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.

1 Introduction

Deadly and disastrous failures [1], [2], [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 nonalgebraic functions. Despite all of the developments concerning real analysis in theorem provers [4], [5], [6], [7], [8], the formal verification of the correctness of these calculations is not routine.

Take, for example, the formula

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

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 degree is about 3 degree/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] 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 (1), within a proof assistant. The point of departure is a collection of lower and upper bounds for rational and nonrational 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 nonalgebraic 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] and [11]. The rest of this document is organized as follows: Section 2 defines bounds for elementary functions. Section 3 presents a rational interval arithmetic based on these bounds. Section 4 describes a method to prove numerical propositions. The implementation of this method in a theorem prover is described in Section 5. Section 6 summarizes our work and compares it to related work.

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

• *M. Daumas is with the Laboratoire Electronique, Informatique, Automatique et Systèmes (ELIAUS), Université of Perpignan Via Domitia (UPVD), 52, avenue Paul Alduy, F-66860 Perpignan Cedex, France. E-mail: marc.daumas@ens-lyon.org.*
• *D. Lester is with the School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK. E-mail: dlester@cs.man.ac.uk.*
• *C. Muñoz is with the National Institute of Aerospace, 144 Research Drive, Hampton, VA 23666. E-mail: Munoz@NIANet.Org.*

Manuscript received 22 July 2007; revised 25 Mar. 2008; accepted 18 Sept. 2008; published online 11 Nov. 2008.
Recommended for acceptance by P. Kornerup, P. Montuschi, J.-M. Muller, and E. Schwarz.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-2007-07-0362.
Digital Object Identifier no. 10.1109/TC.2008.213.

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 the authors.[4] In this paper, we identify as *Propositions* those theorems that have been formalized in PVS. Only one theorem is identified as a *Meta-Theorem* since it was not verified in PVS. Although it can be mathematically proved at the metalevel, the formal proof is not needed in our development as it can be replaced by a proof rule that mechanically discharges any particular instance of the theorem.

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.

2 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 (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 quadrant localization for the trigonometric functions [13] and we return a trivial interval if the bounds cannot be located in the four primary quadrants since the convergence of Taylor series is usually best for small values. We performed multiplicative range reduction by a factor 2^m for the logarithm function. Additive range reduction is not automatically applied to the trigonometric functions as it would use a parameterized approximation of π. Elaborate range reduction techniques [14] would significantly enhance the speed and the accuracy of the functions defined in Sections 2 and 3.

All the stated propositions in this section have been formally verified in the verification system PVS.

2.1 Square Root

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

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

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 $x > 0$.

2.2 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. $\forall x,\ n : \underline{\cos}(x, n) \leq \cos(x) \leq \overline{\cos}(x, n)$.

Propositions 2 and 3 hold for any value of x, including $|x| \geq \pi/2$. However, the smaller the value of x, the better the accuracy of the approximations. We could have implemented a range reduction to the interval $[-\pi/2, \pi/2]$. Since this interval is not rational, extra work is needed in order to parameterize efficiently a range reduction using rational arithmetic. Such reduction is planned as a future improvement.

2.3 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 naively wish to use to define $\pi/4$ and hence π) the series: $1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$

2. PVS is available from http://pvs.csl.sri.com.
3. http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html.
4. http://research.nianet.org/ munoz/Interval.

does converge, but very slowly. Instead, we use Machin's Formula $\frac{\pi}{4} = 4\,\mathrm{atan}(1/5) - \mathrm{atan}(1/239)$, that has much better convergence properties [15]. 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. $\forall n : \underline{\pi}(n) \le \pi \le \overline{\pi}(n)$.

Now, using properties of arctangent, we extend the range of the function to the whole set of real numbers:

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

Proposition 5. $\forall x,\ n : \underline{\mathrm{atan}}(x,n) \le \mathrm{atan}(x) \le \overline{\mathrm{atan}}(x,n)$.

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

2.4 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 $x \le 0$. Yet, this property *holds* for $-1 \le x \le 0$.

