Verified Real Number Calculations: A Library for Interval Arithmetic

Marc Daumas, David Lester, César Muñoz

Verified Real Number Calculations: A Library for Interval Arithmetic

Marc Daumas, David Lester, and César Muñoz

***Abstract*— Real number calculations on elementary functions are remarkably difficult to handle in mechanical proofs. In this paper, we show how these calculations can be performed within a theorem prover or proof assistant in a convenient and highly automated as well as interactive way. First, we formally establish upper and lower bounds for elementary functions. Then, based on these bounds, we develop a rational interval arithmetic where real number calculations take place in an algebraic setting. In order to reduce the dependency effect of interval arithmetic, we integrate two techniques: interval splitting and taylor series expansions. This pragmatic approach has been developed, and formally verified, in a theorem prover. The formal development also includes a set of customizable strategies to automate proofs involving explicit calculations over real numbers. Our ultimate goal is to provide guaranteed proofs of numerical properties with minimal human theorem-prover interaction.**

***Index Terms*— Real number calculations, interval arithmetic, proof checking, theorem proving**

I. Introduction

Deadly and disastrous failures [1]–[3] confirm the shared belief that traditional testing, simulation, and peer-review are not sufficient to guarantee the correctness of critical software. *Formal Methods* in computer science refers to a set of mathematical techniques and tools to verify safety properties of a system design and its implementation functional requirements. In the verification of engineering applications, such as aerospace systems, it is often necessary to perform explicit calculations with non-algebraic functions. Despite all of the developments concerning real analysis in theorem provers [4]–[8], the formal verification of the correctness of these calculations is not routine.

Take, for example, the formula

$$\frac{3\pi}{180} \leq \frac{g}{v}\tan(\frac{35\pi}{180}) \leq \frac{3.1\pi}{180}, \qquad (1)$$

where g is the gravitational force and $v = 250$ knots is the ground speed of an aircraft. This formula appears in the verification of NASA's Airborne Information for Lateral Spacing (AILS) algorithm [9]. It states that the turn rate of an aircraft flying at ground speed v with a bank angle of 35^o is about 3^o per second. A direct proof of this formula is about a page long and requires the use of several trigonometric properties.

In many cases the formal checking of numerical calculations is so cumbersome that the effort seems futile; it is then tempting to perform the calculations out of the system, and introduce the results as axioms.[1] However, chances are that the external calculations will be performed using floating-point arithmetic. Without formal checking of the results, we will never be sure of the correctness of the calculations.

In this paper we present a set of interactive tools to automatically prove numerical properties, such as Formula (1), within a proof assistant. The point of departure is a collection of lower and upper bounds for rational and non-rational operations. Based on provable properties of these bounds, we develop a rational interval arithmetic which is amenable to automation. The series approximations and interval arithmetic presented here are well-known. However, to our knowledge, this is the most complete formalization in a theorem prover of interval arithmetic that includes non-algebraic functions.

Our ultimate goal is to provide guaranteed formal proofs of numerical properties with minimum human effort. As automated processes are bound to fail on degenerate cases and waste time and memory on simple ones, we have designed a set of highly customizable proof strategies. The default values of the parameters are sufficient in most simple cases. However, a domain expert can set these parameters to obtain a desired result, *e.g.*, the accuracy of a particular calculation.

This paper merges and extends the results presented in [10], [11]. The rest of this document is organized as follows. Section II defines bounds for elementary functions. Section III presents a rational interval arithmetic based on these bounds. Section IV describes a method to prove numerical propositions. The implementation of this method in a theorem prover is described in Section V. Last section summarizes our work and compares it to related work.

The mathematical development presented in this paper has been written and fully verified in the Prototype Verification System (PVS) [12][2]. PVS provides a strongly typed specification language and a theorem prover for higher-order logic. It is developed by SRI International. Our development is freely available on the Internet. The results on upper and lower bounds have been integrated to the NASA Langley PVS Libraries[3] and the rational interval arithmetic and the PVS strategies for numerical propositions are available from one of

M. Daumas (`marc.daumas@lirmm.fr`) is with the LIRMM, CNRS, UM2 and ELIAUS, UPVD and he is supported in part by the PICS 2533 of the CNRS.

D. Lester (`dlester@cs.man.ac.uk`) is with the University of Manchester, Oxford Road, Manchester M13 9PL, UK and he is supported in part by the French Rgion Languedoc-Roussillon.

C. Muñoz (`munoz@nianet.org`) is with the National Institute of Aerospace, 100 Exploration Way, Hampton, VA 23666, USA and he is supported in part by he National Aeronautics and Space Administration under NASA Cooperative Agreement NCC-1-02043 and by the University of Perpignan.

[1] As a matter of fact, the original verification of NASA's AILS algorithm contained several such axioms.

[2] PVS is available from `http://pvs.csl.sri.com`.

[3] `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html`.

the authors[4].

For readability, we will use standard mathematical notations along this paper and PVS notations will be limited to illustrate the use of the library. In the following, we use the first letters of the alphabet $a, b, \ldots$ to denote rational numbers, and the last letters of the alphabet $\ldots x, y, z$ to denote arbitrary real variables. We use **boldface** for interval variables. Furthermore, if $\mathbf{x}$ is an interval variable, $\underline{\mathbf{x}}$ denotes its lower bound and $\overline{\mathbf{x}}$ denotes its upper bound.

II. Bounds for Elementary Functions

A PVS basic theory of bounds for square root and trigonometric functions was originally proposed for the verification of NASA's AILS algorithm [9]. We have completed it and extended with bounds for natural logarithm, exponential, and arctangent. The basic idea is to provide for each real function $f : \mathbb{R} \mapsto \mathbb{R}$, functions $\underline{f} : (\mathbb{R}, \mathbb{N}) \mapsto \mathbb{R}$ and $\overline{f} : (\mathbb{R}, \mathbb{N}) \mapsto \mathbb{R}$ closed under $\mathbb{Q}$, such that for all x, n

$$\underline{f}(x,n) \;\leq\; f(x) \;\leq\; \overline{f}(x,n), \tag{2}$$
$$\underline{f}(x,n) \;\leq\; \underline{f}(x,n+1), \tag{3}$$
$$\overline{f}(x,n+1) \;\leq\; \overline{f}(x,n), \tag{4}$$
$$\lim_{n\to\infty} \underline{f}(x,n) \;=\; f(x) \;=\; \lim_{n\to\infty} \overline{f}(x,n). \tag{5}$$

Formula (2) states that $\underline{f}$ and $\overline{f}$ are, respectively, lower and upper bounds of f, and formulas (3), (4), and (5) state that these bounds can ultimately be improved, as much as needed, by increasing the approximation parameter n.

For transcendental functions, we use taylor approximation series. We performed a coarse range reduction [13] since the convergence of taylor series is usually best for small values. More elaborate range reduction techniques [14] would significantly enhance the speed and the accuracy of the functions defined in Sections II and III. All the stated propositions in this section have been formally verified in the verification system PVS.

A. Square root

For square root, we use a simple approximation by Newton's method. For $x \geq 0$,

$$\begin{aligned}
\overline{\mathrm{sqrt}}(x,0) &= x+1,\\
\overline{\mathrm{sqrt}}(x,n+1) &= \frac{1}{2}(y+\frac{x}{y}), \quad \text{where } y = \overline{\mathrm{sqrt}}(x,n),\\
\underline{\mathrm{sqrt}}(x,n) &= \frac{x}{\overline{\mathrm{sqrt}}(x,n)}.
\end{aligned}$$

Proposition 1: $\forall x \geq 0, n : \; 0 \;\leq\; \underline{\mathrm{sqrt}}(x,n) \;\leq\; \sqrt{x} \;<\; \overline{\mathrm{sqrt}}(x,n)$.
The first inequality is strict when $x > 0$.

[4] http://research.nianet.org/~munoz/Interval.

B. Trigonometric functions

We use the partial approximation by series.

$$\begin{aligned}
\underline{\sin}(x,n) &= \sum_{i=1}^{m} (-1)^{i-1} \frac{x^{2i-1}}{(2i-1)!}\\
\overline{\sin}(x,n) &= \sum_{i=1}^{m+1} (-1)^{i-1} \frac{x^{2i-1}}{(2i-1)!},\\
\underline{\cos}(x,n) &= 1 + \sum_{i=1}^{m+1} (-1)^{i} \frac{x^{2i}}{(2i)!},\\
\overline{\cos}(x,n) &= 1 + \sum_{i=1}^{m} (-1)^{i} \frac{x^{2i}}{(2i)!},
\end{aligned}$$

where $m = 2n$ if $x < 0$; otherwise, $m = 2n+1$.
Proposition 2: $\forall x, n : \; \underline{\sin}(x,n) \;\leq\; \sin(x) \;\leq\; \overline{\sin}(x,n)$.
Proposition 3: $\forall x, n : \; \underline{\cos}(x,n) \;\leq\; \cos(x) \;\leq\; \overline{\cos}(x,n)$.

