# A Decision Procedure for Univariate Real Polynomials in Isabelle/HOL

Manuel Eberl
Technische Universität München
eberlm@in.tum.de

## Abstract

*Sturm sequences* are a method for computing the number of real roots of a univariate real polynomial inside a given interval efficiently. In this paper, this fact and a number of methods to construct Sturm sequences efficiently have been formalised with the interactive theorem prover Isabelle/HOL. Building upon this, an Isabelle/HOL proof method was then implemented to prove interesting statements about the number of real roots of a univariate real polynomial and related properties such as non-negativity and monotonicity.

***Categories and Subject Descriptors*** G.1.5 [*Numerical Analysis*]: Roots of Nonlinear Equations—Polynomials, methods for; D.2.4 [*Software Engineering*]: Software/Program verification—Correctness proofs; I.1.2 [*Symbolic and Algebraic Manipulation*]: Algorithms—Algebraic algorithms

***General Terms*** Algorithms, Verification

***Keywords*** Sturm sequences; real arithmetic; decision procedure; Isabelle

## 1. Introduction

*Sturm sequences* are finite sequences of univariate polynomials with certain properties that, as shown by *Sturm's Theorem* [13], can be used to determine the number of roots of a univariate real polynomial in a given interval algorithmically. This is used to prove properties about the roots of specific polynomials directly, but can also be part of other algorithms such as approximation of roots through bisection.

There are different ways to construct Sturm sequences for a fixed polynomial $P$. The 'canonical' Sturm sequence construction works only if $P$ has no multiple roots, but it can be adapted to the more general case of any non-zero polynomial.

Both Sturm's Theorem and the correctness of these constructions are formally proven in Isabelle/HOL in this work. In the following, we will give a detailed account of the formalisation and the proof method derived from it, including a very explicit proof that closely follows the structure of the formal proof in Isabelle/HOL, but omits the proofs for basic lemmas about real analysis and polynomials.

The main contribution of this work is the use of reflection to use Sturm's theorem, together with some pre-processing and an interval splitting algorithm, to create a decision procedure for many interesting and non-trivial properties on univariate real polynomials, e.g.

$$|(x :: \text{real}) - 3| < 2 \implies 5 + 6 * x - x^2 > 0\,.$$

The step from Sturm's theorem to the decision procedure is done with Isabelle's Code Generator [4], which can generate executable code for Sturm sequences from the definitions. It must be noted that a very similar approach, albeit in the PVS system, was developed in parallel and independently by Anthony Narkawicz and César Muñoz. [11]

Such a decision procedure has many applications in the context of theorem proving, since non-linear real inequalities are often difficult to prove by hand.

***Outline.*** Section 2 introduces some basic notation for well-known concepts that are not directly related to Sturm sequences. Section 3.1 then defines a number of basic concepts that are used in Sturm's theorem – in particular, the concept of a *Sturm sequence*. Section 3.2 contains some simple auxiliary lemmas that will be required in the main proof. Section 3.3 contains the main portion of the proof: the fact that Sturm sequences can be used to count the real roots of polynomials. Section 4 then shows how Sturm sequences can be constructed in different cases and Sect. 5 explains how these constructions can be used to implement a verified decision procedure for a number of interesting properties of real polynomials. Section 6 demonstrates how this proof method can be used in practice and compares it to other, related Isabelle proof methods. Finally, Sect. 7 summarises the work we did and what the overall result was.

## 2. Notation

In the following proofs, we will use the following notation and terminology:

- Some property holds *in a neighbourhood* of some $x_0 \in \mathbb{R}$ if there exists an $\varepsilon > 0$ such that the property holds for all $x \in (x_0 - \varepsilon; x_0 + \varepsilon) \setminus \{x_0\}$ . Note that it does *not* have to hold *at* $x_0$ *itself*. We will write this as

  $$\text{some property involving } x$$
  $$\text{(for all } x \text{ in a NH of } x_0\text{)}$$

- $\tilde{\mathbb{R}}$ denotes the *extended real numbers* $\mathbb{R} \cup \{\pm\infty\}$ .
- $\mathbb{R}[X]$ denotes the set of all univariate polynomials in the variable $X$ with real coefficients. When speaking about *polynomials*, we will always implicitly mean *univariate real polynomials*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*CPP '15,* January 13–14 2015, Mumbai, India.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3296-5/15/01…$15.00.
http://dx.doi.org/10.1145/2676724.2693166

- $P'$ denotes the formal derivative of a polynomial $P \in \mathbb{R}[X]$ .
- $\mathrm{lc}(P)$ denotes the leading coefficient of a polynomial $P$.

Moreover, the following Isabelle notation will appear in the Isabelle code examples we give:

- Function application is written in Lambda-calculus style, i. e. $f\ x$ stands for $f(x)$ . It associates to the left, i. e. $f\ x\ y$ is $(f\ x)\ y$ .
- Lambda abstractions denote functions, e. g. $\lambda x.\ x+1$ stands for the function $x \mapsto x+1$.
- The syntax $[: a_0, \ldots, a_n :]$ denotes the polynomial $a_n x^n + \ldots + a_0$ .
- The poly function evaluates a polynomial, i. e. poly $p\ x$ is the value of the polynomial $p$ at $x$.
- The card function returns the cardinality of a set (or, by convention, 0 if the set is infinite).

## 3. Proof of Sturm's theorem

*Sturm's Theorem*, which we ultimately want to prove, states, informally: given a sequence $P_0, \ldots, P_n$ of polynomials that fulfil a certain set of properties and real numbers $a$ and $b$, the number of real roots between $a$ and $b$ can be computed by computing the sequences $P_0(a), \ldots, P_n(a)$ and $P_0(b), \ldots, P_n(b)$, counting the number of sign changes in these sequences, and subtracting the latter from the former. [13]

In this section, we will go through the definitions, auxiliary lemmas, and the proof of Sturm's theorem itself. All lemmas given here have been formally proven by us in the course of the formalisation effort as they had not existed in Isabelle's library yet. The definitions and proofs are given in standard mathematical notation in order to make them easier to comprehend for readers not familiar with Isabelle; the Isabelle proofs and definitions, are, however, very similar. Every lemma we give will be labelled with the name of the corresponding Isabelle lemma in `monospaced font`. Unless an explicit theory file is given with the name (e. g. `Misc_Polynomial.some_lemma`), the lemmas reside in `Sturm_Theorem.thy`.

### 3.1 Definitions

First, we define two notions that will be important in the proof of Sturm's theorem.

***Quasi-Sturm sequences.*** We introduce the term *quasi-Sturm sequence* for a list of polynomials $P_0, \ldots, P_n \in \mathbb{R}[X]$ that fulfils the following properties:

- $n \geq 0$ , i. e. the sequence is not empty
- $P_n$ does not change its sign, i. e. for any $x, y \in \mathbb{R}$ , we have $\mathrm{sgn}(P_n(x)) = \mathrm{sgn}(P_n(y))$
- for any $i \in \{0, \ldots, n-2\}$ and any root $x_0$ of $P_{i+1}$, the property $P_i(x_0)P_{i+2}(x_0) < 0$ holds

It can easily be seen that any non-empty suffix of a quasi-Sturm sequence is again a quasi-Sturm sequence. The term 'quasi-Sturm sequence' is, to our knowledge, not used in literature. It was 'invented' solely for the purpose of this formal proof, since we need to perform explicit induction on Sturm sequences, which does not preserve the Sturm sequence property, but does preserve the weaker notion of quasi-Sturm sequence.