We define

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

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

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

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

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

2.5 Natural Logarithm

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

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

Therefore, we define

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

Using properties of the natural logarithm function, we obtain

$$\begin{aligned}\underline{\ln}(1,n) &= \overline{\ln}(1,n) = 0,\\ \underline{\ln}(x,n) &= -\underline{\ln}\left(\frac{1}{x},n\right), \quad \text{if } 0 < x < 1,\\ \overline{\ln}(x,n) &= -\overline{\ln}\left(\frac{1}{x},n\right), \quad \text{if } 0 < x < 1.\end{aligned}$$

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* $(m, y) = \mathtt{lnnat}(x, k)$.

If $(m, y) = \mathtt{lnnat}(2, x)$, we observe that

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

Hence

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

Proposition 8. $\forall x > 0,\ n : \underline{\ln}(x,n) \le \ln(x) \le \overline{\ln}(x,n)$.

These are strict inequalities except when $x = 1$.

3 Rational Interval Arithmetic

Interval arithmetic has been used for decades as a standard tool for numerical analysis on engineering applications [16],

PVS Listing 1 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 =
    if    X >= 0 AND Y >= 0 then pXp(X,Y)
    elsif X >= 0 AND Y <= 0 then pXn(X,Y)
    elsif X >= 0            then pXm(X,Y)
    elsif X <= 0 AND Y <= 0 then nXn(X,Y)
    elsif X <= 0 AND Y >= 0 then nXp(X,Y)
    elsif X <= 0            then nXm(X,Y)
    elsif Y >= 0            then mXp(X,Y)
    elsif Y <= 0            then mXn(X,Y)
    else                         mXm(X,Y)
    endif
  pXp(X,Y): Interval = [|lb(X)*lb(Y),
                          ub(X)*ub(Y)|]
  /(X,Y):   Interval = X * [|1/ub(Y),
                            1/lb(Y)|]
  Abs(X):   Interval = ...
  ^(X,n):   Interval = ...

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

  ##(x,X): bool = lb(X) <= x AND x <= ub(X)
  Proper?(X): bool = lb(X) <= ub(X)
  StrictlyProper?(X): bool = lb(X) < ub(X)
  ...