C. Arctangent and π

We first use the alternating partial approximation by series for $0 \leq x \leq 1$.

$$\begin{aligned}
\underline{\mathrm{atan}}(x,n) &= \sum_{i=1}^{2n+1} x^{2i+1} \frac{(-1)^i}{2i+1}, \quad \text{if } 0 < x \leq 1,\\
\overline{\mathrm{atan}}(x,n) &= \sum_{i=1}^{2n} x^{2i+1} \frac{(-1)^i}{2i+1}, \quad \text{if } 0 < x \leq 1.
\end{aligned}$$

We note that for $x = 1$ (which we might naïvely wish to use to define $\pi/4$ and hence π) the series: $1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$ *does* converge, but very slowly. Instead, we use the equality $\frac{\pi}{4} = 4\,\mathrm{atan}(1/5) - \mathrm{atan}(1/239)$, that has much better convergence properties. Using this identity we can define bounds on π:

$$\begin{aligned}
\underline{\pi}(n) &= 16\;\underline{\mathrm{atan}}(1,n) - 4\;\overline{\mathrm{atan}}(1,n),\\
\overline{\pi}(n) &= 16\;\overline{\mathrm{atan}}(1,n) - 4\;\underline{\mathrm{atan}}(1,n).
\end{aligned}$$

Proposition 4: $\forall n : \; \underline{\pi}(n) \;\leq\; \pi \;\leq\; \overline{\pi}(n)$.
Now, using properties of arctangent, we extend the range of the function to the whole set of real numbers:

$$\begin{aligned}
\underline{\mathrm{atan}}(0,n) &= \overline{\mathrm{atan}}(0,n) \;=\; 0,\\
\underline{\mathrm{atan}}(x,n) &= \frac{\underline{\pi}(n)}{2} - \overline{\mathrm{atan}}(\frac{1}{x},n), \quad \text{if } 1 < x,\\
\underline{\mathrm{atan}}(x,n) &= -\overline{\mathrm{atan}}(-x,n), \quad \text{if } x < 0,\\
\overline{\mathrm{atan}}(x,n) &= \frac{\overline{\pi}(n)}{2} - \underline{\mathrm{atan}}(\frac{1}{x},n), \quad \text{if } 1 < x,\\
\overline{\mathrm{atan}}(x,n) &= -\underline{\mathrm{atan}}(-x,n), \quad \text{if } x < 0.
\end{aligned}$$

Proposition 5: $\forall x, n : \; \underline{\mathrm{atan}}(x,n) \;\leq\; \mathrm{atan}(x) \;\leq\; \overline{\mathrm{atan}}(x,n)$.
These are strict inequalities except when $x = 0$.

The PVS definition of bounds on atan and π are presented in Listing 1. PVS developments are organized in theories, which are collections of mathematical and logical objects such as function definitions, variable declarations, axioms, and

lemmas. The `atan_approx` theory first imports the definition of the arctangent function. Then, it declares variables `n,x,px` of types `nat` (natural numbers), `real` (real numbers), and `posreal` (positive real numbers), respectively. For the scope of the theory, these variables are implicitly universally quantified. Though writing definitions, lemmas, theorems and specially proofs in PVS requires some training, reading theories is possible to anybody with a minimal background in logic.

D. Exponential

The series we use for the exponential function is

$$\exp(x) = \sum_{i=0}^{\infty} \frac{x^i}{i!}.$$

We could directly find bounds for negative x from this series as, in this case, the series is alternating. However, we will subsequently find that it is convenient to show that our bounds for the exponential function are strictly positive, and this is not true for all $x \leq 0$. Yet, this property *holds* for $-1 \leq x \leq 0$.

We define

$$\begin{aligned} \underline{\exp}(x,n) &= \sum_{i=0}^{2(n+1)+1} \frac{x^i}{i!}, \quad \text{if } -1 \leq x < 0, \\ \overline{\exp}(x,n) &= \sum_{i=0}^{2(n+1)} \frac{x^i}{i!}, \quad \text{if } -1 \leq x < 0. \end{aligned}$$

Using properties of the exponential function, we obtain bounds for the whole set of real numbers:

$$\begin{aligned} \underline{\exp}(0,n) &= \overline{\exp}(0,n) = 1, \\ \underline{\exp}(x,n) &= \underline{\exp}(\frac{x}{-\lfloor x \rfloor}, n)^{-\lfloor x \rfloor}, \quad \text{if } x < -1 \\ \underline{\exp}(x,n) &= \frac{1}{\overline{\exp}(-x,n)}, \quad \text{if } x > 0, \\ \overline{\exp}(x,n) &= \overline{\exp}(\frac{x}{-\lfloor x \rfloor}, n)^{-\lfloor x \rfloor}, \quad \text{if } x < -1 \\ \overline{\exp}(x,n) &= \frac{1}{\underline{\exp}(-x,n)}, \quad \text{if } x > 0. \end{aligned}$$

Notice that unless we can ensure that all of the bounding functions are strictly positive we will run into type-checking problems using the bound definitions for $x > 0$, *e.g.*, $1/\overline{\exp}(-x,n)$ is only defined provided $\overline{\exp}(-x,n) \neq 0$.

Proposition 6: $\forall x, n : 0 < \underline{\exp}(x,n) \leq \exp(x) \leq \overline{\exp}(x,n)$.

These are strict inequalities except when $x = 0$.

E. Natural Logarithm

For $0 < x \leq 1$, we use the alternating series for natural logarithm:

$$\ln(x+1) = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}.$$

PVS Listing 1 Definition of bounds on atan and π

```
atan_approx: THEORY
BEGIN

  IMPORTING atan

  n:  VAR nat
  x:  VAR real
  px: VAR posreal

  atan_pos_le1_ub(n,x): real =
    atan_series_n(x,2*n)

  atan_pos_le1_lb(n,x): real =
    atan_series_n(x,2*n+1)

  atan_pos_le1_bounds: LEMMA
    0 < x AND x <= 1 IMPLIES
      atan_pos_le1_lb(n,x) < atan(x) AND
      atan(x) < atan_pos_le1_ub(n,x)

  pi_lbn(n): posreal =
    4*(4*atan_pos_le1_lb(n,1/5) -
       atan_pos_le1_ub(n,1/239))

  pi_ubn(n): posreal =
    4*(4*atan_pos_le1_ub(n,1/5) -
       atan_pos_le1_lb(n,1/239))

  pi_bounds: THEOREM
    pi_lbn(n) < pi AND pi < pi_ubn(n)

  atan_pos_lb(n,px): real =
    IF px <= 1 THEN
      atan_pos_le1_lb(n,px)
    ELSE
      pi_lbn(n)/2 - atan_pos_le1_ub(n,1/px)
    ENDIF

  atan_pos_ub(n,px): real =
    IF px <= 1 THEN
      atan_pos_le1_ub(n,px)
    ELSE
      pi_ubn(n)/2 - atan_pos_le1_lb(n,1/px)
    ENDIF

  atan_lb(x,n): real =
    IF x > 0 THEN atan_pos_lb(n,x)
    ELSIF x = 0 THEN 0
    ELSE -atan_pos_ub(n,-x) ENDIF

  atan_ub(x,n): real =
    IF x > 0 THEN atan_pos_ub(n,x)
    ELSIF x = 0 THEN 0
    ELSE -atan_pos_lb(n,-x) ENDIF

  atan_bounds: THEOREM
    atan_lb(x,n) <= atan(x) AND
    atan(x) <= atan_ub(x,n)

END atan_approx
```

Therefore, we define

$$\underline{\ln}(x,n) = \sum_{i=1}^{2n}(-1)^{i+1}\frac{(x-1)^i}{i}, \quad \text{if } 1 < x \le 2,$$
$$\overline{\ln}(x,n) = \sum_{i=1}^{2n+1}(-1)^{i+1}\frac{(x-1)^i}{i}, \quad \text{if } 1 < x \le 2.$$

Using properties of the natural logarithm function, we obtain

$$\underline{\ln}(1,n) = \overline{\ln}(1,n) = 0$$
$$\underline{\ln}(x,n) = -\underline{\ln}(\frac{1}{x},n), \quad \text{if } 0 < x < 1,$$
$$\overline{\ln}(x,n) = -\overline{\ln}(\frac{1}{x},n), \quad \text{if } 0 < x < 1.$$

Finally, we extend the range to the whole set of positive reals. If $x > 2$, we find a natural number m and real number y such that $x = 2^m y$ and $1 < y \le 2$, by using the following recursive algorithm similar in spirit to Euclidean division:

```
lnnat(x:posreal,k:posnat): [nat,posreal] =
  if x < k then (0,x)
  else
    let (m,y) = lnnat(x/k,k) in
    (m+1,y)
  endif
```

We next prove the following property:

Proposition 7: $\forall x \ge 1, k > 1 : \; k^m \le x < k^{m+1}, y < k, x = k^m y$, where $(m,y) =$ `lnnat`(x,k).

If $(m,y) =$ `lnnat`$(2,x)$, we observe that

$$\ln(x) = \ln(2^m y) = m\ln(2) + \ln(y).$$

Hence,

$$\underline{\ln}(x,n) = m\,\underline{\ln}(2,n) + \underline{\ln}(y,n), \quad \text{if } x > 2,$$
$$\overline{\ln}(x,n) = m\,\overline{\ln}(2,n) + \overline{\ln}(y,n), \quad \text{if } x > 2.$$

Proposition 8: $\forall x > 0, n : \underline{\ln}(x,n) \le \ln(x) \le \overline{\ln}(x,n)$. These are strict inequalities except when $x = 1$.