***Sturm sequences.*** We call a list of real polynomials a *Sturm sequence* if it is a quasi-Sturm sequence and fulfils the following additional properties:

- $n \geq 1$ , i. e. the sequence contains at least two polynomials
- for any root $x_0$ of $P_0$, the product $P_0(x)P_1(x)$ is negative in some sufficiently small interval $(x_0 - \varepsilon; x_0)$ and positive in some sufficiently small interval $(x_0; x_0 + \varepsilon)$ [1]
- $P_0$ and $P_1$ have no common roots.

This definition was adapted from the book by Michel Coste [2] as it has the virtue of being very general, allowing us to easily generalise the canonical construction later.

In Isabelle, the two concepts of Sturm and quasi-Sturm sequences are captured in predicates of the same name in precisely the same way.

***Sign changes.*** Next, we define the notion of *sign changes*. Let $P_0, \ldots, P_n \in \mathbb{R}[X]$ be a sequence of polynomials and $x \in \mathbb{R}$ . By evaluating the $P_i$ at $x$, we obtain a sequence of real numbers $y_0, \ldots, y_n$ . We now traverse this sequence from left to right and count how often the sign changes, i. e. how often we see a positive number and the previous number in the sequence was negative (or vice versa), skipping zeros. This is called the number of *sign changes* of the sequence $P_0, \ldots, P_n$ at the position $x$ and is denoted as $\sigma(P_0, \ldots, P_n; x)$ .

In Isabelle, this is realised in the `sign_changes` function (see Fig. 1).

```
definition sign_changes :: real poly list ⇒ real ⇒ nat where
  sign_changes ps x =
    length (remdups_adj (filter (λx. x ≠ 0)
      (map (λp. sgn (poly p x)) ps))) − 1
```

**Figure 1.** The Isabelle/HOL definition of the function that counts the sign changes of a polynomial sequence ps at a point x.
The signs of the polynomials are evaluated at $x$ and all zeros deleted from the resulting list. Then, *remdups_adj* is applied to the remaining list, which merges equal adjacent elements (e. g. turning $[1, 1, -1, -1, 1]$ to $[1, -1, 1]$). The length of the resulting list minus one is then the number of sign changes.

A related notion are the sign changes 'at infinity'. For this, we compute the sequences

$$\left(\mathrm{sgn}\left(\lim_{x \to -\infty} P_i(x)\right)\right)_{0 \leq i < n} \quad \text{and} \quad \left(\mathrm{sgn}\left(\lim_{x \to \infty} P_i(x)\right)\right)_{0 \leq i < n}$$

and count the sign changes in these sequences in the same way as before. We write $\sigma(P_0, \ldots, P_n; -\infty)$ and $\sigma(P_0, \ldots, P_n; \infty)$ , respectively. Note that the signs of these limits for a $P_i$ can easily be determined from the sign of the leading coefficient of $P_i$: The sign of $P_i$ at $\infty$ is the sign of the leading coefficient; the sign of $P_i$ at $-\infty$ is the sign of the leading coefficient if $P_i$ has even degree and the opposite if it has odd degree.

### 3.2 Important auxiliary lemmas

The first auxiliary lemma describes the behaviour of polynomials at infinity and will be important for counting roots in intervals of infinite length.

[1] In Isabelle/HOL, this is expressed as 'the property $\mathrm{sgn}(P_0P_1(x)) = (\textbf{if } x > x_0 \textbf{ then } 1 \textbf{ else } -1)$ holds eventually at $x_0$'. Isabelle's analysis library provides *filters* to express such properties concisely.

**Lemma 1** (`Misc_Polynomial.polys_inf_sign_thresholds`)**.** *Let $P \in \mathbb{R}[X] \setminus \{0\}$. Then there exist $l, u \in \mathbb{R}$ such that $l < u$, all roots of $P$ are in $(l;u)$, and*

$$\forall x \leq l.\ \operatorname{sgn}(P(x)) = \operatorname{sgn}\left(\lim_{x\to-\infty} P(x)\right)$$
$$\forall x \geq u.\ \operatorname{sgn}(P(x)) = \operatorname{sgn}\left(\lim_{x\to\infty} P(x)\right)$$

*In words: there exist upper and lower bounds s. t. the sign of the polynomial does not change beyond the bounds and are equal to the signs of the limit.*

*Proof.* If $P$ is constant, the statement trivially holds, so suppose $P$ is not constant. Then the limit of $P(x)$ at $-\infty$ (resp. $\infty$) is either $\infty$ or $-\infty$. Therefore, for any bound $c$, there exists an $l$ (resp. a $u$) such that $P(x)$ is beyond $c$ for every $x \leq l$ (resp. $x \geq u$). Choosing $c = 0$, we obtain that the sign remains constant for all $x \leq l$ (resp. $x \geq u$).

Also, since the $P(x)$ is either $> 0$ or $< 0$ for all $x$ below $l$ (resp. above $u$), it has no roots outside the interval $(l;u)$. Lastly, should $l < u$ not hold already, we can simply increase $u$ until it does. □

Since Sturm's theorem is based on counting sign changes, a crucial part in proving it is to formalise the counting of sign changes in a way that is convenient for subsequent proofs. In our formalisation, this was done by proving that under certain conditions, the computation of the number of sign changes of a sequence can be 'decomposed':

**Lemma 2** (`sign_changes_distrib`)**.** *Let $P_0, \ldots, P_n$ be a sequence of polynomials and $i \in \{0, \ldots, n\}$ and $x \in \mathbb{R}$. Furthermore, assume $P_i(x) \neq 0$. Then we have $\sigma(P_0, \ldots, P_n; x) = \sigma(P_0, \ldots, P_i; x) + \sigma(P_i, \ldots, P_n; x)$*

*Proof.* For our informal definition of sign changes, this is obvious; for the Isabelle definition using list operation, it can be proven using simple properties of the list functions. □

**Lemma 3** (`sign_changes_sturm_triple`)**.** *Let $P, Q, R \in \mathbb{R}[X]$ and $x \in \mathbb{R}$ with $P(x) \neq 0$ and $\operatorname{sgn}(R(x)) = -\operatorname{sgn}(P(x))$. Then $\sigma(P, Q, R; x) = 1$*

*Proof.* By case distinction. □

Later, we will also require some algebraic properties of polynomials, such as how to eliminate multiple roots of polynomials:

**Lemma 4** (`Misc_Polynomial.poly_div_gcd_squarefree`)**.** *Let $P \in \mathbb{R}[X] \setminus \{0\}$. Let $D := \gcd(P, P')$. Then the following holds:*

1. $\gcd(P/D, (P/D)') = 1$, *i. e. $P/D$ and its derivative are coprime, meaning that $P/D$ is* square-free *and thus has no multiple roots*
2. $\forall x \in \mathbb{R}.\ (P/D)(x) = 0 \iff P(x) = 0$, *i. e. $P$ and $P/D$ have the same roots, disregarding multiplicity*

*Proof.* Omitted due to complexity; see formal proof in Isabelle. □

### 3.3 Relating roots and Sturm sequences

We will now show how Sturm sequences can be used to determine the number of roots of polynomials in a given interval. To this end, we shall first explore when and how the number of sign changes of a Sturm sequence changes in the neighbourhood of roots and non-roots.

Informally, what we will show is that when passing through the real numbers in ascending order and tracking the number of sign changes of a Sturm sequence of a real polynomial $P$, that number decreases by 1 every time one passes a root of $P$ and remains constant otherwise. From this, it should be intuitively clear that Sturm sequences can be used to count roots.