END Interval
```

Fig. 1. Definition of interval arithmetic.

[17]. 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.

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

The bounds a and b are called the *lower bound* and *upper bound* of $[a, b]$, respectively. We say that the interval $[a, b]$ is *proper* if $a \leq b$. Furthermore, it is *strictly proper* if $a < b$. Note that a nonproper interval is equivalent to the empty set. The notation $[a]$ abbreviates the point-wise interval $[a, a]$.

Interval computations can be performed on the end-points 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 2.

Fig. 1 shows a few definitions from the PVS theory `Interval`. PVS developments are organized in theories, which are collections of mathematical and logical objects such as function definitions, variable declarations, axioms, and lemmas. 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`. For the scope of the theory, these variables are implicitly universally quantified. Dots are used to emphasis on the fact that some parts of the theory have been removed to simplify the presentation and hide some technical points. 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 and functional programming.

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 $\mathbf{y}$. Furthermore, the inclusion $x \in \mathbf{x}$ is written in PVS `x ## X`. This notation is not very intuitive but the set of infix operators available in PVS is quite limited. In particular, the more natural keyword `in` is already reserved.

The four basic interval operations are defined as follows [18]:

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

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

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

Interval operations are defined such that they include the result of their corresponding real operations. This property is called the *inclusion property*. As illustrated in Fig. 2, Proposition 9 corresponds to several lemmas in PVS, one per operator.

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.*

The inclusion property is fundamental to interval arithmetic. It guarantees that evaluations of an expression using interval arithmetic bound its exact real value. Any

PVS Listing 2 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)

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

Fig. 2. Basic inclusion properties.

operation in interval arithmetic must satisfy the inclusion property with respect to its corresponding real operation.

3.1 Interval Comparisons

There are several possible ways to compare intervals [19]. 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 $\mathbf{x}\ \mathcal{R}\ a$, then $x\ \mathcal{R}\ a$, for $\mathcal{R} \in \{<, \leq, >, \geq\}$, and*
2. *if $\mathbf{x} \subseteq \mathbf{y}$, then $x \in \mathbf{y}$.*

We use $\mathcal{R}^{-1}$ to denote $\geq$, $>$, $\leq$, or $<$, when $\mathcal{R}$ is, respectively, $<$, $\leq$, $>$, or $\geq$.

Proposition 11. *If $\mathbf{x}\ \mathcal{R}\ a$ and $\mathbf{x}\ \mathcal{R}^{-1}\ a$, then $\mathbf{x}$ is empty. Notice that $\neg(\mathbf{x}\ \mathcal{R}\ a)$ does not imply $\mathbf{x}\ \mathcal{R}^{-1}\ a$. For instance, $[-1, 1]$ is neither greater nor less than 0.*

As in the case of Proposition 9, Propositions 10 and 11 actually correspond to several lemmas in PVS, one per each order relation.

3.2 Square Root, Arctangent, Exponential, and Natural Logarithm

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

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

As consequence of Propositions 1, 5, 6, and 8 in Section 2, and the fact that these functions are increasing, the above functions satisfy the following inclusion property that corresponds to several lemmas in PVS, one per function.

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 nonnegative in the case of square root, and $\mathbf{x}$ is positive in the case of natural logarithm.*

3.3 Trigonometric Functions

Parametric functions for interval trigonometric functions are defined by case analysis on quadrants where the functions are increasing or decreasing. The mathematical definitions are presented in Fig. 3.

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 (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 $\mathbf{x} \subseteq [-\frac{\underline{\pi}(n+5)}{2}, \frac{\underline{\pi}(n+5)}{2}]$, $\tan(x) \in [\tan(\mathbf{x})]_n$.*

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

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

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

Fig. 3. Interval trigonometric functions.

As in previous cases, this proposition corresponds to several lemmas in PVS, one per trigonometric function. We will see in Section 5 that the fact that all the previous propositions do not appear in the PVS theory as we reported them does not reduce the strength of the results computed and prove by our strategies.

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

4 Mechanical Proofs of Numerical Propositions

This section describes the method we propose to prove numerical propositions via interval arithmetic. The implementation of this method as a set of PVS strategies will be described in Section 5.

We consider the set of *arithmetic expressions* defined by the following grammar, where $\mathcal{V}$ is a denumerable set of real variables:

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

Numerical propositions P have either the form $e_1 \mathrel{\mathcal{R}} e_2$, where $\mathcal{R} \in \{<, \leq, >, \geq\}$, 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.

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 inductively define the interval expression $[e]_n^\Gamma$:

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

Meta-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. □

Since real and interval expressions are not reflected as abstract data types in the PVS specification language, the function $[e]_n^\Gamma$, and therefore, Meta-Theorem 1, are provided at the metalevel. This is not a major drawback as PVS provides an expressive proof strategy language where the function $[e]_n^\Gamma$ can be easily defined by structural induction on e and Meta-Theorem 1 can be implemented as a proof rule.

4.1 A General Method for Numerical Propositions

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

$$\Gamma \vdash e_1 \mathrel{\mathcal{R}} e_2,$$

where Γ is a ground context:

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

The method above can be easily adapted to judgments of the form $\Gamma \vdash e \mathrel{\mathcal{R}} \mathbf{a}$. 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 Meta-Theorem 1 and Proposition 10. Otherwise, the method should fail.

The general method is *sound*. In particular, all evaluations can be effectively performed and each step is logically justified. For instance, propositions $[e]_n^\Gamma \mathcal{R}\, 0$, $[e]_n^\Gamma / \mathcal{R}\, 0$, and $[e]_n^\Gamma \subseteq \mathbf{a}$ can be mechanically computed as they only involve rational arithmetic and constant numerical values. On the other hand, 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 $[e]_n^\Gamma \mathcal{R}\, 0$ and $[e]_n^\Gamma \mathcal{R}^{-1}\, 0$ 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 $x \mathcal{R} y$. 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.

4.2 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 3.1, 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 4.1, 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 nonnegative real. Using our method, we first consider the arithmetic expression $e = 2 \times x - x$ and then construct the interval expression $[e]^{\Gamma}_{n} = 2 \times \mathbf{x} - \mathbf{x}$, where $\mathbf{x} = [0,1]$. For any approximation parameter n, $[e]^{\Gamma}_{n}$ 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 $[x]^{\Gamma}_{n} = [0,1]$ 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 $[e]^{\Gamma}_{n}$, for $x \in [0,1]$, results in $[-1,2]$, which is not sufficient to prove that $[e]^{\Gamma}_{n} \geq 0$. On the other hand, the expression $[e]^{\Gamma}_{n}$ 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 $x \in [0,1]$, $[e]^{\Gamma}_{n} \subseteq [-1/2,1] \cup [0,3/2]$, i.e., $[e]^{\Gamma}_{n} \subseteq [-1/2,3/2]$, 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 $[e]^{\Gamma}_{n} + \epsilon \geq 0$ for an arbitrary small $\epsilon > 0$.

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

4.3 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 *paving* and is expressed by the following proposition, written as a 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 integration of the rule Splitting into the general method is straightforward. Consider the judgment $\Gamma \vdash P$. First, a paving of size n is generated for a given variable $\mathbf{x}$ in Γ, e.g., $\mathbf{x_1}, \ldots \mathbf{x_n}$ such that their union is $\mathbf{x}$. Then, the original method is applied to each one of the n judgments $\Gamma_i \vdash P$, where $\Gamma_i = \Gamma \setminus \{x \in \mathbf{x}\} \cup \{x \in \mathbf{x_i}\}$. If the general method is successful in all of them, by Theorem 14, the original judgment holds. Otherwise, the method fails and a paving of size $n+1$ is considered.

Notice that the rule Splitting can be iterated over multiple variables. However, the method is computationally inefficient for multivariable pavings. Indeed, the number of tiles generated by interval splitting is exponential in the number of variables. For instance, 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 $\prod_{1 \leq j \leq m} k_j$.

4.4 Taylor Series Expansions

Replacing $2 \times x - x$ by x can be done automatically. In fact, as we will see in Section 5, 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 nonalgebraic functions are involved. The idea here is to automatically replace the original expression, seen as a function on one variable, by its Taylor's expansion. The rationale for this replacement is that, Taylor's expansion removes any first-order dependency effect in relation to the variable provided all the occurrences of the variable are explicit. Higher-order dependency effects decrease geometrically with the order of Taylor's expansion as soon as the radius is less than center in the interval used for the variable.

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 proposition, written as a 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:*

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

PVS Listing 3 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
```