III. Rational Interval Arithmetic

Interval arithmetic has been used for decades as a standard tool for numerical analysis on engineering applications [15], [16]. In interval arithmetic, operations are evaluated on range of numbers rather than on real numbers. A *(closed) interval* $[a,b]$ is the set of real numbers between a and b, *i.e.*,

$$[a,b] = \{x \mid a \le x \le b\}.$$

The bounds a and b are called the *lower bound* and *upper bound* of $[a,b]$, respectively. Note that if $a > b$, the interval is the empty set. The notation $[a]$ abbreviates the point-wise interval $[a,a]$.

Interval computations can be performed on the endpoints or on the center and the radius. For this work, we decided to work on rational endpoints. Trigonometric and transcendental functions for interval arithmetic are defined using the bounds presented in Section II.

Listing 2 shows a few definitions from the PVS theory `Interval`. Dots are used to simplify the presentation and hide some technical parts. The theory defines the type `Interval` as a record with fields `ub` and `lb` of type `rat` (rational numbers), variables `x,y` of type `real`, variable `n` of type `nat`, and variables `X,Y` of type `Interval`.

PVS Listing 2 Definition of interval arithmetic

```
Interval : THEORY
BEGIN

  Interval : TYPE = [#
              lb : rat,
              ub : rat
            #]

  x,y : VAR real
  n   : VAR nat
  X,Y : VAR Interval

  +(X,Y): Interval = [|lb(X)+lb(Y),
                       ub(X)+ub(Y)|]
  -(X,Y): Interval = [|lb(X)-ub(Y),
                       ub(X)-lb(Y)|]
  -(X)  : Interval = [|-ub(X),
                       -lb(X))|]
  *(X,Y): Interval = ...
  /(X,Y): Interval = X * [|1/ub(Y),
                           1/lb(Y)|]
  Abs(X): Interval = ...
  Sq(X) : Interval = ...
  ^(X,n): Interval = ...

  U(X,Y) : Interval = [|min(lb(X),lb(Y)),
                        max(ub(X),ub(Y))|]
  ...

END Interval
```

If `X` is a PVS interval, `lb(X)` is the lower bound and `ub(X)` is the upper bound of `X`. In PVS, we define the syntactic sugar `[|x,y|]` to represent the interval $[x,y]$. Interval union $\mathbf{x} \cup \mathbf{y}$, written in PVS `X U Y`, is defined as the smallest rational interval that contains both $\mathbf{x}$ and $\mathbf{y}$.

The four basic interval operations are defined as follows [17]:

$$\mathbf{x}+\mathbf{y} = [\underline{\mathbf{x}}+\underline{\mathbf{y}}, \overline{\mathbf{x}}+\overline{\mathbf{y}}],$$
$$\mathbf{x}-\mathbf{y} = [\underline{\mathbf{x}}-\overline{\mathbf{y}}, \overline{\mathbf{x}}-\underline{\mathbf{y}}],$$
$$\mathbf{x}\times\mathbf{y} = [\min\{\underline{\mathbf{x}}\underline{\mathbf{y}}, \underline{\mathbf{x}}\overline{\mathbf{y}}, \overline{\mathbf{x}}\underline{\mathbf{y}}, \overline{\mathbf{x}}\overline{\mathbf{y}}\}, \max\{\underline{\mathbf{x}}\underline{\mathbf{y}}, \underline{\mathbf{x}}\overline{\mathbf{y}}, \overline{\mathbf{x}}\underline{\mathbf{y}}, \overline{\mathbf{x}}\overline{\mathbf{y}}\}],$$
$$\mathbf{x}/\mathbf{y} = \mathbf{x} \times [\frac{1}{\overline{\mathbf{y}}}, \frac{1}{\underline{\mathbf{y}}}], \quad \text{if } \underline{\mathbf{y}}\overline{\mathbf{y}} > 0.$$

We also define the unary negation, absolute value, and power operators for intervals:

$$-\mathbf{x} = [-\overline{\mathbf{x}}, -\underline{\mathbf{x}}],$$
$$|\mathbf{x}| = [\min\{|\underline{\mathbf{x}}|, |\overline{\mathbf{x}}|\}, \max\{|\underline{\mathbf{x}}|, |\overline{\mathbf{x}}|\}], \quad \text{if } \underline{\mathbf{x}}\overline{\mathbf{x}} \ge 0.$$
$$|\mathbf{x}| = [0, \max\{|\underline{\mathbf{x}}|, |\overline{\mathbf{x}}|\}], \quad \text{if } \underline{\mathbf{x}}\overline{\mathbf{x}} < 0.$$
$$\mathbf{x}^n = \begin{cases} [1] & \text{if } n = 0, \\ [\underline{\mathbf{x}}^n, \overline{\mathbf{x}}^n] & \text{if } \underline{\mathbf{x}} \ge 0 \text{ or odd?}(n), \\ [\overline{\mathbf{x}}^n, \underline{\mathbf{x}}^n] & \text{if } \overline{\mathbf{x}} \le 0 \text{ and even?}(n), \\ [0, \max\{\underline{\mathbf{x}}^n, \overline{\mathbf{x}}^n\}] & \text{otherwise.} \end{cases}$$

Interval operations are defined such that they include the result of their corresponding real operations. This property is called the *inclusion property*.

Proposition 9 (Inclusion Property for Basic Operators): If $x \in \mathbf{x}$ and $y \in \mathbf{y}$ then $x \otimes y \in \mathbf{x} \otimes \mathbf{y}$, where $\otimes \in \{+, -, \times, /\}$. Moreover, $-x \in -\mathbf{x}$, $|x| \in |\mathbf{x}|$, and $x^n \in \mathbf{x}^n$, for $n \geq 0$. It is assumed that $\mathbf{y}$ does not contain 0 in the case of interval division.

Listing 3 specifies this property in PVS. The proposition $x \in \mathbf{x}$ is written `x ## X`.

PVS Listing 3 Basic inclusion properties

```
Add_inclusion : LEMMA
  x ## X AND y ## Y ⟹ x+y ## X+Y

Sub_inclusion : LEMMA
  x ## X AND y ## Y ⟹ x-y ## X-Y

Neg_inclusion : LEMMA
  x ## X ⟹ -x ## -X

Mult_inclusion : LEMMA
  x ## X AND y ## Y ⟹ x*y ## X*Y

Div_inclusion : LEMMA
  NOT 0 ## Y AND
  x ## X AND y ## Y ⟹ x/y ## X/Y

Abs_inclusion : LEMMA
  x ## X ⟹ abs(x) ## abs(X)

Sq_inclusion : LEMMA
  x ## X ⟹ sq(x) ## sq(X)

Pow_inclusion : LEMMA
  x ## X ⟹ x^n ## X^n
```

The inclusion property is fundamental to interval arithmetic. It guarantees that evaluations of an expression using interval arithmetic bound its exact real value. Any operation in interval arithmetic must satisfy the inclusion property with respect to its corresponding real operation.

A. Interval comparisons

There are several possible ways to compare intervals [18]. In this work, we use interval-rational comparisons and interval inclusions.

$$\begin{aligned} \mathbf{x} &< a \quad \text{if } \overline{\mathbf{x}} < a, \text{ similarly for } \leq, \\ \mathbf{x} &> a \quad \text{if } \underline{\mathbf{x}} > a, \text{ similarly for } \geq, \\ \mathbf{x} &\subseteq \mathbf{y} \quad \text{if } \underline{\mathbf{y}} \leq \underline{\mathbf{x}} \text{ and } \overline{\mathbf{x}} \leq \overline{\mathbf{y}}. \end{aligned}$$

Proposition 10: Assume that $x \in \mathbf{x}$,

1) if $\mathbf{x} \bowtie a$ then $x \bowtie a$, for $\bowtie \in \{<, \leq, >, \geq\}$, and
2) if $\mathbf{x} \subseteq \mathbf{y}$ then $x \in \mathbf{y}$.

We use $\not\bowtie$ to denote $\geq$, $>$, $\leq$, or $<$, when $\bowtie$ is, respectively, $<$, $\leq$, $>$, or $\geq$.

Proposition 11: If $\mathbf{x} \bowtie a$ and $\mathbf{x} \not\bowtie a$, then $\mathbf{x}$ is empty.

Notice that $\neg(\mathbf{x} \bowtie a)$ does not imply $\mathbf{x} \not\bowtie a$. For instance, $[-1, 1]$ is neither greater nor less than 0.