First, we show that the number of sign changes does not change in the neighbourhood of a non-root:

**Lemma 5** (`hd_nonzero_imp_sign_changes_const`)**.** *Let $P_0, \ldots, P_n$ be a quasi-Sturm sequence and $x_0 \in \mathbb{R}$. Assume $P_0(x_0) \neq 0$. Then $\sigma(P_0, \ldots, P_n; x)$ is constant for all $x$ in a neighbourhood of $x_0$.*

*Proof.* By induction over the length of the sequence

- if the sequence has length 1, the number of sign changes is, of course, 0 everywhere.
- if the sequence has length 2, we know that $P_1$ does not change its sign anywhere by definition of a quasi-Sturm sequence. Furthermore, since $P_0(x_0) \neq 0$, the sign of $P_0$ does not change in the neighbourhood of $x_0$, as polynomials are continuous. Since the signs are constant, the number of sign changes of the sequence $P_0, P_1$ then also remains constant in the neighbourhood of $x_0$.
- if the sequence has length $\geq 3$ and $P_1(x_0) \neq 0$, then we have $P_1(x) \neq 0$ for all $x$ in a neighbourhood of $x_0$ due to continuity. Lemma 2 then implies
  $$\sigma(P_0, \ldots, P_n; x) = \sigma(P_0, P_1; x) + \sigma(P_1, \ldots, P_n; x)$$
  (for all $x$ in a NH of $x_0$)
  The first summand is constant in a neighbourhood of $x_0$ because $P_0(x_0)$ and $P_1(x_0)$ and non-zero, so their signs do not change in a neighbourhood of $x_0$. The second summand is also constant in a neighbourhood of $x_0$ by induction hypothesis.
- if the sequence has length $\geq 3$ and $P_1(x_0) = 0$, we have $P_0(x_0)P_2(x_0) < 0$ by the definition of a quasi-Sturm sequence and thus $P_0(x_0) \neq 0$, $P_2(x_0) \neq 0$, and $\operatorname{sgn}(P_2(x_0)) = -\operatorname{sgn}(P_0(x_0))$. Due to continuity, we then have $P_0(x) \neq 0$, $P_2(x) \neq 0$, and $\operatorname{sgn}(P_2(x)) = -\operatorname{sgn}(P_0(x))$ for all $x$ in a neighbourhood of $x_0$. With Lemma 2, we have:
  $$\sigma(P_0, \ldots, P_n; x) = \sigma(P_0, P_1, P_2; x) + \sigma(P_2, \ldots, P_n; x)$$
  (for all $x$ in a NH of $x_0$)
  With Lemma 3, we then have $\sigma(P_0, P_1, P_2; x) = 1$ (i. e. constant). As for the second summand, we know that $P_2, \ldots, P_n$ is again a quasi-Sturm sequence and $P_2(x_0) \neq 0$, so we can apply the induction hypothesis and obtain that $\sigma(P_2, \ldots, P_n; x)$, too, is constant for all $x$ in the neighbourhood of $x_0$.

□

Note: the Isabelle proof works with a recursive function called `split_sign_changes` that breaks the quasi-Sturm sequence into a series of sequences of length $\leq 3$ in the same manner as the induction above. This allows us to carry out the proof in several smaller steps. Moreover, the definition of `split_sign_changes` as a recursive function in Isabelle automatically generates an appropriate induction rule that makes the proof more convenient.

Now we will show that the number of sign changes decreases when passing through a root:

**Lemma 6** (`p_zero`)**.** *Let $P_0, \ldots, P_n$ be a Sturm sequence and $x_0 \in \mathbb{R}$. Assume $P_0 \neq 0$ and $P_0(x_0) = 0$. Then:*

$$\sigma(P_0, \ldots, P_n; x) = \begin{cases} \sigma(P_0, \ldots, P_n; x_0) + 1 & \text{for } x < x_0 \\ \sigma(P_0, \ldots, P_n; x_0) & \text{for } x \geq x_0 \end{cases}$$

(for all $x$ in a NH of $x_0$)

*Proof.* From the definition of a Sturm sequence, we know that $P_0(x_0) = 0$ implies $P_1(x_0) \neq 0$ . Therefore, $P_1(x_0)$ has the same non-zero sign in a neighbourhood of $x_0$. Moreover, for all $x$ in a neighbourhood of $x_0$, we have $P_0(x)P_1(x) < 0$ if $x < x_0$ and $P_0(x)P_1(x) > 0$ if $x > x_0$ . With Lemma 2, we have:

$$\sigma(P_0,\ldots,P_n;x) = \sigma(P_0,P_1;x) + \sigma(P_1,\ldots,P_n;x)$$
(for all $x$ in a NH of $x_0$)

Since $P_1(x) \neq 0$ in a neighbourhood of $x_0$, we can apply Lemma 5 and obtain:

$$\sigma(P_1,\ldots,P_n;x) = \sigma(P_1,\ldots,P_n;x_0)$$
(for all $x$ in a NH of $x_0$)

Of course, since $P_0(x_0) = 0$ , we also have:

$$\sigma(P_1,\ldots,P_n;x_0) = \sigma(P_0,\ldots,P_n;x_0)$$

In summary, we have shown so far that:

$$\sigma(P_0,\ldots,P_n;x) = \sigma(P_0,P_1;x) + \sigma(P_0,\ldots,P_n;x_0)$$
(for all $x$ in a NH of $x_0$)

Therefore, what remains to be shown is that $\sigma(P_0,P_1;x) = 1$ for $x < x_0$ and $\sigma(P_0,P_1;x) = 0$ for $x > x_0$ for all $x$ in a neighbourhood of $x_0$. This is a simple consequence of $P_0(x)P_1(x) < 0$ for $x < x_0$ and $P_0(x)P_1(x) > 0$ for $x > x_0$. □

Using these results, we can now prove Sturm's Theorem with induction over the number of roots in the given interval:

**Lemma 7** (`count_roots_between`)**.** *Let $P_0,\ldots,P_n$ be a Sturm sequence with $P_0 \neq 0$ and $x_0 \in \mathbb{R}$ . Fix $a,b \in \mathbb{R}$ with $a \leq b$ . Then the following holds:*

$$|\{x \in (a;b] \mid P_0(x) = 0\}| = \\ = \sigma(P_0,\ldots,P_n;a) - \sigma(P_0,\ldots,P_n;b)$$

*Proof.* By induction over $k := |\{x \in (a;b] \mid P_0(x) = 0\}|$ for arbitrary $a$, $b$.

- if $k = 0$ , we have $P_0(x) \neq 0$ for all $x \in (a;b]$ . We now distinguish two cases:
  - if $P_0(a) \neq 0$ , we have $P_0(x) \neq 0$ for all $x \in [a;b]$ . Lemma 5 then implies that $\sigma(P_0,\ldots,P_n)$ is constant in the neighbourhood of all $x \in [a;b]$ and must therefore be constant in the entire interval $[a;b]$ .
  - if $P_0(a) = 0$ , we know from Lemma 6 that
    $$\sigma(P_0,\ldots,P_n;x) = \sigma(P_0,\ldots,P_n;a)$$
    for all $x \in (a;a+\varepsilon)$ for some $\varepsilon > 0$ , i. e. $\sigma(P_0,\ldots,P_n)$ is constant in the right neighbourhood of $a$. Furthermore, using $P_0(x) \neq 0$ for all $x \in (a;b]$ and Lemma 5, we obtain that $\sigma(P_0,\ldots,P_n)$ is constant in the neighbourhood of every $x \in (a;b]$ . Therefore, $\sigma(P_0,\ldots,P_n)$ is constant on the entire interval $[a;b]$.