Fig. 4. Accuracy of the arctangent approximation.

In the rule above, the 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 in Fig. 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.

From a practical point of view, the rule Taylor requires more work than the rule Splitting. 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 follows immediately that $c \in \mathbf{x}$. For $0 \leq i < n$, we choose $\mathbf{x}_i = [f^{(i)}(c)]_n$ and, by Meta-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 Meta-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 the rule Taylor and Proposition 10. If the evaluation returns false, the method fails and the expansion degree $n+1$ is 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 [20], 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 of 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 5 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.

5 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.

5.1 Strategies

The strategy numerical is the basic strategy that implements the general method and its extensions described in Section 4. For instance, (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 numerical, which can be entered interactively or in batch mode, as in this case, via the ProofLite library developed by one of the authors [21].

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

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

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

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

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 strategy numerical is aimed to practicality rather than completeness. In particular, it always terminates 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 two 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. Ongoing 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 strategy numerical 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 nonfactorized 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 parameterized 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 3.3.

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 4.4 can be proved to be in the range $[0, 9/32]$ using a paving of 16 tiles.

```
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 strategy instint. This strategy is built on top of numerical and performs some basic logic manipulations such as introduction of real variables and interval constants. The strategy instructs PVS to introduce the real variable x and then to apply numerical by using a paving of 16 tiles on the interval [0, 1].

The Taylor's series expansion technique is implemented in two steps. First, the strategy taylor automatically applies Proposition 15 to 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, and its first and second derivative.

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

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

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

5.2 Implementation and Performance Issues

Actual definitions in PVS have been slightly modified for technical reasons. For instance, interval operations that may return an unbounded interval such as the reciprocal or the tangent are completed by returning an empty interval if side conditions are not satisfied. This technique avoids the generation of type correctness conditions (TCCs) during the application of strategies. TCCs are sometimes difficult to handle within a strategy as the strategy developer has little control on how, where, and when those TCCs are generated. By using this technique, those conditions do not simply disappear, they will show as premises to be discharged by the user once the strategies have finished.

The strategies in this library use the PVS built-in real numbers. Technically, we do not provide a deep embedding [22] of real or interval expressions, i.e., real and interval expression are not reflected in the PVS specification language as abstract data types. 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 `numerical` allow for the specification of arbitrary real functions. The trade-off for the use of the PVS type `real`, in favor of a defined data type for arithmetic expressions, is that Meta-Theorem 1 cannot be specified nor verified inside PVS. However, PVS provides an expressive strategy language where this theorem can be implemented as a proof rule. More precisely, the proof of this lemma is expressed as a strategy, namely `inclusion`, that mechanically discharges each particular instance of the theorem. PVS strategies are conservative in the sense that they do not add inconsistencies to the theorem prover. Therefore, if `inclusion` succeeds to discharge a particular goal, the answer is sound.