B. Square root, arctangent, exponential, and natural logarithm

Interval functions for square root, arctangent, π, exponential, and natural logarithm are defined for an approximation parameter $n \geq 0$:

$$\begin{aligned} [\sqrt{\mathbf{x}}]_n &= [\underline{\mathrm{sqrt}}(\underline{\mathbf{x}}, n), \overline{\mathrm{sqrt}}(\overline{\mathbf{x}}, n)], \quad \text{if } \mathbf{x} \geq 0, \\ [\mathrm{atan}(\mathbf{x})]_n &= [\underline{\mathrm{atan}}(\underline{\mathbf{x}}, n), \overline{\mathrm{atan}}(\overline{\mathbf{x}}, n)], \\ [\pi]_n &= [\underline{\pi}(n), \overline{\pi}(n)], \\ [\exp(\mathbf{x})]_n &= [\underline{\exp}(\underline{\mathbf{x}}, n), \overline{\exp}(\overline{\mathbf{x}}, n)], \\ [\ln(\mathbf{x})]_n &= [\underline{\ln}(\underline{\mathbf{x}}, n), \overline{\ln}(\overline{\mathbf{x}}, n)], \quad \text{if } \mathbf{x} > 0. \end{aligned}$$

As consequence of Propositions 1, 5, 6, and 8 in Section II, and the fact that these functions are increasing, the above functions satisfy the following inclusion property.

Proposition 12: For all n, if $x \in \mathbf{x}$ then $f(x) \in [f(\mathbf{x})]_n$, where $f \in \{\sqrt{\ }, \mathrm{atan}, \exp, \ln\}$. Moreover, $\pi \in [\pi]_n$. It is assumed that $\mathbf{x}$ is non-negative in the case of square root, and $\mathbf{x}$ is positive in the case of natural logarithm.

C. Trigonometric functions

Parametric functions for interval trigonometric functions are defined by cases analysis on quadrants where the functions are increasing or decreasing. The mathematical definitions are presented in Figure 1.

Note that $\sin$ and $\cos$ are defined for the whole real line. However, for angles α such that $|\alpha| > \underline{\pi}$ both functions will return the interval $[-1, 1]$, a valid bound but not a very good one. Furthermore, the expression $n + 5$ in Formula (8) is necessary to guarantee that lower and upper bounds of cosine are strictly positive in the interval $[-\frac{\underline{\pi}(n+5)}{2}, \frac{\underline{\pi}(n+5)}{2}]$, and thus, the interval tangent function is always defined in that interval.

The interval trigonometric functions satisfy the inclusion property.

Proposition 13: If $x \in \mathbf{x}$ then $f(x) \in [f(\mathbf{x})]_n$, where $f \in \{\sin, \cos\}$. Moreover, if $\mathbf{x} \subseteq [-\frac{\underline{\pi}(n+5)}{2}, \frac{\underline{\pi}(n+5)}{2}]$, $\tan(x) \in [\tan(\mathbf{x})]_n$.

The next section proposes a method to prove numerical propositions based on the interval arithmetic described here.

IV. Mechanical Proofs of Numerical Propositions

Arithmetic expressions are defined by the following grammar, where $\mathcal{V}$ is an denumerable set of real variables:

$$\begin{array}{lcl} e & ::= & a \mid x \mid e + e \mid e - e \mid -e \mid e \times e \mid \\ & & e/e \mid |e| \mid e^i \mid \sqrt{e} \mid \pi \mid \sin(e) \mid \\ & & \cos(e) \mid \tan(e) \mid \exp(e) \mid \ln(e) \mid \mathrm{atan}(e) \\ a & \in & \mathbb{Q} \\ i & \in & \mathbb{N} \\ x & \in & \mathcal{V} \end{array}$$

Numerical propositions P have either the form $e_1 \bowtie e_2$, where $\bowtie \in \{<, \leq, >, \geq\}$, or the form $e \in \mathbf{a}$, where $\mathbf{a}$ is a constant interval (an interval with constant rational endpoints). As usual, parentheses are used to group real and interval expressions as needed.

$$[\sin(\mathbf{x})]_n = \begin{cases} [\underline{\sin}(\underline{\mathbf{x}}, n), \overline{\sin}(\overline{\mathbf{x}}, n)] & \text{if} \quad \mathbf{x} \subseteq [-\frac{\underline{\pi}(n)}{2}, \frac{\underline{\pi}(n)}{2}], \\ [\underline{\sin}(\overline{\mathbf{x}}, n), \overline{\sin}(\underline{\mathbf{x}}, n)] & \text{else if} \quad \mathbf{x} \subseteq [\frac{\overline{\pi}(n)}{2}, \underline{\pi}(n)], \\ [\min\{\underline{\sin}(\underline{\mathbf{x}}, n), \underline{\sin}(\overline{\mathbf{x}}, n)\}, 1] & \text{else if} \quad \mathbf{x} \subseteq [0, \underline{\pi}(n)], \\ -[\sin(-\mathbf{x})]_n & \text{else if} \quad \mathbf{x} \subseteq [-\underline{\pi}(n), 0], \\ [-1, 1] & \text{otherwise}, \end{cases} \tag{6}$$

$$[\cos(\mathbf{x})]_n = \begin{cases} [\underline{\cos}(\overline{\mathbf{x}}, n), \overline{\cos}(\underline{\mathbf{x}}, n)] & \text{if} \quad \mathbf{x} \subseteq [0, \underline{\pi}(n)], \\ [\cos(-\mathbf{x})]_n & \text{else if} \quad \mathbf{x} \subseteq [-\underline{\pi}(n), 0], \\ [\min\{\underline{\cos}(\underline{\mathbf{x}}, n), \underline{\cos}(\overline{\mathbf{x}}, n)\}, 1] & \text{else if} \quad \mathbf{x} \subseteq [-\frac{\underline{\pi}(n)}{2}, \frac{\underline{\pi}(n)}{2}], \\ [-1, 1] & \text{otherwise}, \end{cases} \tag{7}$$

$$[\tan(\mathbf{x})]_n = [\frac{\underline{\sin}}{\overline{\cos}}(\underline{\mathbf{x}}, n+5), \frac{\overline{\sin}}{\underline{\cos}}(\overline{\mathbf{x}}, n+5)], \quad \text{if } \mathbf{x} \subseteq [-\frac{\underline{\pi}(n+5)}{2}, \frac{\underline{\pi}(n+5)}{2}]. \tag{8}$$

Fig. 1. Interval trigonometric functions

A *context* Γ is a set of hypotheses of the form $x \in \mathbf{x}$. A *ground context* is a context where all the intervals are constant. In the following, we use logical judgments in the sequent calculus style, *e.g.*, $\Gamma \vdash P$, where all free variables occurring in P are in Γ. The intended semantics of a judgment $\Gamma \vdash P$ is that the numerical proposition P is true under the hypotheses Γ.

Given a context Γ, an approximation parameter n, and an expression e, such that the free variables of e are in Γ, we define the interval expression $[e]_n^\Gamma$ by recursion on e.

$$\begin{aligned} [a]_n^\Gamma &= [a], \\ [x]_n^\Gamma &= \mathbf{x}, \text{ where } (x \in \mathbf{x}) \in \Gamma, \\ [e_1 \otimes e_2]_n^\Gamma &= [e_1]_n^\Gamma \otimes [e_2]_n^\Gamma, \quad \text{where } \otimes \in \{+, -, \times, /\}, \\ [e^i]_n^\Gamma &= ([e]_n^\Gamma)^i, \\ [-e]_n^\Gamma &= -[e]_n^\Gamma, \\ [|e|]_n^\Gamma &= |[e]_n^\Gamma|, \\ [\pi]_n^\Gamma &= [\pi]_n, \\ [f(x)]_n^\Gamma &= [f([x]_n^\Gamma)]_n, \\ & \quad \text{where } f \in \{\sin, \cos, \tan, \exp, \ln, \text{atan}\}. \end{aligned}$$

Theorem 1 (Inclusion): Let Γ be a context, n an approximation parameter, and e a well-defined arithmetic expression in Γ, *i.e.*, side conditions for division, square root, logarithm, and tangent are satisfied,

$$\Gamma \vdash e \in [e]_n^\Gamma. \tag{9}$$

Proof: By structural induction on e and propositions 4, 9, 12, and 13. ∎

A. A general method for numerical propositions

We propose a general method to prove numerical propositions. First, consider a judgment of the form

$$\Gamma \vdash e_1 \bowtie e_2,$$

where Γ is a ground context.

1) Select an approximation parameter n.
2) Define $e = e_1 - e_2$.
3) Evaluate $[e]_n^\Gamma \bowtie 0$. If it evaluates to true, the following judgment holds
$$\Gamma \vdash [e]_n^\Gamma \bowtie 0.$$
In that case go to step 5.
4) Evaluate $[e]_n^\Gamma \not\bowtie 0$. If this evaluates to true then fail. By Proposition 11, the judgment $\Gamma \vdash [e]_n^\Gamma \bowtie 0$ cannot hold. If $[e]_n^\Gamma \not\bowtie 0$ evaluates to false, increase the approximation parameter and return to step 3.
5) By Theorem 1,
$$\Gamma \vdash e \in [e]_n^\Gamma.$$
6) Proposition 10 yields
$$\Gamma \vdash e \bowtie 0.$$
7) By definition,
$$\Gamma \vdash e_1 - e_2 \bowtie 0.$$
8) Therefore,
$$\Gamma \vdash e_1 \bowtie e_2.$$

The method above can be easily adapted to judgments of the form $\Gamma \vdash e \bowtie \mathbf{a}$. In this case, the interval expression $[e]_n^\Gamma \subseteq \mathbf{a}$ is evaluated. If the expression evaluates to true, then the original judgment holds by Theorem 1 and Proposition 10. Otherwise, the method should fail.