  Since $\sigma(P_0,\ldots,P_n)$ is constant on the entire interval $[a;b]$ , we have
  $$\sigma(P_0,\ldots,P_n;a) - \sigma(P_0,\ldots,P_n;b) = 0\,.$$
  This is exactly the number of roots of $P_0$ in $(a;b]$ .

- if $k > 0$ , we take the smallest root of $P_0$ in the interval $(a;b]$ and call it $x_2$. From Lemma 6, we know that for all $x$ in a sufficiently small left neighbourhood of $x_2$:
  $$\sigma(P_0,\ldots,P_n;x) = \sigma(P_0,\ldots,P_n;x_2) + 1$$
  Let $x_1$ then be some value in $(a;x_2)$ that is in this neighbourhood. We know that $P_0$ has no roots in the interval $(a;x_1]$ and thus
  $$\sigma(P_0,\ldots,P_n;a) \overset{\text{IH}}{=} \\ \overset{\text{IH}}{=} \sigma(P_0,\ldots,P_n;x_1) = \\ = \sigma(P_0,\ldots,P_n;x_2) + 1$$
  Furthermore, the number of roots of $P_0$ in the interval $(x_2;b]$ is exactly $k-1$ , so by induction hypothesis,
  $$\sigma(P_0,\ldots,P_n;x_2) - \sigma(P_0,\ldots,P_n;b) = k-1$$
  Adding these two equations yields
  $$\sigma(P_0,\ldots,P_n;a) - \sigma(P_0,\ldots,P_n;b) = k$$

□

**Lemma 8** (`count_roots`)**.** *Let $P_0,\ldots,P_n$ be a Sturm sequence with $P_0 \neq 0$ . Then the following holds:*

$$|\{x \mid P_0(x) = 0\}| = \\ = \sigma(P_0,\ldots,P_n;-\infty) - \sigma(P_0,\ldots,P_n;\infty)$$

*Proof.* Using Lemma 1 for every $P_i$ and taking the smallest lower bound $l$ and largest upper bound $u$, we obtain $l,u \in \mathbb{R}$ with $l < u$ such that $P_0$ has no roots outside the interval $(l;u)$ and for all $P_i$:

$$\operatorname{sgn}(P_i(l)) = \operatorname{sgn}\left(\lim_{x\to-\infty} P_i(x)\right)$$
$$\operatorname{sgn}(P_i(u)) = \operatorname{sgn}\left(\lim_{x\to\infty} P_i(x)\right)$$

It is then easy to see that the number of sign changes of the sequence $P_0,\ldots,P_n$ at $l$ resp. $u$ can be determined by considering the signs of the limits of the $P_i$ at $\pm\infty$ instead of $P_i(u)$ and $P_i(l)$ . Since all roots of $P_0$ lie in the interval $(l;u)$ , we have thus shown that the total number of roots of $P_0$ can be determined in the way described above. □

Note: similar statements for the usage of $\sigma$ at $\pm\infty$ to compute the number of roots $> a$, $< a$, $\geq a$, or $\leq a$ for some fixed $a \in \mathbb{R}$ can be proven analogously.

## 4. Constructing Sturm sequences

### 4.1 The canonical Sturm sequence

The canonical Sturm sequence $P_0,\ldots,P_n$ of a polynomial $P \in \mathbb{R}[X]$ is constructed as follows:

$$P_i = \begin{cases} P & \text{for } i = 0 \\ P' & \text{for } i = 1 \\ -P_{i-2} \bmod P_{i-1} & \text{otherwise} \end{cases}$$

The corresponding Isabelle function is `Sturm_Theory.sturm`.

$n$ is chosen such that $n \geq 1$ and $P_n$ is constant, since this will simplify formalisation. Choosing a higher value for $n$ only results in a number of zero polynomials at the end of the sequence, which do not change the result in any way. Note that this construction always terminates as the degree of the polynomials involved strictly decreases with every step, so that one reaches a constant polynomial in finitely many steps.

We will now show that this construction is indeed a Sturm sequence as defined before.

**Lemma 9** (`sturm_gcd`)**.** *In a canonical Sturm sequence $P_0, \ldots, P_n$ any $P_i$ is divided by* $\gcd(P, P')$.

*Proof.* By induction on $i$ using the fact that

$$\gcd(R,S) = \gcd(S, R \bmod S) = \gcd(S, -R \bmod S)$$

□

**Lemma 10** (`sturm_adjacent_root_not_squarefree`)**.** *If two adjacent polynomials in a canonical Sturm sequence have a common root, the root is a multiple root of the first polynomial.*

*Proof.* We show that if $P_i(x) = P_{i+1}(x) = 0$ , then $\forall i.\ P_i(x) = 0$, i. e. if $x$ is a root of two adjacent polynomials in the canonical Sturm sequence, then it is also a root of all the polynomials to their left. This can easily be shown by induction over $i$, using the definition of the canonical Sturm sequence.

We then have, in particular, that $P_0(x) = P_1(x) = P_0'(x) = 0$ , which means that $x$ is a multiple root of $P_0$. □

**Lemma 11** (`sturm_seq_sturm`)**.** *Let $P \in \mathbb{R}[X]$ with no multiple roots. Then the canonical Sturm sequence construction of P yields an actual Sturm sequence.*

*Proof.* We simply prove the five conditions a Sturm sequence has to satisfy:

- *The sequence has at least length 2*
  *Proof.* Obvious, by definition
- *$P_n$ does not change its sign*
  *Proof.* Obvious, since $P_n$ is constant
- *$P_0$ and $P_1$ have no common roots:*
  *Proof.* $P_0 = P$ and $P_1 = P'$ , so if $P_0$ and $P_1$ had a common root, it would be a multiple root of $P$, which contradicts our assumption.
- *For any $i \in \{0, \ldots, n-2\}$ and any root $x_0$ of $P_{i+1}$, the property $P_i(x_0)P_{i+2}(x_0) < 0$ holds*
  *Proof.* By construction, we have $P_{i+2} = -P_i \bmod P_{i+1}$ and thus $P_i = P_{i+1}Q - P_{i+2}$ for some $Q \in \mathbb{R}[X]$ . With $P_{i+1}(x_0) = 0$ , this implies that $P_{i+2}(x) = -P_i(x)$ , and because of Lemma 10 and the fact that $P$ has no multiple roots, we also have $P_i(x) \neq 0$ and thus $P_i(x)P_{i+2}(x) < 0$ .
- *For any root $x_0$ of $P_0$, the product $P_0(x)P_1(x)$ is negative in some sufficiently small interval $(x_0 - \varepsilon; x_0)$ and positive in some sufficiently small interval $(x_0; x_0 + \varepsilon)$*
  *Proof.* Since $P_0 = P$ and $P_1 = P'$ are polynomials and therefore continuous, there exists some neighbourhood of $x_0$ that does not contain any roots of either $P_0$ or $P_1$. For any $x < x_0$ in that neighbourhood, we can then apply the mean value theorem to obtain some $\xi \in [x; x_0]$ with

  $$P'(\xi)(x - x_0) = P(x) - P(x_0) = P(x)$$

  Due to $x < x_0$ , this implies $\operatorname{sgn}(P(x)) = -\operatorname{sgn}(P'(\xi))$ . Furthermore, recall that we chose a neighbourhood of $x_0$ that contains no roots of $P$ or $P'$; therefore,

  $$0 \neq \operatorname{sgn}(P'(\xi)) = \operatorname{sgn}(P'(x)) = -\operatorname{sgn}(P(x))$$

  This directly implies $P(x)P'(x) < 0$ . Similarly, for any $x > x_0$ in that neighbourhood, we can use the same argument to find that $P(x)P'(x) > 0$ .