Finally, our method relies on the ability of the theorem prover to evaluate rational interval arithmetic. Usually, these calculations are performed using symbolic evaluation, which can be extremely inefficient for the interval functions that we want to calculate. The strategy `numerical` allows the user to evaluate rational expressions using computational reflection [23], [24], [25]. In this case, PVS expressions involving rational functions are first translated into Common Lisp (the implementation language of PVS) using the extraction mechanism provided by the PVS ground evaluator [26]. They are evaluated by the Common Lisp engine, which relies on the native language implementation of big numbers. We emphasize that only rational arithmetic is involved in this evaluation. The result of the evaluation is translated back into the PVS theorem prover using the PVSio library developed by one of the authors [27]. Of course, the result is as sound as the forth and back translations between PVS and Lisp, and the Common Lisp engine. In any case, the user has always the option of a purely symbolic evaluation of rational arithmetic.

5.3 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 [28]. For efficiency reasons, one may want to approximate the function $\mathrm{atan}(x)$ to single precision by the polynomial

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

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

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 Fig. 4. All the lemmas are automatically discharged by the strategy `instint` 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 $2^{-i} \cdot 10^6/2.37$. These 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.

Fig. 5 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. It shows that the simple calculation performed in the previous paragraphs is correct. It is too early for a comparison with any standard interval library as properties that are formally proved in a matter of minutes with our library are evaluated almost instantaneously with floating-point intervals and most properties that are checked in a few minutes with floating-point intervals use interval techniques that are still not available in our library.

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

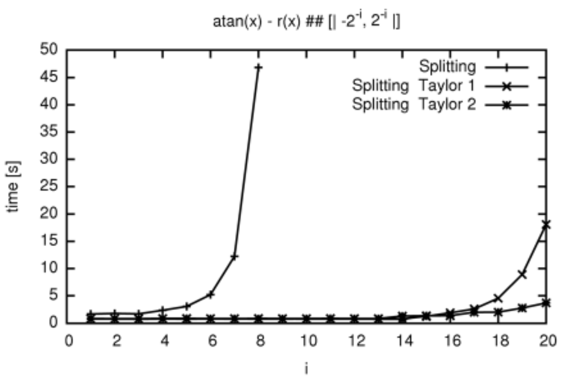

Fig. 5. Time required to prove $\tan(x) - r(x) \in [-1/30, 1/30]$.

6 Conclusion

We have presented a pragmatic approach to verify common real number computations in theorem provers. To this end, bounds for nonalgebraic 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 discharge numerical inequalities and interval inclusions.

The PVS Interval library contains 306 lemmas in total. It is roughly 10,000 lines of specification and proofs and 1,000 lines of strategy definitions. These numbers do not take into account the bounding functions, which have been integrated into 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 nonalgebraic functions.

Research on interval analysis and exact arithmetic is rich and abundant (see for example, [18], [29], and [30]). One goal of interval analysis is to bound 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 [31], an axiomatic definition of reals is given in [7], and constructive definitions of reals are provided in [32] and [33]. The goal of Section 2 could be attained by very different means using coinductive streams [34] on a formal system that handles them efficiently. As real numbers are built-in in PVS, there is not much metatheoretical work on real numbers. However, a PVS library of real analysis was originally developed by Dutertre [35] 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 [36] and [37]. The first one generates 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 offline using a proof assistant. Zumkeller presented a formalization of Taylor's Models in Coq using real interval arithmetic [37]. His final goal is similar to ours: to provide formal proofs of numerical nonlinear inequalities. However, the two approaches are different. Zumkeller's work is motivated by the Flyspeck Project,[6] which aims to formalize Hales' proof of Kepler's Conjecture. That proof requires the solution of highly complex nonlinear inequalities. Hence, precision, performance, and scalability are the main requirements for that task. Our goal is in some sense more modest. We focus on the mechanization and automation of proofs of *routine* numerically properties in theorem provers. These properties are typically introduced as axioms after they have been validated using a pocket calculator. We argue that this practice is error prone and we propose a set of strategies that safely performs these computations. In order to do that, we formalize several ad-hoc techniques that have been used in interval analysis for decades but that, to our knowledge, have never been formalized before. The strategies presented in [10] have been completely redesigned for this work in order to provide a higher degree of automation in their default mode. Furthermore, the redesigned strategies allow the user to parameterize them for either better accuracy or better performance as needed.

