Real Number Calculations and Theorem Proving*

César Muñoz[1] and David Lester[2]

[1] National Institute of Aerospace, 144 Research Drive, Hampton VA 23666, USA
munoz@nianet.org
[2] University of Manchester, Oxford Road, Manchester M13 9PL, UK
dlester@cs.man.ac.uk

Abstract. Wouldn't it be nice to be able to *conveniently* use ordinary real number expressions within proof assistants? In this paper we outline how this can be done within a theorem proving framework. First, we formally establish upper and lower bounds for trigonometric and transcendental functions. Then, based on these bounds, we develop a rational interval arithmetic where real number calculations can be performed in an algebraic setting. This pragmatic approach has been implemented as a strategy in PVS. The strategy provides a safe way to perform explicit calculations over real numbers in formal proofs.

1 Introduction

In the verification of an engineering application it is often necessary to perform explicit calculations on non-algebraic functions. Despite all the developments on real analysis in theorem provers [7–9, 12, 15], formal justification of these calculations is not routine.

Take, for example, the formula

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

where g is the gravitational force and $v = 250$ kt is the ground speed of an aircraft. This formula appears in the verification of the NASA's Airborne Information for Lateral Spacing (AILS) algorithm presented in [18]. It states that the maximum turn rate of an aircraft flying at ground speed v with a bank angle of 35^o is 3^o per second. The original proof is about a dozen lines and requires the use of several trigonometric properties.

In some cases the formal checking of a numerical inequality is so cumbersome that the effort seems futile; it is then tempting to perform the calculation out of the system, and introduce the result as an axiom[3]. However, the chances are

* Many thanks to Behzad Akbarpour for pointing out errors in the formulas for exponential in the formal publication [19].

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

that the external calculation will be performed using floating-point arithmetic. Without formal checking of the result, we will never be sure of the correctness of the calculation.

In this paper we present a method to automatically prove numerical inequalities, 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 used here are well-known. However, to our knowledge, this is the most complete formalization of exact real arithmetic and interval arithmetic within a theorem prover.

The rest of this paper is organized as follows. Section 2 defines bounds for square root and transcendental functions. Section 3 presents a rational interval arithmetic based on these bounds. Section 4 describes a method to prove numerical inequalities. The implementation of this method in PVS is described in Section 5. Last section summarizes our work and compares it to related work. For readability, we will use standard mathematical notation along this paper. However, we remark that the mathematical development presented in this paper has been written and fully verified in PVS. Furthermore, all the development is freely available on the Internet. The results on upper and lower bounds have been integrated to the NASA Langley PVS Libraries at `http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html`. The rational interval arithmetic and the PVS strategy for numerical inequalities are available from `http://research.nianet.org/~munoz/Interval`.

2 Bounds for Square Root and Transcendental Functions

A PVS basic theory of bounds for square root and trigonometric functions was originally proposed for the verification of an algorithm for aircraft conflict detection [18]. It has been completed 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 be improved, as much as needed, by increasing the approximation parameter n.

For transcendental functions, we use Taylor's approximation series. Because the convergence is usually best for a small range of values, we have used Taylor's Theorem only on a small range, and then exploited the technique of *range*

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

2.1 Square root

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

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

Proposition 1. $\forall x, n: \; x \geq 0 \;\Rightarrow\; 0 \;\leq\; \underline{\text{sqrt}}(x, n) \;\leq\; \sqrt{x} \;<\; \overline{\text{sqrt}}(x, n)$.

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. $\forall x, n: \; \underline{f}(x, n) \;\leq\; f(x) \;\leq\; \overline{f}(x, n)$, *for* $f \in \{\sin, \cos\}$.

2.3 Arctangent and π

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

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

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 $\frac{\pi}{4} = 4\ \mathrm{atan}(1/5) - \mathrm{atan}(1/239)$, which then has much better convergence properties. Using this identity we can define bounds on π:

$$\underline{\pi}(n) = 16\ \underline{\mathrm{atan}}(1, n) - 4\ \overline{\mathrm{atan}}(1, n),$$
$$\overline{\pi}(n) = 16\ \overline{\mathrm{atan}}(1, n) - 4\ \underline{\mathrm{atan}}(1, n).$$

Proposition 3. $\forall n :\ \underline{\pi}(n)\ \leq\ \pi\ \leq\ \overline{\pi}(n)$.