□

### 4.2 The case of multiple roots

We now need to find a way to handle polynomials that contain multiple roots. Of course, one way to handle a polynomial with multiple roots is to use Lemma 4 and 'divide away' the 'excess roots' by dividing the polynomial $P$ by $\gcd(P, P')$ . However, it turns out that as long as the interval bounds $a$ and $b$ are not multiple roots of $P$ themselves, i. e. at least one of $P(a) \neq 0$ and $P'(a) \neq 0$ and at least one of $P(b) \neq 0$ and $P'(b) \neq 0$ holds, one can simply use the canonical Sturm sequence without any modification. To show this, we will first prove the following auxiliary result:

**Lemma 12** (`sturm_seq_sturm_squarefree'`)**.** *Let $P \in \mathbb{R}[X] \setminus \{0\}$ . Let $P_0, \ldots, P_n$ be the result of the canonical Sturm sequence construction of P and define $Q_i := P_i / \gcd(P, P')$ . Then $Q_0, \ldots, Q_n$ is a Sturm sequence and $Q_0$ has the same roots, disregarding multiplicity, as P.*

*Proof.* First, we note that $\gcd(P, P') \neq 0$ , since $P \neq 0$ by assumption. Furthermore, Lemma 9 implies that $\gcd(P, P') \mid P_i$ for all $i$, thus $Q_i \in \mathbb{R}[X]$ , i. e. the $Q_i$ really are polynomials. This means that our definition of $Q_0, \ldots, Q_n$ is well-defined. Now we show that the five properties for a Sturm sequence are satisfied:

- *The sequence has at least length 2*
  *Proof.* Obvious by construction of $Q_0, \ldots, Q_n$
- *$Q_n$ does not change its sign*
  *Proof.* Obvious; $P_n$ is constant, so $Q_n$ is, too
- *$Q_0$ and $Q_1$ have no common roots*
  *Proof.* By construction, we know that $Q_0 = P / \gcd(P, P')$ and $Q_1 = P' / \gcd(P, P')$ . Obviously, $Q_0$ and $Q_1$ are then coprime and cannot have any common roots.
- *For any $i \in \{0, \ldots, n\}$ and any root $x_0$ of $Q_{i+1}$, the property $Q_i(x_0)Q_{i+2}(x_0) < 0$ holds*
  *Proof.* By construction of the canonical Sturm sequence, we have:

  $$P_{i+2} = -P_i \bmod P_{i+1}$$

  Therefore, we have:

  $$(P_i \operatorname{div} P_{i+1}) \cdot P_{i+1} - P_{i+2} = P_i$$

  Let $D := \gcd(P, P')$ . Since $D$ divides all the $P_j$, we have $P_i = D \cdot Q_i$ , $P_{i+1} = D \cdot Q_{i+1}$ , and $P_{i+2} = D \cdot Q_{i+2}$ and thus:

  $$(D \cdot Q_i \operatorname{div} D \cdot Q_{i+1}) \cdot D \cdot Q_{i+1} - D \cdot Q_{i+2} = D \cdot Q_i$$

  Cancelling $D$ (allowed since $\mathbb{R}[X]$ is an integral domain and $D \neq 0$) yields:

  $$(Q_i \operatorname{div} Q_{i+1}) \cdot Q_{i+1} - Q_{i+2} = Q_i$$

  Since $x_0$ is a root of $Q_{i+1}(x_0)$ , we then have:

  $$-Q_{i+2}(x_0) = Q_i(x_0)$$

  Since $\gcd(Q_i, Q_{i+1}) = 1$ and $x_0$ is a root of $Q_{i+1}(x_0)$ , $x_0$ cannot be a root of $Q_i$ and thus we have:

  $$Q_i(x_0)Q_{i+2}(x_0) = -Q_i(x_0)^2 < 0$$

- *For any root $x_0$ of $Q_0$, the product $Q_0(x)Q_1(x)$ is negative in some sufficiently small interval $(x_0 - \varepsilon; x_0)$ and positive in some sufficiently small interval $(x_0; x_0 + \varepsilon)$:*
  *Proof.* Let, again, $D := \gcd(P, P')$ . Obtain some $\varepsilon > 0$ such that the following two properties hold:
  1. $D$ does not have a root in $(x_0 - \varepsilon; x_0 + \varepsilon) \setminus \{x_0\}$
  2. $P_0(x)P_1(x) < 0$ for all $x \in (x_0 - \varepsilon; x_0)$ and $P_0(x)P_1(x) > 0$ for all $x \in (x_0; x_0 + \varepsilon)$

Then we have, for any $x$ in $(x_0-\varepsilon;x_0+\varepsilon)\setminus\{x_0\}$:

$$P_0(x)P_1(x) = Q_0(x)Q_1(x)\cdot\underbrace{D(x)^2}_{>0}$$

This then obviously implies the property we want to show.

It remains to show that $Q_0$ has the same roots, disregarding multiplicity, as $P = P_0$. This is precisely the statement of Lemma 4. □

Of course, there is no reason to use this construction in practice, since if one has computed $\gcd(P,P')$ already, it is easier to compute the canonical Sturm sequence of $P/\gcd(P,P')$ directly than to compute the canonical Sturm sequence of $P$ and then divide every polynomial in it by $\gcd(P,P')$. However, the result becomes useful when combined with the following insight:

**Lemma 13** (`sturm_sturm_squarefree'_same_sign_changes`)**.** *Let $P \in \mathbb{R}[X]\setminus\{0\}$. Let $P_0,\ldots,P_n$ and $Q_0,\ldots,Q_n$ be as in the previous lemma. Then the following holds:*

1. $\forall x{\in}\mathbb{R}.\ P(x)\neq 0 \vee P'(x)\neq 0 \Longrightarrow \sigma(Q_0,\ldots,Q_n;x) = \sigma(P_0,\ldots,P_n;x)$
2. $\sigma(Q_0,\ldots,Q_n;\pm\infty) = \sigma(P_0,\ldots,P_n;\pm\infty)$

*Proof.*

1. Let $D := \gcd(P,P')$. Consider an arbitrary $x\in\mathbb{R}$ for which $P(x)\neq 0$ or $P'(x)\neq 0$, i. e. $D(x)\neq 0$. Then we have for all $i$:

   $$\operatorname{sgn}(Q_i(x)) = \operatorname{sgn}(P_i(x)/D(x)) = \operatorname{sgn}(P_i(x))\cdot\operatorname{sgn}(D(x))$$

   It is now easy to see that in both of the two cases $D(x)>0$ and $D(x)<0$, the number of sign changes in the two sequences is the same, since the signs in $Q_0,\ldots,Q_n$ are all either the same as in $P_0,\ldots,P_n$ or all flipped w. r. t. the signs in $P_0,\ldots,P_n$.
2. We now consider the case of $\pm\infty$. Since $P_i = Q_i\cdot D$, we obviously have $\operatorname{lc}(P_i) = \operatorname{lc}(D)\cdot\operatorname{lc}(Q_i)$. When considering the way in which $\sigma$ at $\pm\infty$ can be computed using the leading coefficients, it is then again obvious that the two sign sequences are either the same or one is 'flipped' w. r. t. the other; either way, we have $\sigma(Q_0,\ldots,Q_n;\pm\infty) = \sigma(P_0,\ldots,P_n;\pm\infty)$.