Another benefit of this work is that it can be easily replicated in a different theorem prover. Our interval library only requires rational arithmetic. Our strategies can be implemented in any strategy language that provides a mechanism to access the syntactical structure of arithmetic expressions. Other developments focus on getting better performances or precision by taking advantage of advanced features such as, for example, efficient exact real arithmetic. Theoretical and practical advances on theorem prover are necessary to handle complex problems, but this work shows that modern theorem provers, such as PVS, already provides the basic capabilities to solve routine problems in a practical way.

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 [38] in order to generate guaranteed proofs of round-off-errors [36],
- integration of this library and an exact arithmetic formalization in PVS developed by one of the authors [39], and
- implementation of latest developments on Taylor Models [40], [41], [42], which will enable a greater automation of the Taylor's series expansion technique.

Acknowledgments

Marc Daumas is supported in part by the PICS 2533 of the NRS and by the EVA-Flo project of the French ANR. David Lester is supported in part by the French Région

6. http://www.lix.polytechnique.fr/~zumkeller/Flyspeck.html.

Languedoc-Roussillon and the University of Perpignan. César Muñoz is supported in part by the US National Aeronautics and Space Administration under NASA Cooperative Agreement NCC-1-02043 and by the University of Perpignan.

References

[1] Information Management and Technology Division, *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia,* US General Accounting Office, Report B-247094, http://www.fas.org/spp/starwars/gao/im92026.htm, 1992.

[2] J.-L. Lions et al., "Ariane 5 Flight 501 Failure Report by the Inquiry Board," technical report, European Space Agency, http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf, 1996.

[3] D. Gage and J. McCormick, "We Did Nothing Wrong," *Baseline,* vol. 1, no. 28, pp. 32-58, http://common.ziffdavisinternet.com/download/0/2529/Baseline0304-DissectionNEW.pdf, 2004.

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

[5] J. Fleuriot and L. Paulson, "Mechanizing Nonstandard Real Analysis," *LMS J. Computation and Math.,* vol. 3, pp. 140-190, http://www.lms.ac.uk/jcm/3/lms1999-027/, 2000.

[6] R. Gamboa, "Mechanically Verifying Real-Valued Algorithms in ACL2," PhD dissertation, Univ. of Texas at Austin, ftp://ftp.cs.utexas.edu/pub/boyer/diss/gamboa.pdf, 1999.

[7] M. Mayero, "Formalisation et Automatisation de Preuves en Analyse Réelle et Numérique," PhD dissertation, Université Pierre et Marie Curie, http://www.pps.jussieu.fr/~mayero/specif/these-mayero.ps, 2001.

[8] H. Gottliebsen, "Automated Theorem Proving for Mathematics: Real Analysis in PVS," PhD dissertation, Univ. of St. Andrews, http://www.dcs.qmul.ac.uk/~hago/thesis.ps.gz, 2001.

[9] C. Muñoz, V. Carreño, G. Dowek, and R. Butler, "Formal Verification of Conflict Detection Algorithms," *Int'l J. Software Tools for Technology Transfer,* vol. 4, no. 3, pp. 371-380, http://dx.doi.org/10.1007/s10009-002-0084-3, 2003.

[10] M. Daumas, G. Melquiond, and C. Muñoz, "Guaranteed Proofs Using Interval Arithmetic," *Proc. 17th IEEE Symp. Computer Arithmetic (ARITH '05),* P. Montuschi and E. Schwarz, eds., pp. 188-195, http://hal.archives-ouvertes.fr/hal-00164621, 2005.

[11] C. Muñoz and D. Lester, "Real Number Calculations and Theorem Proving," *Proc. 18th Int'l Conf. Theorem Proving in Higher Order Logics (TPHOLs '05),* pp. 239-254, http://dx.doi.org/10.1007/11541868_13, 2005.

[12] S. Owre, J.M. Rushby, and N. Shankar, "PVS: A Prototype Verification System," *Proc. 11th Int'l Conf. Automated Deduction (CADE '92),* D. Kapur, ed., pp. 748-752, http://pvs.csl.sri.com/papers/cade92-pvs/cade92-pvs.ps, 1992.

[13] M. Abramowitz and I.A. Stegun, eds., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover Publications, 1972.

[14] J.-M. Muller, *Elementary Functions, Algorithms and Implementation.* Birkhaüser, http://www.springer.com/west/home/birkhauser/computer+science?SGWID=4-40353-22-72377986-0, 2006.

[15] *Mathematics by Experiment: Plausible Reasoning in the 21st Century,* J. Borwein and D.H. Bailey, eds. A.K. Peters, 2003.

[16] A. Neumaier, *Interval Methods for Systems of Equations.* Cambridge Univ. Press, 1990.

[17] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis.* Springer, http://www.springeronline.com/sgw/cda/frontpage/0, 10735, 5-40106-22-2093571-0, 00.html, 2001.