Now, we extend the range of the arctangent 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 4. $\forall x, n :\ \underline{\mathrm{atan}}(x, n)\ \leq\ \mathrm{atan}(x)\ \leq\ \overline{\mathrm{atan}}(x, n)$.

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

2.4 Exponential

The fundamental 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 with the above bounds this is not true. It *is* true for $-1 \leq x \leq 0$. This is proven by using Taylor's Theorem for the exponential function on the range $-1 \leq x < 0$.

We define

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

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

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

2.5 Natural Logarithm

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

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

Therefore, we define

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

Using properties of the natural logarithm function, we obtain

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

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 $1 < y \leq 2$, by using an algorithm similar to Euclidean division. Then, 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 6. $\forall x, n :\ 0 < x \ \Rightarrow\ \underline{\ln}(x, n) \ \leq\ \ln(x) \ \leq\ \overline{\ln}(x, n)$.

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

3 Rational Interval Arithmetic

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. Note that if $a > b$, the interval is empty. The notation $[a]$ abbreviates the point-wise interval $[a, a]$.

Since interval computations are mostly performed on the bounds of the intervals, the set where these bounds are defined is critical. This set is called the *base number type*. Systems for interval analysis and exact arithmetic usually consider the base number type to be machine floating-point numbers. In general, interval computations assume a correct implementation of the IEEE 754 standard [13]. For this work, we take a different approach where the base number type is the set of rational numbers. Trigonometric and transcendental functions for interval arithmetic are defined using the parameterizable bounding functions presented in Section 2.

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. If $\mathbf{x}$ is an interval variable, $\underline{\mathbf{x}}$ denotes its lower bound and $\overline{\mathbf{x}}$ denotes its upper bound. The four basic interval operations are defined as follows [14]:

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

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* and is formally expressed as follows:

Proposition 7. *If $x \in \mathbf{x}$ and $y \in \mathbf{y}$ then $x \otimes y \in \mathbf{x} \otimes \mathbf{y}$, where $\otimes \in \{+, -, \times, \div\}$. 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 the evaluations of an expression using interval arithmetic is a correct approximation of the exact real value. Any operation in interval arithmetic must satisfy the inclusion property with respect to the corresponding real operation.

3.1 Interval comparisons

There are several possible ways to compare intervals [28]. In this work, we use interval-rational comparisons and interval inclusions.

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

Proposition 8. *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 9. *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.

3.2 Square root, arctangent, exponential, and natural logarithm

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

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

The above functions satisfy the following inclusion property.

Proposition 10. *If $x \in \mathbf{x}$ then $f(x) \in [f(\mathbf{x})]_n$, where $f \in \{\sqrt{}, \mathrm{atan}, \exp, \ln\}$. 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.*

Proof. This is a consequence of Propositions 1, 4, 5, and 6 in Section 2, and the fact that these functions are increasing. □

3.3 Trigonometric functions

Parametric trigonometric functions for intervals are defined as follows:

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

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

$$[\tan(\mathbf{x})]_n = [\frac{\underline{\sin}}{\overline{\cos}}(\underline{\mathbf{x}}, n+3), \frac{\overline{\sin}}{\underline{\cos}}(\overline{\mathbf{x}}, n+3)], \quad \text{if } \mathbf{x} \subseteq [-\frac{\underline{\pi}(n)}{2}, \frac{\underline{\pi}(n)}{2}]. \tag{8}$$

The $n + 3$ in Formula (8) is necessary to guarantee that the lower and upper bounds of cosine are always positive. Therefore, the tangent function is well-defined in the interval $[-\frac{\underline{\pi}(n)}{2}, \frac{\underline{\pi}(n)}{2}]$.

The above functions satisfy the following inclusion property.

Proposition 11. *If $x \in \mathbf{x}$ then $f(x) \in [f(\mathbf{x})]_n$, where $f \in \{\sin, \cos\}$. Moreover, if $\mathbf{x} \subseteq [-\frac{\underline{\pi}(n)}{2}, \frac{\underline{\pi}(n)}{2}]$, $\tan(x) \in [\tan(\mathbf{x})]_n$.*

Proof. This is a consequence of Proposition 2 in Section 2, and a case analysis on the quadrant where the functions are increasing or decreasing. □

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

4 Proving Numerical Inequalities

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

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

As usual, parenthesis are used to group subexpressions as needed.

