & (-\frac{93}{47}-\frac{49}{8}i)+(\frac{187}{47}+\frac{547}{88}i)x+(-\frac{203}{67})+\frac{538}{11}i)x^2+(\frac{2}{67}-\frac{1181}{24})x^3+(\frac{133}{81}-\frac{25}{24})x^4+\\
& (\frac{2111}{5670}+\frac{71}{12})x^5+(\frac{5949}{70}-\frac{64}{15})x^6+(-\frac{305}{3}+\frac{18}{5})x^7+(\frac{4067}{255}-\frac{411}{89})x^8+(-\frac{9}{68}+\frac{2669}{4895})x^9+(-\frac{3}{20}+\frac{4}{55})x^{10}\\
\mathrm{P}_6(\mathrm{x}) = \quad & (51-83i)+(-82+29i)x+(-37-6i)x^2(1+45i)x^3+(145-57i)x^4+(-10+17i)x^5+(-39+22i)x^6\\
& +(40-35i)x^7+(-112-27i)x^8+(106-2i)x^9+(-63+97i)x^{10}
\end{aligned}$$

Figure 1. Some example polynomials.

Table 1. Applying various procedures to count the number of real roots over an interval.

	Time (s)			
Polynomial	Sturm	Ex_Sturm	Fourier	Descartes
P_1	12.123	23.418	.002	.002
P_2	1.612	1.742	.001	0
P_3	322.569	524.975	.007	.007
P_4	8.894	13.425	.003	.003

8 Conclusion

In this paper, we have strengthened the existing root-counting tools in Isabelle. In particular, we have

- formalised a proof of the Budan-Fourier theorem, and thereby implemented the Descartes roots test,
- extended our previous formalisation of the classical Sturm theorem to count real roots with multiplicity,
- applied part of the results above to improve our previous complex-root-counting procedures by allowing roots on the border in the half-plane case and providing a procedure to count roots within a ball.

The proofs described in this paper are about 6000 LOC in total, and took around 6 person-months to complete.

Table 2. Counting the number of complex roots within the upper half-plane and a ball.

	Time (s)	
Polynomial	proots_upper	proots_ball
P_5	4.359	24.509
P_6	0.633	0.256

As counting polynomial roots is a fundamental topic in computer algebra and numerical computing, we believe our verified routines will be of use when certifying continuous systems and for coding tactics.

Acknowledgments

Both Li and Paulson were supported by the ERC Advanced Grant ALEXANDRIA (Project 742178), funded by the European Research Council. We thank Manuel Eberl and René Thiemann for various fruitful discussions and for pointing us to the approach of counting multiple roots through square-free factorisation. We are also grateful for suggestions from anonymous reviewers.

References

[1] Alkiviadis Akritas, Adam Strzeboński, and Panagiotis Vigklas. 2008. On the various bisection methods derived from Vincent's theorem. *Serdica Journal of Computing* 2, 1 (2008), 89–104.

[2] Alkiviadis G Akritas. 2008. There is no Descartes' Method. In *Computer Algebra in Education*, M.J. Wester and M. Beaudin (Eds.). AulonaPress, 19–35. On the Internet at https://faculty.e-ce.uth.gr/akritas/articles/71.pdf.

[3] Vladimir I Arnold. 1992. *Ordinary Differential Equations.* Springer.

[4] R. D. Arthan. 2007. Descartes' Rule of Signs by an Easy Induction. Online at https://arxiv.org/abs/0710.1881.

[5] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. 2006. *Algorithms in Real Algebraic Geometry.* Algorithms and Computation in Mathematics, Vol. 10. Springer.

[6] Yves Bertot, Frédérique Guilhot, and Assia Mahboubi. 2011. A formal study of Bernstein coefficients and polynomials. *Mathematical Structures in Computer Science* 21, 04 (Aug. 2011), 731–761.

[7] Cyril Cohen. 2012. Construction of Real Algebraic Numbers in Coq.. In *4th International Conference on Interactive Theorem Proving, ITP 2013*, Lennart Beringer and Amy Felty (Eds.). Springer, 67–82.

[8] George E Collins and Alkiviadis G Akritas. 1976. Polynomial Real Root Isolation using Descartes' Rule Of Signs. In *SYMSAC '76: ACM Symposium on Symbolic and Algebraic Computation*, Richard D. Jenks (Ed.). 272–275.

[9] George E Collins, Jeremy R Johnson, and Werner Krandick. 2002. Interval Arithmetic in Cylindrical Algebraic Decomposition. *J. Symb. Comput.* 34, 2 (2002), 145–157.

[10] George E Collins and Werner Krandick. 1992. An efficient algorithm for infallible polynomial complex root isolation. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC '90.* ACM, Berkeley, CA, USA, 189–194.