The general method is *sound, i.e.*, all the steps can be effectively computed and each one is formally justified. In particular, the propositions $[e]_n^\Gamma \bowtie 0$, $[e]_n^\Gamma \not\bowtie 0$, and $[e]_n^\Gamma \subseteq \mathbf{a}$ can be mechanically computed as they only involve rational arithmetic and constant numerical values. The method is not *complete* as it does not necessarily terminate. Even if e only involves the four basic operations and no variables, it may be that both $[e]_n^\Gamma \bowtie 0$ and $[e]_n^\Gamma \not\bowtie 0$ evaluate to false.

The absence of a completeness result is a fundamental limitation on any general computable arithmetic. At a practical level, the problem arises because all we have available are a sequence of approximations to the real numbers x and y; provided x and y differ, with luck we will eventually have a pair of approximations whose intervals do not overlap, and hence we can return a result for $x \bowtie y$. However, if x and y

are the same real number (note we might not necessarily get the same sequence of approximations for both x and y), we can never be sure whether further evaluation might result in us being able to distinguish the numbers.

B. Dependency effect

The *dependency effect* is a well-known behavior of interval arithmetic due to the fact that interval identity is lost in interval evaluations. This may have surprising results, for instance $\mathbf{x} - \mathbf{x}$ is $[0]$ only if $\mathbf{x}$ is point-wise. Moreover, as we have seen in Section III-A, both $\mathbf{x} \geq a$ and $\mathbf{x} < a$ may be false. Additionally, interval arithmetic is subdistributive, *i.e.*, $\mathbf{x} \times (\mathbf{y} + \mathbf{z}) \subseteq \mathbf{x} \times \mathbf{y} + \mathbf{x} \times \mathbf{z}$. In the general case the inclusion is strict and some dependency effects appear as soon as a variable is used more than once in an expression.

For the method presented in Section IV-A, it means that the arrangement of the expression e matters. For instance, assume that we want to prove $x \in [0,1] \ \vdash \ 2 \times x \ \geq \ x$. This is pretty obvious in arithmetic as x is a non-negative real. Using our method, we first consider the arithmetic expression $e = 2 \times x - x$ and then construct the interval expression $[e]_n^\Gamma = 2 \times \mathbf{x} - \mathbf{x}$, where $\mathbf{x} = [0,1]$. For any approximation parameter n, $[e]_n^\Gamma$ evaluates to $[-1,2]$ which is neither greater nor less than 0. Therefore, the method will not terminate. On the other hand, if instead of the arithmetic expression $2 \times x - x$, we consider the equivalent arithmetic expression x, we have $[x]_n^\Gamma = [0,1]$ and $[0,1] \geq 0$ evaluates to true.

A second observation is that because of the dependency effect the width of intervals also matters. Consider again the expression $e = 2 \times x - x$. We have seen that the interval evaluation of $[e]_n^\Gamma$, for $x \in [0,1]$, results in $[-1,2]$, which is not sufficient to prove that $[e]_n^\Gamma \geq 0$. On the other hand, the expression $[e]_n^\Gamma$ evaluates to $[-1/2,1]$ when $x \in [0,1/2]$ and it evaluates to $[0,3/2]$ when $x \in [1/2,1]$. Therefore, we can prove that, for $x \in [0,1]$, $[e]_n^\Gamma \subseteq [-1/2,1] \cup [0,3/2]$, *i.e.*, $[e]_n^\Gamma \subseteq [-1/2,3/2]$, which is a better approximation than $[-1,2]$. If we continue dividing the interval $[0,1]$ and computing the union of the resulting intervals, we can eventually prove that $[e]_n^\Gamma + \epsilon \geq 0$ for an arbitrary small $\epsilon > 0$.

These observations lead to two enhancements of the general method. First, we could divide each interval in Γ before applying the general technique. Second, we may want to replace the original expression by an equivalent one that is less prone to the dependency effect.

C. Interval splitting

In interval arithmetic, the dependency effect of the union of the parts is less than the dependency effect of the whole. Indeed, the simplest way to reduce the dependency effect is to divide the interval variables into several tiles (subintervals) and to evaluate the original expression on these tiles separately. This technique is called *interval splitting* or *sub-paving* and is expressed by the following deduction rule.

Proposition 14: Let Γ be a context, e an expression whose free variables are x and those in Γ, $\mathbf{e}$ an interval expression, and $\mathbf{x}, \mathbf{x}_1, \ldots, \mathbf{x}_n$ intervals such that $\mathbf{x} = \bigcup_{1 \leq i \leq n} \mathbf{x}_i$,

$$\frac{\forall\ 1 \leq i \leq n :\ x \in \mathbf{x}_i, \Gamma \ \vdash \ e \in \mathbf{e}}{x \in \mathbf{x}, \Gamma \ \vdash \ e \in \mathbf{e}} \ \text{[Splitting]}$$

The Splitting rule can be iterated to obtain a splitting for multiple variables. Note that the number of tiles generated by interval splitting is exponential in the number of variables. Indeed, if k_1 is the number of tiles of the first variable alone, k_2 is the number of tiles of the second variables alone, and so forth, the total number of tiles to be considered for m variables is $\prod_{1 \leq j \leq m} k_j$.

The integration of the Splitting rule into the general method is straightforward. First, a splitting is computed for a given set of variables in Γ. Then, the general method is applied to all cases. If the general method is successful in all of them, by Proposition 14, the original judgment holds. Otherwise, the method fails and a new splitting may be considered.

D. Taylor Series Expansions

Replacing $2 \times x - x$ by x can be done automatically. In fact, as we will see in Section V, these kinds of simplifications are performed by our PVS implementation of the general method. However, these simplifications may not be sufficient even for simple expressions such as $x \times (1 - x)$, where $x \in [0.1]$. The subdistributivity property of interval arithmetic states that the interval evaluation of $x \times (1 - x)$ is better than that of the equivalent expression $x - x^2$. Unfortunately, that evaluation is not good enough to prove that $x \times (1 - x) \in [0, 1/4]$. In this case, as a domain expert knows, the optimal answer is obtained with the equivalent expression $1/4 - (1/2 - x)^2$. The solution is a lot less intuitive when non-algebraic functions are involved.

Taylor's theorem states that a n-differentiable function can be approximated near a given point by a polynomial of degree n whose coefficients depend on the derivatives of the function at that point. In interval arithmetic, taylor's theorem can be expressed by the following deduction rule.

Proposition 15: Let $\mathbf{x}, \mathbf{x}_0, \ldots, \mathbf{x}_n$ be strictly proper intervals, f a n-differentiable function on a variable $x \in \mathbf{x}$, and $c \in \mathbf{x}$ a constant,

$$\frac{\begin{array}{c}\forall\ 0 \leq i < n : \vdash \ f^{(i)}(c) \in \mathbf{x}_i \\ x \in \mathbf{x} \ \vdash \ f^{(n)}(x) \in \mathbf{x}_n\end{array}}{x \in \mathbf{x} \ \vdash \ f(x) \in \Sigma_{i=0}^{n} (\mathbf{x}_i \times (\mathbf{x} - c)^i)/i!} \ \text{[Taylor]}$$

The expression of Taylor's rule shows that interval $\mathbf{x}$ appears only once in each term of order i for i between 1 and $n-1$ preventing any dependency effect due to $\mathbf{x}$ in a term alone. The term of order n suffers some dependency effect as $\mathbf{x}$ also appears in the definition of $\mathbf{x}_n$. In most cases, $n = 2$ is used to cancel first order dependency effects as presented Listing 4. But in cases where the first derivatives nearly vanish or where the evaluation of the last derivative introduces significant dependency effects, we compute more terms to reach some better bounds.

Using Taylor's rule require more work than the Splitting rule. In particular, we need to provide intervals $\mathbf{x}_0, \ldots, \mathbf{x}_n$ and constant c that satisfy the hypotheses of the rule. For c we choose the middle point of $\mathbf{x}$ unless the user proposes another point. It follow immediately that $c \in \mathbf{x}$. For $0 \leq i < n$, we choose $\mathbf{x}_i = [f^{(i)}(c)]_n$ and, by Theorem 1, we have $f^{(i)}(c) \in \mathbf{x}_i$. Finally, we choose $\mathbf{x}_n = [f^{(n)}(x)]_n^\Gamma$, where Γ is the context $x \in \mathbf{x}$. By Theorem 1, we have $\Gamma \ \vdash \ f^{(n)}(x) \in \mathbf{x}_n$.

In order to prove the judgment $x \in \mathbf{x} \vdash f(x) \in \mathbf{a}$, we consider the interval expression $\Sigma_{i=0}^{n}(\mathbf{x}_i \times (\mathbf{x}-c)^i)/i! \subseteq \mathbf{a}$ for a given n. If it evaluates to true, then the original judgment holds by Taylor's rule and Proposition 10. If the evaluation returns false, the method fails and a higher expansion degree n may be considered.