A *context* Γ is a set of couples $(x, \mathbf{x})$ where x is a real variable and $\mathbf{x}$ is a non-empty interval. The intended semantics of contexts is given by the following deduction rule in the sequent calculus style:

$$\frac{(x, \mathbf{x}) \in \Gamma}{\Gamma \ \vdash \ x \in \mathbf{x}}.$$

Given a real expression e, a context Γ that includes all variables in e, and an approximation parameter n, the interval expression $[e]_n^\Gamma$ is recursively defined as follows:

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

Proposition 12. *Let e be an arithmetic expression, Γ be a context that includes all variables in e, n an approximation parameter, and $\mathbf{e} = [e]_n^\Gamma$. Assume that e and $\mathbf{e}$ are well-defined, i.e., side conditions are satisfied for division, square root, logarithm, and tangent for real and interval values. Therefore,*

$$\Gamma \ \vdash \ e \in \mathbf{e}. \tag{9}$$

Proof. By structural induction on e and propositions 3, 7, 10, and 11.

4.1 A general method

We propose the following general method to prove the sequent

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

where e_1 and e_2 are well-defined arithmetic expressions, and $\bowtie \in \{<, \leq, >, \geq\}$:

1. Select an approximation parameter n.
2. Define $e = e_1 - e_2$.
3. Define $\mathbf{e} = [e]_n^{\Gamma}$ and show that it is well-defined.
4. By Proposition 12,

$$\Gamma \vdash e \in \mathbf{e}.$$

5. Evaluate $\mathbf{e} \bowtie 0$. If it evaluates to true, it means that the following sequent holds

$$\Gamma \vdash \mathbf{e} \bowtie 0.$$

   In that case go to step 6. In the other case, evaluate $\mathbf{e} \not\bowtie 0$. If this evaluates to true then fail. By Proposition 9, the sequent $\Gamma \vdash \mathbf{e} \bowtie 0$ cannot hold. If $\mathbf{e} \not\bowtie 0$ evaluates to false, increase the approximation parameter and return to step 3.
6. Proposition 8 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 is a *sound*, i.e., all the steps can be effectively computed and each one is formally justified. In particular, well-definedness of $\mathbf{e}$ and the inequalities $\mathbf{e} \bowtie 0$ and $\mathbf{e} \not\bowtie 0$ can be mechanically checked as they only involve rational arithmetic. However, the method is not *complete* as the it does not necessarily terminate. Even if e only involves the four basic operations, it may be that both $\mathbf{e} \bowtie 0$ and $\mathbf{e} \not\bowtie 0$ 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. Theoreticians might prefer the statement that to be computable, a function must at least be (computably) continuous, and that any attempt to define non-constant continuous functions from the (computable) reals to the booleans is futile [22, 23, 25, 26].

4.2 Sub-distributive arithmetic

Interval arithmetic is sub-distributive, 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. This effect is also called *decorrelation* and it is due to the fact that interval identity is lost in interval arithmetic. This may have surprising effects, for instance $\mathbf{x} - \mathbf{x}$ is $[0]$ only if $\mathbf{x}$ is point-wise, e.g., $[0,1] - [0,1] = [-1,1]$. Moreover, as we have seen in Section 3.1, both $\mathbf{x} \geq a$ and $\mathbf{x} < a$ may be false.

For the method presented in the previous section, it means that the arrangement of the expression e matters. For instance, assume that we want to prove $\Gamma \vdash 2 \times x \geq x$ assuming that $x \in [0,1]$ is in Γ. This is pretty obvious in arithmetic as x is a non-negative real. Using our method, we consider the arithmetic expression $e = 2 \times x - x$ and construct the interval expression $\mathbf{e} = 2 \times \mathbf{x} - \mathbf{x}$, where $\mathbf{x} = [0,1]$. For any approximation parameter, $\mathbf{e}$ evaluates to $[-1,2]$ which is neither greater nor less than 0. Therefore, the method will not terminate. This effect may happen even if e is ground.