[11] Manuel Eberl. 2015. A Decision Procedure for Univariate Real Polynomials in Isabelle/HOL. In *Conference on Certified Programs and Proofs, CPP 2015.* ACM Press, 75–83.

[12] Manuel Eberl. 2015. Descartes' Rule of Signs. *Archive of Formal Proofs* (Dec. 2015). http://isa-afp.org/entries/Descartes_Sign_Rule.html, Formal proof development.

[13] Arno Eigenwillig. 2008. *Real Root Isolation for Exact and Approximate Polynomials using Descartes' Rule of Signs.* Ph.D. Dissertation. University of Saarland.

[14] Alexander Kobel, Fabrice Rouillier, and Michael Sagraloff. 2016. Computing Real Roots of Real Polynomials ... and now For Real!. In *41st International Symposium on Symbolic and Algebraic Computation, ISSAC '16.* ACM Press, 303–310.

[15] Wenda Li. 2014. The Sturm-Tarski Theorem. *Archive of Formal Proofs* (Sept. 2014).

[16] Wenda Li. 2017. Count the Number of Complex Roots. *Archive of Formal Proofs* (Oct. 2017).

[17] Wenda Li. 2018. The Budan-Fourier Theorem and Counting Real Roots with Multiplicity. *Archive of Formal Proofs* (Sept. 2018).

[18] Wenda Li, Grant Olney Passmore, and Lawrence C Paulson. 2017. Deciding Univariate Polynomial Problems Using Untrusted Certificates in Isabelle/HOL. *Journal of Automated Reasoning* 44, 3 (Aug. 2017), 175–23.

[19] Wenda Li and Lawrence C Paulson. 2016. A modular, efficient formalisation of real algebraic numbers. In *5th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2016*, Jeremy Avigad and Adam Chlipala (Eds.). ACM, 66–75.

[20] Wenda Li and Lawrence C Paulson. 2016. A Formal Proof of Cauchy's Residue Theorem. In *7th International Conference on Interactive Theorem Proving, ITP 2016*, Jasmin Christian Blanchette and Stephan Merz (Eds.). Springer, 235–251.

[21] Wenda Li and Lawrence C Paulson. 2018. Evaluating Winding Numbers and Counting Complex Roots through Cauchy Indices in Isabelle/HOL. *CoRR* abs/1804.03922 (2018).

[22] Assia Mahboubi and Cyril Cohen. 2012. Formal Proofs in Real Algebraic Geometry: from Ordered Fields to Quantifier Elimination. *Logical Methods in Computer Science* 8, 1 (2012).

[23] Morris Marden. 1949. *Geometry of Polynomials. Second Edition.* American Mathematical Society, Providence, Rhode Island.

[24] Sean McLaughlin and John Harrison. 2005. A Proof-Producing Decision Procedure for Real Arithmetic. In *CADE-20: 20th International Conference on Automated Deduction (Lecture Notes in Computer Science)*, Robert Nieuwenhuis (Ed.), Vol. 3632. Springer, 295–314.

[25] Anthony Narkawicz, César A 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.

[26] Lawrence C Paulson. 1994. *Isabelle: A generic theorem prover.* Vol. 828. Springer.

[27] Qazi Ibadur Rahman and Gerhard Schmeisser. 2002. *Analytic Theory of Polynomials.* Oxford University Press.

[28] Fabrice Rouillier and Paul Zimmermann. 2004. Efficient isolation of polynomial's real roots. *J. Comput. Appl. Math.* 162, 1 (2004), 33 – 50. https://doi.org/10.1016/j.cam.2003.08.015 Proceedings of the International Conference on Linear Algebra and Arithmetic 2001.

[29] René Thiemann and Akihisa Yamada. 2016. Algebraic Numbers in Isabelle/HOL. In *7th International Conference on Interactive Theorem Proving, ITP 2016*, Jasmin Christian Blanchette and Stephan Merz (Eds.). Springer, 391–408.

[30] René Thiemann and Akihisa Yamada. 2016. Formalizing Jordan Normal Forms in Isabelle/HOL. In *5th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2016.* ACM Press, 88–99.

[31] René Thiemann and Akihisa Yamada. 2016. Polynomial Factorization. *Archive of Formal Proofs* (Jan. 2016). http://isa-afp.org/entries/Polynomial_Factorization.html, Formal proof development.

[32] Herbert S Wilf. 1978. A Global Bisection Algorithm for Computing the Zeros of Polynomials in the Complex Plane. *Journal of the ACM (JACM)* 25, 3 (July 1978), 415–420.

[33] Chee-Keng Yap and Michael Sagraloff. 2011. A Simple but Exact and Efficient Algorithm for Complex Root Isolation. In *36th International Symposium on Symbolic and Algebraic Computation, ISSAC '11.* ACM Press, 353.