For better results, the evaluation of $\Sigma_{i=0}^{n}(\mathbf{x}_i \times (\mathbf{x}-c)^i)/i! \subseteq \mathbf{a}$ can be performed using the splitting technique. Contrary to the approach described in [19], we do not have to generate a new taylor approximation for each tile. By using an interval-based taylor expansion, the same expression can be reused for all the tiles. One single global taylor expansion has to be validated, and the proofs for all the tiles simply consist in an interval evaluation of this expansion. We do not suffer from the taylor coefficients being irrational numbers, they are simply given by interval expressions involving rational functions. Relying on rational interval arithmetic leads to conceptually simpler proofs.

Section V describes how the general method and its extensions are implemented in the PVS theorem prover and illustrates the practical use of the library with a few examples.

V. Verified Real Number Calculations in PVS

The interval arithmetic presented in this paper has been developed as a PVS library called `Interval`. This library contains the specification of interval arithmetic described here and the formal proofs of its properties. We believe that a domain expert can use this library with a basic knowledge of theorem provers. Minimal PVS expertise is required as most of the technical burden of proving numerical properties is already implemented as proof strategies.

A. Strategies

The `numerical` strategy is the basic strategy that implements the general method and its extensions described in Section IV. For instance, Formula 1 can be specified in PVS as follows (comments in PVS start with the symbol `%` and extend to the end of the line):

```
g : posreal = 9.8        %[m/s^2]
v : posreal = 250*0.514  %[m/s]

tr35: LEMMA
  (g*tan(35*pi/180)/v) * 180/pi
    ## [| 3, 3.1 |]

%|- tr35: PROOF (numerical) QED
```

We emphasize that, in PVS, `tan` and `pi` are the real mathematical function $\tan$ and constant π, respectively. Lemma `tr35` is automatically discharged by the `numerical` strategy, which can be entered interactively or in batch mode, as in this case, via the ProofLite library developed by one of the authors [20].

Another example is the proof of the inequality 4.1.35 in [13]:

$$\forall x:\ 0 < x \leq 0.5828 \implies |\ln(1-x)| < \frac{3x}{2}.$$

The key to prove this inequality is to prove that the function

$$G(x) \quad = \quad \frac{3x}{2} - \ln(1-x)$$

satisfies $G(0.5828) > 0$. In PVS:

```
G(x|x < 1): real = 3*x/2 - ln(1-x)

A_and_S : lemma G(0.5828) > 0

%|- A_and_S : PROOF (numerical :defs "G") QED
```

In this case, the optional parameter `:defs "G"` tells `numerical` that the user-defined function `G` has to be expanded before performing the numerical evaluation. The original proof of this lemma in PVS required the manual expansion of 19 terms of the $\ln$ series.

The `numerical` strategy is aimed to practicality rather than completeness. In particular, it always terminate and it is configurable for better accuracy (at the expense of performance).

Termination is trivially achieved as the strategy does not iterate for different approximations, *i.e.*, step 3 either goes to step 5 or fails. In other words, if `numerical` does not succeed, it does nothing. Furthermore, `numerical` uses a default approximation parameter $n = 3$, which gives an accuracy of about 2 decimals for trigonometric functions. However, the user can increase this parameter or set a different approximation to each function according to his/her accuracy needs and availability of computational power. Currently, there is no direct relation between the approximation parameter and the accuracy, as all the bounding functions have different convergence rates. On-going work aims to provide, an absolute error of at most 2^{-p} for any expression with a new approximation parameter p. The strategy has not been designed to reuse past computations. Therefore, it will be prohibitively expensive to automatically iterate `numerical` to achieve a small approximation on a complex arithmetic expression.

In order to reduce the dependency effect, the `numerical` strategy automatically rearranges arithmetic expressions using a simple factorization algorithm. Due to the subdistributivity property, the evaluation of factorized interval expressions is more accurate than that of non-factorized ones. A set of lemmas of the NASA Langley PVS Libraries are also used as rewriting rules on arithmetic expressions prior to numerical evaluations. This set of lemmas is parameterizable and can be extended by the user. For instance, trigonometric functions applied to notable angles are automatically rewritten to their exact value. Therefore, `numerical` is able to prove that $\sin(\pi/2) \in \mathbf{1}$, even if this proposition is not provable using our interval arithmetic operators alone. Although it is not currently implemented, this approach can also be used to normalize angles to the range $[-\pi, \pi]$ that is suitable for the interval trigonometric functions in Sections III-C.

The splitting technique is implemented by allowing the user to specify the number of tiles to be considered for each interval variable or a default value for all of them. The strategy will evenly divide each interval. For example, the simple expression in Section IV-D can be proven to be in the range

$[0, 9/32]$ using a splitting of 16 subintervals.

```
  fair : LEMMA
    x ## [|0,1|] IMPLIES x*(1-x) ## [|0,9/32|]

%|- fair : PROOF (instint :splitting 16) QED
```

In this example we have used the `instint` strategy. This strategy is built on top of `numerical` and performs some basic logic manipulations such as introduction of real variables and interval constants. In this case, the proof command `(initint :splitting 16)` is equivalent to `(then (skeep) (numerical :vars ("x" "[|0,1|]" 16)))`. It instructs PVS to introduce the real variable `x` and then to apply `numerical` by splitting 16 times the interval $[0, 1]$.

The taylor series expansion technique is implemented in two steps. First, the `taylor` strategy automatically proves Proposition 15 for a particular function f and degree n. In the following example, we show that $x \in \mathbf{x} \vdash x \times (1 - x) \in \sum_{i=0}^{2}(\mathbf{x}_i \times (\mathbf{x} - c)^i)/i!$, provided that $\mathbf{x}$ is strictly proper.

```
  F(X)  : MACRO Interval = X*(1-X)
  DF(X) : MACRO Interval = 1 - 2*X
  D2F(X): MACRO Interval = [| -2 |]

  ftaylor : LEMMA
    x ## X AND StrictlyProper?(X) IMPLIES
    x*(1-x) ## Taylor2[X](F,DF,D2F)

%|- ftaylor : PROOF (taylor) QED
```

The keyword `MACRO` tells the theorem prover to automatically expand the definition of the function. The expression `Taylor2[X](F,DF,D2F)` corresponds to $\sum_{i=0}^{2}(\mathbf{x}_i \times (\mathbf{x} - c)^i)/i!$, where `F`, `DF`, and `D2F` are the interval functions corresponding to f, it 1st, and its 2nd derivative.

Finally, the strategy `instint` is called with the lemma `ftaylor`.

```
  best : LEMMA
    x ## [|0,1|] IMPLIES x*(1-x) ## [|0, 1/4|]

%|- best : PROOF
%|-   (instint :taylor "ftaylor")
%|- QED
```

B. A simple case study

The arctangent function is heavily used in aeronautic applications as it is fundamental to many Geodesic formulas[5]. One common implementation technique uses an approximation of the arctangent on the interval $\mathbf{x} = [-1/30, 1/30]$ after argument reduction [21]. For efficiency reasons, one may want to approximate the function $\mathrm{atan}(x)$ to single precision by the polynomial

$$r(x) \quad = \quad x - \frac{11184811}{33554432}x^3 - \frac{13421773}{67108864}x^5.$$

The coefficients of the polynomial approximation are stored exactly using IEEE single precision.

[5]See, for example, Ed William's Aviation Formulary at `http://williams.best.vwh.net/avform.htm`.

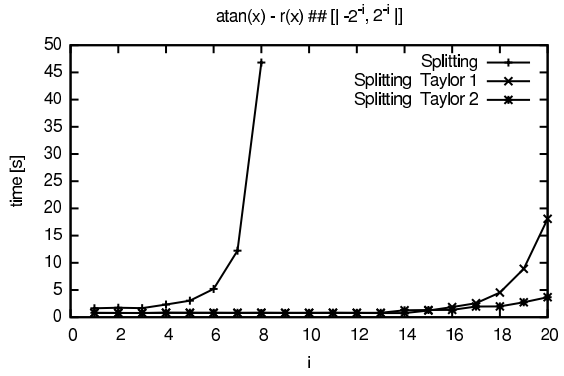

Fig. 2. Time required to prove $\tan(x) - r(x) \in [-1/30, 1/30]$

The objective of this case study is to show that

$$x \in [-1/30, 1/30] \ \vdash \ \mathrm{atan}(x) - r(x) \in [-2^{-i}, 2^{-i}],$$

for different values of i. The PVS specification of this problem for some values of i is presented in Listing 4. All the lemmas are automatically discharged by the `instint` strategy with different splitting and taylor's expansion degrees. As expected taylor's expansions and splitting get better results than splitting alone. Moreover, second degree expansions are almost always better than first degree expansions. This is not necessarily the case as illustrated by lemmas `fair_atan_t1_14` and `fair_atan_t2_14`: for $i = 14$, a first degree expansion with no splitting is enough to prove the property, while a second degree expansion requires a splitting of 2.