□

In conclusion: if we want to count the roots of some $P$ between bounds $a,b\in\tilde{\mathbb{R}}$ with $a\leq b$, we use $\sigma$ with a Sturm sequence of $P$, and such a Sturm sequence can be obtained by applying the canonical Sturm sequence construction to $P$ if neither $a$ nor $b$ are roots of both $P$ and $P'$, or by applying it to $P/\gcd(P,P')$ if they are.

## 5. Proof Method

We now use the results proven so far in order to implement an Isabelle proof method for a number of interesting categories of properties of real polynomials.[2] The name of this proof method is `sturm`. Given a goal statement, `sturm` performs some pre-processing and analysis and applies an appropriate rule. This replaces the goal with another, executable statement involving Sturm sequences. Then, Isabelle's code generator [4] is invoked on this new goal in order to prove it by evaluation.

We will now give an overview of the different categories of statements `sturm` can prove and how it achieves this.

***Number of roots.*** The statements that were proven in Lemma 7 and Lemma 8 were that Sturm sequences can be used to count the roots of a polynomial $P$ in an interval of the form $(a;b]$ for $a,b\in\tilde{\mathbb{R}}$ with $a<b$. However, this can be generalised to arbitrary intervals $(a;b)$, $[a;b)$, $(a;b]$, and $[a;b]$ for $a,b\in\tilde{\mathbb{R}}$. This is achieved by performing a case distinction on whether $a$ and $b$ are roots themselves and, if they are, adding/subtracting 1 from the number determined with the Sturm sequence.

***Pre-processing.*** Pre-processing of the goal statement is performed in order to normalise statements such as $x\cdot x = x$ to the form

$$\text{poly } [:0,-1,1:]\ x = 0$$

and analogously for inequalities. Furthermore, statements such as $\forall x{\in}I.\ P(x)\neq 0$ can be reduced to counting the number of roots and checking that it is 0.

***Logical connectives.*** We can further generalise our method to allow composition of polynomial equations with the logical operators $\wedge$ and $\vee$. To do that, note that the following holds:

$$P(x)=0\wedge Q(x)=0 \quad\Longleftrightarrow\quad \gcd(P,Q)(x)=0$$
$$P(x)=0\vee Q(x)=0 \quad\Longleftrightarrow\quad (P\cdot Q)(x)=0$$

We can therefore reduce any combination of polynomial equations with $\wedge$ and $\vee$ to a single equation of the form $P(x)=0$.

***Strict universally-quantified inequalities.*** Note that $\forall x.\ P(x)>0$ holds iff $P$ has no roots and $\lim_{x\to\infty}P(x)=\infty$. Also, $\forall x{\in}I.\ P(x)>0$ holds iff $P$ has no roots in $I$ and $P(x)>0$ for some arbitrary $x\in I$. This allows us to decide single strict inequalities on polynomials, i. e. $\forall x{\in}I.\ P(x)>Q(x)$. However, we cannot decide complex combinations of several inequalities with $\wedge$ and $\vee$ such as $\forall x{\in}I.\ P_1(x)>Q_1(x)\vee P_2(x)>Q_2(x)$.

***Non-strict universally-quantified inequalities.*** Deciding non-negativity (resp. non-positivity) of polynomials is more involved. We need to split the interval in question into disjoint and adjacent sub-intervals such that each of these sub-intervals contains exactly one root and the polynomial is positive (resp. negative) at the border of each of them. (Some additional case distinction is necessary if the interval borders themselves are roots.)

It is more convenient to compute the required partition of the interval *outside* of Isabelle, in unverified ML code. This partition is then used as a *witness* to prove the non-negativity (resp. non-positivity) of the polynomial by using a rule that states that if the premises stated above hold for a partition of an interval, the polynomial is non-negative (resp. non-positive) on the interval.

To prove non-negativity of $P$ on $(a;b]$[3], the ML code finds suitable splitting points $c_1,\ldots,c_k$ by bisection. These splitting points satisfy $a<c_1<\ldots<c_k<b$. To find these points, bisection is performed with the initial partition $(a;b]$ until every sub-interval contains exactly one root. If one of the splitting points found by bisection is a root of $P$, it is shifted to the side by a sufficiently small amount. The end result is then an ascending list of points $a,c_1,\ldots,c_k,b$, none of which are roots, such that exactly one root lies between two adjacent numbers. It is then easy to see that the polynomial is non-negative iff it is positive at each of these points. Obviously, this algorithm always terminates and always finds a correct partition.

[2] To be precise, while it does count *real* roots, both the polynomial coefficients and the bounds of the interval of interest must be from a set of numbers with which Isabelle's code generator can perform computations. At the moment, this means rational numbers; in the future, it could be extended to e. g. algebraic real numbers. General real numbers are, of course, problematic for computations on a computer.

[3] If the given interval bounds are infinite, the algorithm first replaces them by equivalent finite bounds by doubling the initial guess 1 until all roots lie within these bounds. If $a$ or $b$ are roots themselves, they are shifted to the left or right by a sufficiently small amount. This shifting can be justified by the fact that non-zero polynomials are continuous and only have finitely many roots.

**lemma** card $\{x :: \text{real}.\ (x-1)^2 * (x+1) * (x^2+1) = 0\} = 2$ **by** sturm

**lemma** card $\{x :: \text{real}.\ -0.010831 < x \wedge x < 0.010831 \wedge$
$\text{poly}\ [:0, -17/2097152, -49/16777216, 1/6, 1/24, 1/120:]\ x = 0\} = 3$ **by** sturm

**lemma** card $\{x :: \text{real}.\ x^3 + x = 2 * x^2 \wedge x^3 - 6 * x^2 + 11 * x = 6\} = 1$ **by** sturm

**lemma** card $\{x :: \text{real}.\ x^3 + x = 2 * x^2 \vee x^3 - 6 * x^2 + 11 * x = 6\} = 4$ **by** sturm

**lemma** $(x :: \text{real})^2 + 1 > 0$ **by** sturm

**lemma** $(x :: \text{real}) > 1 \implies x^3 > 1$ **by** sturm

**lemma** $[\![(x :: \text{real}) > 0; x \leq 2/3]\!] \implies x * x \neq x$ **by** sturm

**lemma** $|(x :: \text{real}) - 3| < 2 \implies 5 + 6 * x - x^2 > 0$ **by** sturm

**lemma** $(x :: \text{real}) \geq -1 \Longrightarrow x \leq 2 \Longrightarrow 3 * x^3 + 2 \geq x^4 + x^2 + 3 * x$ **by** sturm

**lemma** $\exists x :: \text{real}.\ -1 \leq x \wedge x \leq 1 \wedge 1 - x * x \geq 0$ **by** sturm

**lemma** mono $(\lambda x :: \text{real}.\ x^3 + 12 * x^2 + 60 * x + 120)$ **by** sturm

**lemma** strict_mono $(\lambda x :: \text{real}.\ x^5 + 30 * x^4 + 420 * x^3 + 3360 * x^2 + 15120 * x + 30240)$ **by** sturm

**lemma** $(x :: \text{real}) \neq y \Longrightarrow x^3 \neq y^3$ **by** sturm

**lemma** abs $(x :: \text{real}) < 1 \Longrightarrow$
$(3969/65536 + 63063/4096 * x^6 + 1792791/4096 * x^{10} + 3002285/4096 * x^{18} + 6600165/4096 * x^{14}$
$- 72765/65536 * x^4 - 3558555/32768 * x^8 - 10207769/65536 * x^{20} - 35043645/32768 * x^{12}$
$- 95851899/65536 * x^{16})^{10} > 0$ **by** sturm

**Figure 2.** Concrete examples of statements that can be proven in Isabelle/HOL by the `sturm` method.

The fact that this code is unverified is relatively unproblematic as it does not compromise soundness in any way. If it were to contain a bug and return an incorrect partition, the proof method would fail even if the goal was provable, rendering the proof method *incomplete*. Soundness, on the other end, is not affected, as an incorrect witness would still be rejected by Isabelle.