[18] *Rigorous Global Search: Continuous Problems,* R.B. Kearfott, ed. Kluwer Academic Publishers, 1996.

[19] A. Yakovlev, "Classification Approach to Programming of Localizational (Interval) Computations," *Interval Computations,* vol. 1, no. 1, pp. 61-84, 1992.

[20] J. Sawada, "Formal Verification of Divide and Square Root Algorithms Using Series Calculation," *Proc. Third Int'l Workshop ACL2 Theorem Prover and Its Applications (ACL2 '02),* pp. 31-49, 2002.

[21] C. Muñoz, "Batch Proving and Proof Scripting in PVS," Report NIA report no. 2007-03, NASA/CR-2007-214546, NIA-NASA Langley, Nat'l Inst. Aerospace, Hampton, Va., Feb. 2007.

[22] R. Boulton, A. Gordon, M. Gordon, J. Harrison, J. Herbert, and J.V. Tassel, "Experience with Embedding Hardware Description Languages in HOL," *Proc. IFIP TC10/WG 10.2 Int'l Conf. Theorem Provers in Circuit Design (TPCD '92),* pp. 129-156, 1992.

[23] J. Harrison, "Metatheory and Reflection in Theorem Proving: A Survey and Critique," Technical Report CRC-053, SRI Cambridge, Millers Yard, Cambridge, U.K., 1995.

[24] S. Boutin, "Using Reflection to Build Efficient and Certified Decision Procedures," *Proc. Third Int'l Symp. Theoretical Aspects of Computer Software (TACS '97),* pp. 515-529, 1997.

[25] F.W. von Henke, S. Pfab, H. Pfeifer, and H. Rueß, "Case Studies in Meta-Level Theorem Proving," *Proc. 11th Int'l Conf. Theorem Proving in Higher Order Logics (TPHOLs '98),* J. Grundy and M. Newey, eds., pp. 461-478, Sept. 1998.

[26] N. Shankar, "Efficiently Executing PVS," Computer Science Laboratory, SRI Int'l, Menlo Park, CA, project report, http://www.csl.sri.com/shankar/PVSeval.ps.gz, Nov. 1999.

[27] C. Muñoz, "Rapid Prototyping in PVS," Report NIA Report 2003-03, NASA/CR-2003-212418, NIA-NASA Langley, Nat'l Inst. Aerospace, Hampton, Va., May 2003.

[28] P. Markstein, *IA-64 and Elementary Functions: Speed and Precision.* Prentice Hall, 2000.

[29] P. Gowland and D. Lester, "A Survey of Exact Arithmetic Implementations," *Proc. Fourth Int'l Workshop Computability and Complexity in Analysis (CCA '00),* pp. 30-47, http://www.link.springer.de/link/service/series/0558/bibs/2064/20640030.htm, 2000.

[30] V. Ménissier-Morain, "Arbitrary Precision Real Arithmetic: Design and Algorithms," *J. Logic and Algebraic Programming,* vol. 64, no. 1, pp. 13-39, http://dx.doi.org/10.1016/j.jlap.2004.07.003, 2005.

[31] G. Huet, G. Kahn, and C. Paulin-Mohring, *The Coq Proof Assistant: A Tutorial: Version 8.0,* ftp://ftp.inria.fr/INRIA/coq/current/doc/Tutorial.pdf.gz, 2004.

[32] A. Ciaffaglione and P. Di Gianantonio, "A Certified, Corecursive Implementation of Exact Real Numbers," *Theoretical Computer Science,* vol. 351, no. 1, pp. 39-51, http://dx.doi.org/10.1016/j.tcs.2005.09.061, 2006.

[33] J. Hughes and M. Niqui, "Admissible Digit Sets," *Theoretical Computer Science,* vol. 351, no. 1, pp. 61-73, http://dx.doi.org/10.1016/j.tcs.2005.09.059, 2006.

[34] Y. Bertot, "Affine Functions and Series with Co-Inductive Real Numbers," *Math. Structures in Computer Science,* vol. 17, no. 1, pp. 37-63, http://dx.doi.org/10.1017/S0960129506005809, 2007.

[35] B. Dutertre, "Elements of Mathematical Analysis in PVS," *Proc. Ninth Int'l Conf. Theorem Proving in Higher Order Logics (TPHOLs '96),* J. von Wright, J. Grundy, and J. Harrison, eds., pp. 141-156, http://www.sdl.sri.com/papers/tphol96/, Aug. 1996.

[36] M. Daumas and G. Melquiond, "Generating Formally Certified Bounds on Values and Round-Off Errors," *Real Numbers and Computers,* pp. 55-70, http://hal.inria.fr/inria-00070739, 2004.

[37] R. Zumkeller, "Formal Global Optimisation with Taylor Models," *Proc. Third Int'l Joint Conf. Automated Reasoning (IJCAR '06),* U. Furbach and N. Shankar, eds., pp. 408-422, http://dx.doi.org/10.1007/11814771_35, 2006.

[38] S. Boldo and C. Muñoz, "Provably Faithful Evaluation of Polynomials," *Proc. 21st ACM Symp. Applied Computing (SAC '06),* pp. 1328-1332, http://doi.acm.org/10.1145/1141277.1141586,2006.

[39] 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.

[40] K. Makino and M. Berz, "Taylor Models and Other Validated Functional Inclusion Methods," *Int'l J. Pure and Applied Math.,* vol. 4, no. 4, pp. 379-456, http://bt.pa.msu.edu/pub/papers/TMIJPAM03/TMIJPAM03.pdf, 2003.

[41] F. Cháves and M. Daumas, "A Library to Taylor Models for PVS Automatic Proof Checker," *Proc. NSF Workshop Reliable Eng. Computing (REC '06),* pp. 39-52, http://www.gtsav.gatech.edu/workshop/rec06/papers/Chaves_paper.pdf, 2006.

[42] F. Cháves, M. Daumas, C. Muñoz, and N. Revol, "Automatic Strategies to Evaluate Formulas on Taylor Models and Generate Proofs in PVS," *Proc. Sixth Int'l Congress on Industrial and Applied Math. (ICIAM '07),* http://www.iciam07.ch/, 2007.