On the other hand, if instead of the arithmetic expression $2 \times x - x$, we consider the equivalent arithmetic expression $(2-1) \times x$, the corresponding interval property $([2] - [1]) \times \mathbf{x}$ evaluates to $[0,1]$ which is non-negative.

4.3 The bounds of π

Note that sin and cos are defined for the whole real line. However, for angles α such that $|\alpha| \geq \pi$ both functions will return the interval $[-1,1]$, a valid approximation but not a very good one.

Even for angles less than π, the bounds computed by formulas (6) and (7) may not be very accurate. For example, consider the arithmetic expression $e = \sin(\frac{\pi}{2})$ and, for an approximation parameter n, the corresponding interval expression $\mathbf{e} = [\sin([\frac{\underline{\pi}(n)}{2}, \frac{\overline{\pi}(n)}{2}])]_n$. From the definition of $[\sin(\mathbf{x})]_n$, we get $\mathbf{e} = [-1,1]$, as the interval $[\frac{\underline{\pi}(n)}{2}, \frac{\overline{\pi}(n)}{2}]$ falls in the default case. Therefore, the fact that $\sin(\frac{\pi}{2}) > 0$, cannot be proven using our method.

4.4 Symbolic evaluation

Our method relies on explicit calculations to check for well-definedness of interval expressions in step 3 and to verify interval inequalities in step 5. In theorem provers, explicit calculations usually means symbolic evaluations, which are extremely inefficient for the interval functions that we want to calculate.

Section 5 describes how all these issues are handled in the PVS Interval package that we have developed.

5 Automation in PVS

The interval arithmetic presented in this paper has been formalized and verified in PVS [21]. It is available as a PVS package called Interval. The strategy

`numerical`, which is part of the Interval package, implements the method described in Section 4.1. This strategy automatically discharges sequent of the forms $\Gamma \vdash e_1 \bowtie e_2$, for $\bowtie \in \{<, \leq, >, \geq\}$, and $\Gamma \vdash e \in [a, b]$.

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 the generation of type correctness conditions in some instances.

The strategy `numerical` is aimed to practicality rather than accuracy. For example, it might not be able to prove that $\sqrt{4} \in [2]$, but it can prove that $\sqrt{4} \in [1.5, 2.5]$, or, even better, that $\sqrt{4} \in [1.99, 2.01]$. With this in mind, we designed a strategy that:

- Always terminates (in a reasonable period of time).
- Works over the PVS built-in type `real` (in contrast to a strategy for a new data type of arithmetic expressions).
- Is configurable for better accuracy (at the expense of performance).

5.1 Termination

Termination is trivially achieved as the strategy does not iterate for different approximations, i.e., step 5 either goes to step 6 or fails. In other words, if `numerical` does not succeed, it does nothing. By default, `numerical` picks a default approximation value of 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 its accuracy needs and availability of computational power. Currently, there is no direct relation between the approximation parameter and the degree of the accuracy, as all the bounding functions have different convergence rates. On-going work aims to provide an absolute error of at most 10^{-n} for the approximation parameter n. This will give additional control to the user on the accuracy of the result. However, this technique will not guarantee the precision of the final result as computation errors are accumulative. Automatic iteration of the method for continuously increasing approximation parameters is not supported as the strategies have not being designed to reuse past computations. Without a reusing mechanism it will be prohibitively expensive to automatically iterate `numerical` to achieve a small approximation on a complex arithmetic expression.

5.2 Data type for real numbers

The strategies in the Interval package 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 operators and functions without modifying the strategy code. Indeed, optional parameters to the strategy `numerical` 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 Proposition 12 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, Proposition 12 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 `sharp` that discharges the sequent $\Gamma \vdash e \in [e]_n^\Gamma$ whenever is needed. We assume that 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.

5.3 Decorrelation

Decorrelation is a well-known problem in interval arithmetic. Daumas et al. describe in [5] additional strategies in the Interval package that address this problem. Those strategies, which are intended for verification of numerical algorithms, are computationally intensive and not suitable for interactive theorem proving. In contrast, the strategy `numerical` uses two basic methods to reduce decorrelation. First, it automatically rearranges arithmetic expressions using a simple factorization algorithm. Due to the sub-distributivity property, factorized interval expressions are likely to be more accurate than non-factorized ones. Second, a configurable parameter allows the user to specify a splitting parameter for interval sub-divisions. This technique is described in detail in [5]. The naïve implementation of interval sub-divisions in `numerical` is exponential with respect to the number of interval variables.