***Monotonicity.*** An obvious application of deciding non-strict inequalities is in proving monotonicity of polynomials, i. e. that a polynomial is monotonically increasing. This is done by proving non-negativity of the derivative $P'$, which is equivalent to monotonicity of $P$. Similarly, strict monotonicity is equivalent to non-negativity of $P'$ and $P' \neq 0$. Monotonically decreasing polynomials can be handled by proving that their negation is monotonically increasing.

***Injectivity.*** It is easy to see that a polynomial is injective iff it is strictly monotonically increasing or strictly monotonically decreasing. We can therefore also decide injectivity.

***Existentials.*** Since we can decide universally quantified inequalities with the operators $<$, $\leq$, $\neq$, $\geq$, $>$, we can also decide the opposite: existentials with the operators $<$, $\leq$, $=$, $\geq$, $>$. The cases of $\leq$, $\geq$, and $=$ can simply be mapped to universally-quantified statements with $>$, $<$, and $\neq$, since they only require simple root counting.

The cases of $<$ and $>$, on the other hand, correspond to negations of universally-quantified non-strict inequalities. Since the decision procedure for these runs externally, providing a witness with which they can be proven in Isabelle, we cannot simply use our existing code for non-strict inequalities and negate the result – we must instead provide a witness for 'non-non-negativity'.

The obvious candidate for such a witness is a point $x$ in the desired interval such that $P(x) < 0$ . We therefore use the same interval splitting code in ML as we used before to compute splitting points and then check if the polynomial is negative at any of them, returning the first one that is. Additional care must be taken to ensure that the witness returned is indeed *inside* the interval and not at its border if the interval is open. If necessary, the witness is shifted inwards by some sufficiently small value.

Again, this method always terminates and always finds a witness (if it exists). It is therefore complete.

***Code Generation.*** As mentioned above, the `sturm` method relies on the Code Generator to prove statements by evaluation. For a statement that can be proven by evaluation, Standard ML code is generated and executed, and the result becomes a theorem in Isabelle. The Code Generator is therefore part of Isabelle's trusted code base. For a detailed explanation of Isabelle code generation, see the corresponding work by Haftmann and Nipkow [4].

***Examples.*** For a number of representative examples of statements provable by `sturm`, see figure 5. For a detailed documentation of the `sturm` proof method, see the user guide in the corresponding entry in the *Archive of Formal Proofs* [3].

## 6. Evaluation

The `sturm` method has direct practical applications within Isabelle whenever a simple property of real polynomials has to be shown. A direct, *ad-hoc* proof for properties such as 'The polynomial $P$ has exactly 3 roots in the interval $[-1;1]$' is usually difficult – with `sturm`, a proof can be derived fully automatically.

$$\text{mono } (\lambda x.\ x^3 + 12 * x^2 + 60 * x + 120)$$
$$-x^4/(x^2 - 6 * x + 12)^2 \leq 0$$

**Figure 3.** Two simple examples on which `sos` with CSDP takes 30 and 60 seconds to find a SOS witness, whereas `sturm` can easily prove them within a few seconds. (the latter currently requires manual rewriting into a polynomial inequality)

Apart from `sturm`, there are two other proof methods that can be used in Isabelle to prove properties of real polynomials automatically:

**Sum of squares.** The `sos` method uses an external tool to decompose a polynomial into a sum of squares, which directly implies non-negativity [6]. For univariate real polynomials, this method is complete in the sense that every non-negative univariate real polynomial can be decomposed into $P^2 + Q^2$, where $P, Q \in \mathbb{R}[X]$ [9]. It also works for multivariate polynomials, but is only complete for univariate ones.

The `sos` method works by invoking external tools such as Neos or CSDP to obtain a sum-of-squares decomposition of the polynomial in question. This decomposition is then used as a witness for the proof and can be checked very quickly. However, *finding* such a witness is very difficult; in practice, this method does not work reliably on polynomials of practical interest, since the search for a suitable decomposition often takes intolerably long. Its completeness is therefore only theoretical; see Fig. 3 for an example on which `sos` performs comparatively poorly.

**Approximation.** The `approximation` method uses interval arithmetic in order to prove arithmetical statements by approximation [7]. This is a very powerful and general method, but it often performs poorly on simple polynomial statements, especially if the area of interest is close to a root.

***Comparison.*** Compared to these methods, the `sturm` method is less general and tailored specifically to real polynomials. It does not suffer from any of the performance problems with polynomials that the other two methods have. The computation of the Sturm sequence is straightforward and can be done efficiently even for polynomials of high degree and interval splitting for non-strict inequalities is done by bisection and should also work for any polynomial without significant performance problems.

Larry Paulson uses `sturm` on a number of practical examples in his work on upper and lower bounds for the real exponential function. This work is related to the MetiTarski system [1, 12], which requires knowledge of explicit bounds for functions such as the exponential function, trigonometric functions, etc. The problems in question are simple polynomial inequalities such as $p(x) \geq 0$, $-4.64 \leq x \Longrightarrow p(x) > 0$, or monotonicity of a polynomial. The polynomials involved have degrees $\leq 7$. For two examples, see Fig. 3.

These were previously proven with a sum-of-squares certificate wherever one could be found. In all other cases, the `approximation` method was used with appropriate manually-tuned parameters and often manual splitting of the real line into several segments, proving the desired property for each of them separately.

The sum-of-squares method works almost instantaneously, since its certificates can be verified quickly. Finding the certificates initially, however, often takes very long (between 30 and 150 seconds in the examples in Fig. 3). The `approximation` method, on the other hand, takes between 1 and 10 seconds[4] for each invocation on the examples we considered, with about four separate invocations for every property.

The `sturm` method was able to prove all the properties that were previously handled by `sos` and `approximation` in about 2 seconds without any need for a long certificate search or interval splitting.[5] In fact, even inequalities of polynomials with a degree of 100 or 200 (see Fig. 5) are still proven by `sturm` within a few seconds.

Most of the execution time of the `sturm` method is taken up by code generation, since currently, the code for the method is generated anew with every invocation. This is due to technical limitations of Isabelle's code generator. We are currently investigating if it is possible to avoid this overhead with a reasonable amount of effort. Even so, the `sturm` method is already fast enough for practical use.

***Required auxiliary results.*** While Isabelle/HOL already contained a good formalisation of univariate polynomials – most importantly real analysis on univariate polynomials – a sizeable chunk of background theory we required was missing:

- the `remdups_adj` function used in sign counting (see Par. 3.1)
- some facts about real intervals
- limits of power functions and polynomials at $\pm\infty$ and bounds for roots
- facts about the neighbourhood of roots
- the square-free part of a of polynomial (cf. Lemma 4)
- Bezout's lemma for polynomials and other facts related to the polynomial GCD

Much of this has already been integrated into Isabelle's library. The last point is of particular interest: in the course of our formalisation effort, we found the current definition of the greatest common divisor (GCD) in Isabelle/HOL somewhat lacking due to the fact that there are separate definitions of and lemmas about the GCD of natural numbers, integers, and polynomials. We have since begun work on a unified definition based on Euclidean rings, which will eliminate the need for some of the specialised results we had to prove for polynomials.