On a tile $\mathbf{t}$ of $\mathbf{x}$, the width of the error expression `E` that does not use taylor's theorem evaluated on $\mathbf{t}$ is larger than the sum of the width of expressions `Atan` and `R`. As the derivative of the arctangent is between 0.9989 and 1 on $\mathbf{x}$, we could expect that the width of `R` is at least twice the width of tile $\mathbf{t}$. Therefore, to obtain an error bound of $[-2^{-i}, 2^{-i}]$ we cannot use tiles larger than 2^{-i} and we will need at least $2^i/15 \approx 2^{i-1.4}$ tiles.

We use the same kind of simple calculation to show that since $|e'(x)| \leq 2.37 \cdot 10^{-6}$ we will need about $2^{i-14.8}$ tiles of width $2^{-i} \cdot 10^6/2.37$. This figures are accurate when we use second degree expansion but actual computations may require more tiles due to some dependency effects introduced when we use first degree expansions.

Figure 2 presents a summary of the time required to prove $\tan(x) - r(x) \in [-1/30, 1/30]$ for i in the range $[0, 20]$ using splitting, splitting and first degree taylor's expansion, and splitting and second degree taylor's expansion.

C. Implementation and Performance Issues

Actual definitions in PVS have been slightly modified for efficiency reasons. For instance, multiplication is defined using a case analysis on the sign of the operands. Additionally, all interval operations are completed by returning an empty interval if side conditions are not satisfied. This technique avoids some type correctness checks that are expensive.

PVS Listing 4 Accuracy of the arctangent approximation

```
fair_atan : THEORY
BEGIN

    x    : var   real
    r(x) : MACRO real = x - (11184811/33554432) * x^3 - (13421773/67108864) * x^5
    e(x) : MACRO real = atan(x) - r(x)
    Xt   : Interval   = [| -1/30, 1/30 |]

    fair_atan_8 : LEMMA  x ## Xt IMPLIES e(x) ## [|-2^-8, 2^-8|]
%|- fair_atan_8 : PROOF (instint :splitting 18) QED

    X     : var   Interval
    R(X)  : MACRO Interval = X - 11184811/33554432 * X^3 - 13421773/67108864 * X^5
    E(X)  : MACRO Interval = Atan(X,4) - R(X)
    DE(X) : MACRO Interval =
      1 / (1 + Sq(X)) - 1 + 3*(X^2*(11184811/33554432)) + 5*(X^4*(13421773/67108864))

    atan_taylor1 : LEMMA StrictlyProper?(X) AND x ## X IMPLIES e(x) ## Taylor1[X](E,DE)
%|- atan_taylor1 : PROOF (taylor) QED
    fair_atan_t1_14: LEMMA x ## Xt IMPLIES e(x) ## [|-2^-14, 2^-14|]
%|- fair_atan_t1_14 : PROOF (instint :taylor "atan_taylor1") QED
   fair_atan_t1_20: LEMMA x ## Xt IMPLIES e(x) ## [|-2^-20, 2^-20|]
%|- fair_atan_t1_20 : PROOF (instint :taylor "atan_taylor1" :splitting 13) QED

    D2E(X) : MACRO Interval =
      -2*X/Sq(1 + Sq(X)) + 20*(X^3*(13421773/67108864)) + 6*((11184811/33554432)*X)

    atan_taylor2 : LEMMA StrictlyProper?(X) AND x ## X IMPLIES e(x) ## Taylor2[X](E,DE,D2E)
%|- atan_taylor2 : PROOF (taylor) QED
    fair_atan_t2_14: LEMMA x ## Xt IMPLIES e(x) ## [|-2^-14, 2^-14|]
%|- fair_atan_t2_14 : PROOF (instint :taylor "atan_taylor2" :spitting 2) QED
    fair_atan_t2_20: LEMMA x ## Xt IMPLIES e(x) ## [|-2^-20, 2^-20|]
%|- fair_atan_t2_20 : PROOF (instint :taylor "atan_taylor2" :splitting 5) QED

END fair_atan
```

The strategies in this library work over the PVS built-in real numbers. The major advantage of this approach is that the functionality of the strategies can be extended to handle user defined real functions without modifying the strategy code. Indeed, optional parameters to the `numerical` strategy allow for the specification of arbitrary real functions. If the interval interpretations are not provided, the strategy tries to build them from the syntactic definition of the functions. The trade-off for the use of the PVS type `real`, in favor of a defined data type for arithmetic expressions, is that the function $[e]_n^\Gamma$ and Theorem 1 are at the meta-level, *i.e.*, they are not written in PVS. It also means that the soundness of our method cannot be proven in PVS itself. In particular, Theorem 1 has to be proven for each particular instance of e and $[e]_n^\Gamma$. This is not a major drawback as, in addition to `numerical`, we have developed a strategy called `inclusion` that discharges the sequent $\Gamma \vdash e \in [e]_n^\Gamma$ whenever is needed. PVS strategies are conservative in the sense that they do not add inconsistencies to the theorem prover. Therefore, if `numerical` succeeds to discharge a particular goal the answer is correct.

Finally, our method relies on explicit calculations to evaluate interval expressions. In theorem provers, explicit calculations usually means symbolic evaluations, which are extremely inefficient for the interval functions that we want to calculate. To avoid symbolic evaluations, `numerical` is implemented using computational reflection [22]–[24]. Interval expressions are translated to Common Lisp (the implementation language of PVS) and evaluated there. The extraction and evaluation mechanism is provided by the PVS ground evaluator [25]. The result of the evaluation is translated back to the PVS theorem prover using the PVSio library developed by one of the authors [26].

VI. Conclusion and Limits of Tractability

We have presented a pragmatic approach to verify ordinary real number computations in theorem provers. To this end, bounds for non-algebraic functions were established based on provable properties of their approximation series. Furthermore, a library for interval arithmetic was developed. The library includes strategies that automatically discharges numerical inequalities and interval inclusions.

The PVS Interval library contains 306 lemmas in total. It is roughly 10 thousand lines of specification and proofs and 1 thousand lines of strategy definitions. These numbers do not take into account the bounding functions, which have been fully integrated to the NASA Langley PVS Libraries. It is difficult to estimate the human effort for this development as it has evolved over the years from an original axiomatic specification to a fully foundational set of theories. As far

as we know, this is the most complete formalization within a theorem prover of an interval arithmetic that includes non-algebraic functions.

Research on interval analysis and exact arithmetic is rich and abundant (see for example [17], [27], [28]). The goal of interval analysis is to compute an upper bound of the round-off error in a computation performed using floating-point numbers. In contrast, in an exact arithmetic framework, an accuracy is specified at the beginning of the computation and the computation is performed in such way that the final result respects this accuracy.

Real numbers and exact arithmetic is also a subject of increasing interest in the theorem proving community. Pioneers in this area were Harrison and Gamboa who, independently, developed extensive formalizations of real numbers for HOL [4] and ACL2 [6]. In Coq, an axiomatic definition of reals is given in [7], and constructive definitions of reals are provided in [29] and [30]. As real numbers are built-in in PVS, there is not much meta-theoretical work on real numbers. However, a PVS library of real analysis was originally developed by Dutertre [31] and currently being maintained and extended as part of the NASA Langley PVS Libraries. An alternative real analysis library is proposed in [8].

Closer to our approach are the tools presented in [32] and [10]. These tools generate bounds on the round-off errors of numerical programs, and formal proofs that these bounds are correct. The formal proofs are proof scripts that can be checked off-line using a proof assistant.

Our approach is different from previous works in that we focus on automation and pragmatism. In simple words, our practical contribution is a correct pocket calculator for real number computations in formal proofs. Thanks to all the previous developments in theorem proving and real numbers, lemmas like Lemma `tr35` and Lemma `A_and_S` are provable in HOL, ACL2, Coq, or PVS. The Interval library make these proofs routine in PVS.

As in real life, users benefit in managing both a pocket calculator and a graphic tool. The fact that the example proposed in Section V-B is reaching the limits of tractability is not a problem. Our library aims at providing some simple tools that can be used seamlessly in proofs. Figure 3 would prompt a careful user that `fair_atan` theorems are a consequence of the fact that the derivative of the error is always positive. Such a fact could happen to be difficult to prove leading some one to prove that the error is bounded on some subintervals and that the derivative is always positive on some other subintervals. Anyways, such proofs will involve our library more than once.

We continue developing this library and it is currently being used to check numerical properties of aircraft navigation algorithms developed at the National Institute of Aerospace (NIA) and NASA. Future enhancements include:

- Development of a fully functional floating point arithmetic library [33] in order to generate guaranteed proofs of round-off-errors [32].
- Integration of this library and an exact arithmetic formalization in PVS developed by one of the authors [34].
- Implementation of latest developments on Taylor Models [35]–[37], which will enable a greater automation of the Taylor's series expansion technique.