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. Although is not currently implemented, this approach can also be used to normalize angles to the range $-\pi \ldots \pi$ that is suitable for the trigonometric bounding functions in Sections 3.3.

5.4 Numerical evaluations

To avoid symbolic evaluations, `numerical` is implemented using computational reflection [2, 11, 27]. 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 [24]. The result of the evaluation is translated back to the PVS theorem prover using the PVSio package developed by one of the authors [17].

We illustrate the use of the `numerical` with some examples. Lemma `tr35` is the PVS version of Formula (1). The proof is just one step of `numerical` with no parameters:

```
g : posreal = 98/10         %[m/s^2]
v : posreal = 250×514/1000 %[m/s]

tr35: LEMMA 3×π/180 ≤ g×tan(35×π/180)/v
%|- tr35: PROOF (numerical) QED
```

Another example is the proof of the inequality 4.1.35 in Abramowitz & Stegun [1]:

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

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
  let x = 5828/10000 in
    G(x) > 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 considered when performing the numerical evaluation. The original proof of this lemma in PVS required the manual expansion of 19 terms of the ln series.

6 Conclusion

We have presented a pragmatic and safe way to perform ordinary real number computations in formal proofs. To this end, bounds for non-algebraic functions were established based on provable properties of their approximation series. Furthermore, a package for interval arithmetic was developed. The package includes a strategy that automatically discharges numerical inequalities and interval inclusions.

The PVS Interval package, which is available at `http://research.nianet.org/~munoz/Interval`, contains in total 306 lemmas. 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 (`http://shemesh.larc.nasa.gov/fm/ftp/larc/PVS-library/pvslib.html`). 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 of exact real arithmetic and interval arithmetic within a theorem prover.

Research on interval analysis and exact arithmetic is rich and abundant (see for example [10, 14, 16]). 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 [12] and ACL2 [8]. In Coq, an axiomatic definition of reals is given in [15], and constructive definitions of reals are provided in [3] and [20]. 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 [6] and currently being maintained and extended as part of the NASA Langley PVS Libraries. An alternative real analysis library is proposed in [9].

Closer to our approach are the tools presented in [4] and [5]. 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 rather than accuracy. In simple words, our practical contribution is a symbolic pocket calculator for real number computations in formal proofs.[4] 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 package and, in particular, the strategy `numerical` make these proofs routine in PVS.

Acknowledgment
The authors would like to thank Marc Daumas for his early interest on this work, and his key suggestion to use interval arithmetic to automate our initial lower/upper bound technique. We are also grateful to Jacques Fleuriot for providing the original proofs of trigonometric bounds in Isabelle/HOL, to Hanne Gottliebsen for providing the same proofs in PVS, to Jeff Maddalon for comments on early drafts of this manuscript, and to the anonymous referees for their insightful comments that helped to improve this document. This work was supported by the National Aeronautics and Space Administration under NASA Cooperative Agreement NCC-1-02043.

References

1. M. Abramovitz and I. Stegun. *Handbook of Mathematical Functions*. National Bureau of Standards, 1972.
2. S. Boutin. Using reflection to build efficient and certified decision procedures. *Lecture Notes in Computer Science*, 1281:515–530, 1997.
3. A. Ciaffaglione. *Certified Reasoning on Real Numbers and Objects in Coinductive Type Theory*. PhD thesis, Università degli Studi di Udine, 1993.
4. M. Daumas and G. Melquiond. Generating formally certified bounds on values and round-off errors. In *Real Numbers and Computers*, pages 55–70, Dagstuhl, Germany, 2004.
5. M. Daumas, G. Melquiond, and C. Muñoz. Guaranteed proofs using interval arithmetic. Accepted for publication at the 17th IEEE Symposium on Computer Arithmetic, 1995.
6. B. Dutertre. Elements of mathematical analysis in PVS. In J. von Wright, J. Grundy, , and J. Harrison, editors, *Theorem Proving in Higher Order Logics: 9th International Conference. TPHOLs'97*, number 1125 in Lecture Notes in Computer Science, pages 141–156, Turku, Finland, August 1996. Springer-Verlag.
7. J. Fleuriot and L. Paulson. Mechanizing nonstandard real analysis. *LMS Journal of Computation and Mathematics*, 3:140–190, 2000.
8. R. Gamboa. *Mechanically Verifying Real-Valued Algorithms in ACL2*. PhD thesis, University of Texas at Austin, May 1999.
9. H. Gottliebsen. *Automated Theorem Proving for Mathematics: Real Analysis in PVS*. PhD thesis, University of St Andrews, June 2001.