***Future work.*** As mentioned before, we are still investigating how to improve the Code Generator setup in order to speed up the proof method. Code Generation takes up the vast majority of the run time of the *sturm* method and has to be done anew with every invocation of the method. Another improvement would be additional preprocessing in order to rewrite rational function inequalities (i.e. $P(x)/Q(x)$) to polynomial ones in order to avoid the manual step required in problem such as the second example in Fig. 3.

***Related work.*** The first formalisation of Sturm's theorem in a theorem prover that we are aware of is by John Harrison [5]. However, he only proves it for square-free polynomials since he did not require the more general result for arbitrary polynomials. He also does not use Sturm's theorem to implement a decision procedure for more complex properties of polynomials such as non-negativity.

A related, albeit much more difficult problem is deciding whether a given first-order formula over the real numbers is true. McLaughlin and Harrison implemented a proof-producing version

[4] Intel® Core™ i7-4750HQ @ 2.00GHz, 16 GB RAM

[5] The only manual work required was one example which had the form $-P(x)/Q(x) \leq 0$. It had to be manually reduced to the separate goals $P(x) \geq 0$ and $Q(x) \geq 0$ before it could be proven by `sturm`, whereas `sos` can prove it directly by decomposing it into a sum of squares of rational functions.

of a decision procedure for this in HOL Light [10]. Assia Mahboubi developed a proof-producing implementation of the comparatively efficient *Cylindric Algebraic Decomposition* algorithm in Coq [8].

Julianna Zsidó proved the Theorem of Three Circles in the proof assistant *Coq* [14]. This theorem relates the number of sign changes in the sequence of Bernstein coefficients of a real polynomial to the number of roots in a certain area of the complex plane. This can be used to isolate real roots by bisection. Unlike Sturm's theorem, it does not provide the exact number of real roots in an interval directly.

Lastly, the previously-mentioned work of Anthony Narkawicz and César Muñoz [11] is the most similar to ours. Their work was carried out in parallel to and independently of ours. They proved Sturm's theorem in PVS and implemented a fully-verified decision procedure for existential and universal univariate polynomial inequalities. The capabilities of their decision procedure are therefore roughly the same as those of ours, although they perform more pre-processing for monotonicity statements. They appear to use a similar interval splitting algorithm for the case of non-strict inequalities. The main difference between their work and ours is the different proof for Sturm's theorem: Narkawicz and Muñoz give a direct proof by considering powers of linear divisors of the Sturm sequence obtained by the canonical construction. We, on the other hand, use a generalised definition of Sturm sequences not tied to a particular construction and prove that these properties are sufficient for root counting before introducing concrete constructions. Additionally, they use *pseudo division* instead of regular division on polynomials for better performance. We did not implement this since it introduces more complexity and code generation is the bottleneck in our implementation, not the arithmetic operations. Another difference is that they proved the completeness of their proof method, whereas we avoided this complicated proof by providing externally-computed witnesses.

## 7. Conclusion

We proved Sturm's theorem and an efficient way of constructing Sturm sequences in the theorem prover Isabelle/HOL. We also used reflection to turn this result into an efficient proof method for proving statements about univariate real polynomials completely automatically. Due to its being tailored specifically to proving a small (but interesting) set of properties of polynomials, it performs very well on these properties when compared to earlier, more general methods. For polynomial inequalities taken from a practical example, the new proof method was shown to shorten proofs considerably and eliminate virtually all manual proof work for these properties.

## Acknowledgments

We thank Tobias Nipkow for his helpful comments, suggesting related work, and encouraging this publication in the first place. Larry Paulson contributed interesting insights into the performance of our proof method in practice and encouraged us to implement non-strict inequalities. We also thank César Muñoz for providing us with a preliminary version of his paper, allowing us to reference it as related work and compare our work to his. Finally, our thanks also go to the anonymous reviewers for their valuable suggestions.

## References

[1] B. Akbarpour and L. C. Paulson. Metitarski: An automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 44(3):175–205, 2010. ISSN 0168-7433. . URL `http://dx.doi.org/10.1007/s10817-009-9149-2`.

[2] M. Coste. *An Introduction to Semialgebraic Geometry*. Institut de Recherche Mathématique de Rennes, `http://perso.univ-rennes1.fr/michel.coste/polyens/SAG.pdf`, October 2002.

[3] M. Eberl. Sturm's theorem. *Archive of Formal Proofs*, Jan. 2014. ISSN 2150-914x. `http://afp.sf.net/entries/Sturm_Sequences.shtml`, Formal proof development.

[4] F. Haftmann and T. Nipkow. Code generation via higher-order rewrite systems. In M. Blume, N. Kobayashi, and G. Vidal, editors, *Functional and Logic Programming (FLOPS 2010)*, volume 6009 of *Lecture Notes in Computer Science*, pages 103–117. Springer, 2010.

[5] J. Harrison. Verifying the accuracy of polynomial approximations in HOL. In E. L. Gunter and A. Felty, editors, *Theorem Proving in Higher Order Logics: 10th International Conference, TPHOLs'97*, volume 1275 of *Lecture Notes in Computer Science*, pages 137–152, Murray Hill, NJ, 1997. Springer-Verlag.

[6] J. Harrison. Verifying nonlinear real formulas via sums of squares. In K. Schneider and J. 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.

[7] J. Hölzl. Proving inequalities over reals with computation in Isabelle/HOL. In G. D. Reis and L. Théry, editors, *Proceedings of the ACM SIGSAM 2009 International Workshop on Programming Languages for Mechanized Mathematics Systems (PLMMS'09)*, pages 38–45, Munich, August 2009.

[8] A. Mahboubi. Implementing the cylindrical algebraic decomposition within the Coq system. *Mathematical Structures in Computer Science*, 17(1):99–127, Feb. 2007. ISSN 0960-1295. . URL `http://dx.doi.org/10.1017/S096012950600586X`.

[9] M. Marshall. *Positive Polynomials and Sums of Squares*. University of Saskatchewan, 2008. ISBN 978-0-8218-4402-1.

[10] S. McLaughlin and J. Harrison. A proof-producing decision procedure for real arithmetic. In R. Nieuwenhuis, editor, *CADE-20: 20th International Conference on Automated Deduction, proceedings*, volume 3632 of *Lecture Notes in Computer Science*, pages 295–314, Tallinn, Estonia, 2005. Springer Berlin Heidelberg.

[11] A. Narkawicz and C. Muñoz. A formally-verified decision procedure for univariate polynomial computation based on Sturm's theorem. *Manuscript submitted for publication*, 2014.

[12] L. C. Paulson. Metitarski web site. URL `http://www.cl.cam.ac.uk/~lp15/papers/Arith/`.

[13] J. C. F. Sturm. Mémoire sur la résolution des équations numériques. *Bulletin des Sciences de Férussac*, 11:419–425, 1829.

[14] J. Zsidó. Theorem of three circles in Coq. *Journal of Automated Reasoning*, 53(2):105–127, 2014. ISSN 0168-7433. . URL `http://dx.doi.org/10.1007/s10817-013-9299-0`.