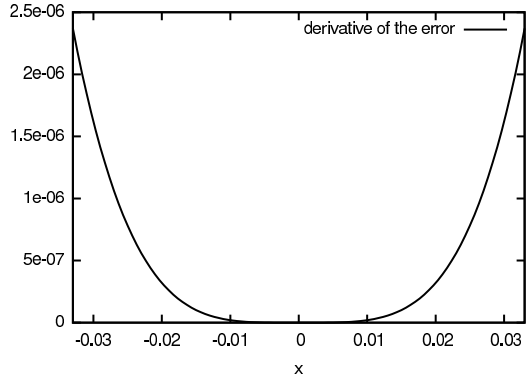

Fig. 3. Alternate `fair_atan` theorems will make use of interval arithmetic

REFERENCES

[1] Information Management and Technology Division, "Patriot missile defense: software problem led to system failure at Dhahran, Saudi Arabia," United States General Accounting Office, Report B-247094, 1992. [Online]. Available: http://www.fas.org/spp/starwars/gao/im92026.htm

[2] J.-L. Lions *et al.*, "Ariane 5 flight 501 failure report by the inquiry board," European Space Agency, Paris, France, Tech. Rep., 1996. [Online]. Available: http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf

[3] D. Gage and J. McCormick, "We did nothing wrong," *Baseline*, vol. 1, no. 28, pp. 32–58, 2004. [Online]. Available: http://common.ziffdavisinternet.com/download/0/2529/Baseline0304-DissectionNEW.pdf

[4] J. Harrison, *Theorem Proving with the Real Numbers*. Springer-Verlag, 1998.

[5] J. Fleuriot and L. Paulson, "Mechanizing nonstandard real analysis," *LMS Journal of Computation and Mathematics*, vol. 3, pp. 140–190, 2000. [Online]. Available: http://www.lms.ac.uk/jcm/3/lms1999-027/

[6] R. Gamboa, "Mechanically verifying real-valued algorithms in ACL2," Ph.D. dissertation, University of Texas at Austin, 1999. [Online]. Available: ftp://ftp.cs.utexas.edu/pub/boyer/diss/gamboa.pdf

[7] M. Mayero, "Formalisation et automatisation de preuves en analyse réelle et numérique," Ph.D. dissertation, Université Pierre et Marie Curie, Paris, France, 2001. [Online]. Available: http://www.pps.jussieu.fr/~mayero/specif/these-mayero.ps

[8] H. Gottliebsen, "Automated theorem proving for mathematics: real analysis in PVS," Ph.D. dissertation, University of St Andrews, 2001. [Online]. Available: http://www.dcs.qmul.ac.uk/~hago/thesis.ps.gz

[9] C. Muñoz, V. Carreño, G. Dowek, and R. Butler, "Formal verification of conflict detection algorithms," *International Journal on Software Tools for Technology Transfer*, vol. 4, no. 3, pp. 371–380, 2003. [Online]. Available: http://dx.doi.org/10.1007/s10009-002-0084-3

[10] M. Daumas, G. Melquiond, and C. Muñoz, "Guaranteed proofs using interval arithmetic," in *Proceedings of the 17th Symposium on Computer Arithmetic*, P. Montuschi and E. Schwarz, Eds., Cape Cod, Massachusetts, 2005, pp. 188–195. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00164621

[11] C. Muñoz and D. Lester, "Real number calculations and theorem proving," in *18th International Conference on Theorem Proving in Higher Order Logics*, Oxford, England, 2005, pp. 239–254. [Online]. Available: http://dx.doi.org/10.1007/11541868_13

[12] S. Owre, J. M. Rushby, and N. Shankar, "PVS: a prototype verification system," in *11th International Conference on Automated Deduction*, D. Kapur, Ed. Saratoga, New-York: Springer-Verlag, 1992, pp. 748–752. [Online]. Available: http://pvs.csl.sri.com/papers/cade92-pvs/cade92-pvs.ps

[13] M. Abramowitz and I. A. Stegun, Eds., *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover publications, 1972, ninth printing.

[14] J.-M. Muller, *Elementary functions, algorithms and implementation*. Birkhauser, 2006. [Online]. Available: http://www.springer.com/west/home/birkhauser/computer+science?SGWID=4-40353-22-72377986-0

[15] A. Neumaier, *Interval methods for systems of equations*. Cambridge University Press, 1990.
[16] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied interval analysis*. Springer, 2001. [Online]. Available: http://www.springeronline.com/sgw/cda/frontpage/0,10735,5-40106-22-2093571-0,00.html
[17] R. B. Kearfott, Ed., *Rigorous global search: continuous problemes*. Kluwer Academic Publishers, 1996.
[18] A. Yakovlev, "Classification approach to programming of localizational (interval) computations," *Interval Computations*, vol. 1, no. 1, pp. 61–84, 1992.
[19] J. Sawada, "Formal verification of divide and square root algorithms using series calculation," in *3rd International Workshop on the ACL2 Theorem Prover and its Applications*. University of Grenoble, 2002, pp. 31–49.
[20] C. Muñoz, "Batch proving and proof scripting in PVS," NIA-NASA Langley, National Institute of Aerospace, Hampton, VA, Report NIA Report No. 2007-03, NASA/CR-2007-214546, February 2007.
[21] P. Markstein, *IA-64 and elementary functions: speed and precision*. Prentice Hall, 2000.
[22] J. Harrison, "Metatheory and reflection in theorem proving: A survey and critique," SRI Cambridge, Millers Yard, Cambridge, UK, Technical Report CRC-053, 1995.
[23] S. Boutin, "Using reflection to build efficient and certified decision procedures," in *Proceedings of the Third International Symposium on Theoretical Aspects of Computer Software*, London, United Kingdom, 1997, pp. 515–529.
[24] F. W. von Henke, S. Pfab, H. Pfeifer, and H. Rueß, "Case Studies in Meta-Level Theorem Proving," in *Proc. Intl. Conf. on Theorem Proving in Higher Order Logics*, ser. Lecture Notes in Computer Science, J. Grundy and M. Newey, Eds., no. 1479. Springer-Verlag, Sept. 1998, pp. 461–478.
[25] N. Shankar, "Efficiently executing PVS," Computer Science Laboratory, SRI International, Menlo Park, CA, Project report, Nov. 1999, available at http://www.csl.sri.com/shankar/PVSeval.ps.gz.
[26] C. Muñoz, "Rapid prototyping in PVS," NIA-NASA Langley, National Institute of Aerospace, Hampton, VA, Report NIA Report No. 2003-03, NASA/CR-2003-212418, May 2003.
[27] P. Gowland and D. Lester, "A survey of exact arithmetic implementations," in *4th International Workshop on Computability and Complexity in Analysis*, Swansea, United Kingdom, 2000, pp. 30–47. [Online]. Available: http://www.link.springer.de/link/service/series/0558/bibs/2064/20640030.htm
[28] V. Mnissier-Morain, "Arbitrary precision real arithmetic: design and algorithms," *Journal of Logic and Algebraic Programming*, vol. 64, no. 1, pp. 13–39, 2005. [Online]. Available: http://dx.doi.org/10.1016/j.jlap.2004.07.003
[29] A. Ciaffaglione and P. Di Gianantonio, "A certified, corecursive implementation of exact real numbers," *Theoretical Computer Science*, vol. 351, no. 1, pp. 39–51, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2005.09.061
[30] J. Hughes and M. Niqui, "Admissible digit sets," *Theoretical Computer Science*, vol. 351, no. 1, pp. 61–73, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2005.09.059
[31] B. Dutertre, "Elements of mathematical analysis in PVS," in *Theorem Proving in Higher Order Logics: 9th International Conference. TPHOLs'97*, ser. Lecture Notes in Computer Science, J. von Wright, J. Grundy, , and J. Harrison, Eds., no. 1125. Turku, Finland: Springer-Verlag, August 1996, pp. 141–156. [Online]. Available: http://www.sdl.sri.com/papers/tphol96/
[32] M. Daumas and G. Melquiond, "Generating formally certified bounds on values and round-off errors," in *Real Numbers and Computers*, Dagstuhl, Germany, 2004, pp. 55–70. [Online]. Available: http://hal.inria.fr/inria-00070739
[33] S. Boldo and C. Muoz, "Provably faithful evaluation of polynomials," in *Proceedings of the 2006 ACM Symposium on Applied Computing*, Dijon, France, 2006, pp. 1328–1332. [Online]. Available: http://doi.acm.org/10.1145/1141277.1141586
[34] D. Lester and P. Gowland, "Using PVS to validate the algorithms of an exact arithmetic," *Theoretical Computer Science*, vol. 291, no. 2, pp. 203–218, Nov. 2002.
[35] K. Makino and M. Berz, "Taylor models and other validated functional inclusion methods," *International Journal of Pure and Applied Mathematics*, vol. 4, no. 4, pp. 379–456, 2003. [Online]. Available: http://bt.pa.msu.edu/pub/papers/TMIJPAM03/TMIJPAM03.pdf
[36] F. Cháves and M. Daumas, "A library to Taylor models for PVS automatic proof checker," in *Proceedings of the NSF workshop on reliable engineering computing*, Savannah, Georgia, 2006, pp. 39–52. [Online]. Available: http://www.gtsav.gatech.edu/workshop/rec06/papers/Chaves_paper.pdf
[37] F. Chves, M. Daumas, C. Muoz, and N. Revol, "Automatic strategies to evaluate formulas on Taylor models and generate proofs in PVS," in *6th International Congress on Industrial and Applied Mathematics*, Zurich, Switzerland, 2007. [Online]. Available: http://www.iciam07.ch/