[4] The results are – of course – only as sound as PVS.

10. P. Gowland and D. Lester. A survey of exact computer arithmetic. In Jens Blanck, Vasco Brattka, Peter Hertling, and Klaus Weihrauch, editors, *Computability and Complexity in Analysis*, volume 272 of *Informatik Berichte*, pages 99–115. FernUniversität Hagen, September 2000. CCA2000 Workshop, Swansea, Wales, September 17–19, 2000.
11. J. Harrison. Metatheory and reflection in theorem proving: A survey and critique. Technical Report CRC-053, SRI Cambridge, Millers Yard, Cambridge, UK, 1995.
12. J. Harrison. *Theorem Proving with the Real Numbers*. Springer-Verlag, 1998.
13. T. Hickey, Q. Ju, and M.H. van Emden. Interval arithmetic: from principles to implementation. *Journal of the ACM*, 2001.
14. R. B. Kearfott. Interval computations: Introduction, uses, and resources. *Euromath Bulletin*, 2(1):95–112, 1996.
15. M. Mayero. *Formalisation et automatisation de preuves en analyses réelle et numérique*. PhD thesis, Université Paris VI, décembre 2001.
16. V. Ménissier. *Arithmétique Exacte : Conception, Algorithmique et Performances d'une Implantation Informatique en Précision Arbitraire*. PhD thesis, Université Paris VI, Paris, France, 1994.
17. C. Muñoz. PVSio reference manual – Version 2.a. Available from `http://research.nianet.org/~munoz/PVSio`, 2004.
18. C. Muñoz, V. Carreño, G. Dowek, and R.W. Butler. Formal verification of conflict detection algorithms. *International Journal on Software Tools for Technology Transfer*, 4(3):371–380, 2003.
19. C. Muñoz and D. Lester. Real number calculations and theorem proving. In J. Hurd and T. Melham, editors, *Proceedings of the 18th International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2005*, volume 3603 of *Lecture Notes in Computer Science*, pages 195–210, Oxford, UK, 2005. Springer-Verlag.
20. M. Niqui. *Formalising Exact Arithmetic: Representations, Algorithms and Proofs*. PhD thesis, Radboud University Nijmegen, 1994.
21. S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, June 1992. Springer-Verlag.
22. M. Pour-El and J. Richards. *Computability in Analysis and Physics*. Perspectives in Mathematical Logic. Springer, Berlin, 1989.
23. R.M. Robinson. Review of "Peter, R., Rekursive Funktionen". *The Journal of Symbolic Logic*, 16:280–282, 1951.
24. N. Shankar. Efficiently executing PVS. Project report, Computer Science Laboratory, SRI International, Menlo Park, CA, November 1999. Available at `http://www.csl.sri.com/shankar/PVSeval.ps.gz`.
25. E. Specker. Nicht konstruktiv beweisbare Sätze der Analysis. *The Journal of Symbolic Logic*, 14(3):145–158, 1949.
26. A. Turing. On computable numbers, with an application to the "Entscheidungsproblem". *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.
27. F. W. von Henke, S. Pfab, H. Pfeifer, and H. Rueß. Case Studies in Meta-Level Theorem Proving. In Jim Grundy and Malcolm Newey, editors, *Proc. Intl. Conf. on Theorem Proving in Higher Order Logics*, number 1479 in Lecture Notes in Computer Science, pages 461–478. Springer-Verlag, September 1998.
28. A. Yakovlev. Classification approach to programming of localizational (interval) computations. *Interval Computations*, 1(3):61–84, 1992.