Formal Global Optimisation with Taylor Models

Roland Zumkeller

École Polytechnique, 91128 Palaiseau Cedex, France

Abstract. Formal proofs and global optimisation are two research areas that have been heavily influenced by the arrival of computers. This article aims to bring both further together by formalising a global optimisation method based on Taylor models: a set of functions is represented by a polynomial together with an error bound. The algorithms are implemented in the proof assistant Coq's term language, with the ultimate goal to obtain formally proven bounds for any multi-variate smooth function in an efficient way. To this end we make use of constructive real numbers, interval arithmetic, and polynomial bounding techniques.

1 Introduction

Global optimisation, as it shall be understood in this article, is concerned with finding the minimum and maximum value of a given objective function $f : \mathbb{R}^n \to \mathbb{R}$ on a certain domain $[a_1; b_1] \times \ldots \times [a_n; b_n]$. Since this is generally difficult, we will in practice content ourselves with a bounding interval [c;d] such that $\forall x \in [a_1; b_1] \times \ldots \times [a_n; b_n].\ f\,x \in [c; d]$, of course desiring $[c; d]$ as narrow as possible.

Problems of this kind arise in a wide spectrum of science, ranging from engineering (aeronautics [4], robotics [16]), over experimental physics (particle motion in accelerators [12]) to geometry. A prominent instance of the last class is the proof of the Kepler conjecture given by Thomas Hales [10], in which some thousand lemmata asserting bounds on geometric functions occur.

From Extremum Criteria to Global Optimisation Algorithms

In 1755 Euler gave (based on previous work by Fermat) the well-known necessary condition $\nabla f x = 0$ for f to assume an extremum at x [5]. However, in most interesting cases effectively solving this equation is an equally difficult problem. During the following centuries a lot of more sophisticated criteria have been developed, but like Euler's most of them reduced the original problem to another difficult one.

The arrival of computers changed this situation: Previously intractable optimisation problems entered the scope of what could be solved. Moreover, this led mathematicians to develop new methods to treat this kind of problems. In 1962 Ramon E. Moore described the use of interval arithmetic on a computer, refined by a branch-and-bound algorithm to optimise a function over an interval [18]. This work has been the basis for many sophisticated refinements, making up the core of numerous current global optimisation algorithms [11].

From Formal Proof in Principle to Formal Proof in Fact

In 1879 Frege was first to introduce the notion of formal proof. With his *Begriffsschrift* he gave a language to express propositions and rules to reason on them in a purely syntactical manner. While this work showed how formal proofs can *in principle* be constructed, its usefulness remained limited by practical constraints: The amount of detail required to formally prove even relatively simple statements was unacceptably large for a human equipped only with pencil and paper.

Again, the arrival of computers changed this situation: In 1967, after mathematical logic had become a more thoroughly studied topic, Nicolaas G. de Bruijn developed the Automath system [3], which could syntactically verify that a given proof indeed demonstrates a given theorem. The fact that formal proofs could now be constructed and checked on a machine made their development more practical, and a certain amount of mathematics has been formalised in different systems since.

1.1 Formal Global Optimisation

The aim of the work described in this article is to apply formal proof techniques to global optimisation. More precisely, we are going to formalise an algorithm based on Taylor models [12] in the Coq system. Taylor models are the basis of one of the more recent global optimisation methods in the tradition of interval arithmetic, while Coq is a state-of-the-art proof assistant in the tradition of de Bruijn's system.

Solutions to other computationally difficult problems have already been formalised: a formal proof of the Four Colour Theorem has been given by Georges Gonthier and Benjamin Werner [6]. Also, a verification algorithm for Pocklington certificates of prime numbers has been proven correct [7]. In a slightly different setting, other computational parts of the Kepler conjecture proof have been formalised, namely a large graph enumeration problem [20] and linear programs [22].

Computational proofs are supported by an important characteristic of type theory: The so-called *conversion rule* identifies terms modulo β-conversion (computation). In fact, functional programs written in type theory can be referred to in proofs. For example assume that $\mathsf{test} : \mathsf{term} \to \mathsf{intvl} \to \mathsf{bool}$ implements a global optimisation method which attempts to prove that a certain function (described by its term) is positive on a given interval. Its correctness lemma states:

$$\forall f : \mathsf{term}, X : \mathsf{intvl}.\, \mathsf{test}\, f\, X = \mathsf{true} \to \forall x : \mathbb{R}.\, x \in X \to [\![f]\!]\, x > 0$$

If the method used is reasonably good $\mathsf{test}\,(\frac{1}{2} + \mathcal{X} + \mathcal{X}^2)\,[-1;1]$ will evaluate to true. Then we can prove $\forall x : \mathbb{R}.\ x \in [-1;1] \to \frac{1}{2} + x + x^2 > 0$ simply by applying the correctness lemma. The attractive feature of this technique, called *proof by reflection*, is that the computation steps do not have to be made explicit.

It suffices in fact to refer to the decision procedure test, whose computational trace can be reproduced if the proof needs to be re-checked. Therefore the trace does not need to be stored, which can dramatically reduce the size of certain proofs.

The rest of this article presents an implementation of Taylor models in Coq. Its description has to remain on a rather abstract level, due to the number of different concepts involved. However, the implementation follows quite closely the theoretical presentation given here. Our ultimate goal is to entirely prove correct the Taylor model algorithm and to refine our implementation sufficiently, so that it can treat all of the above-mentioned lemmata appearing in the proof of the Kepler conjecture.

Outline

Section 2 presents interval arithmetic and its traditional use for global optimisation. The dependency problem and the loss of sharpness of the united extension are discussed. A solution to the latter is suggested in section 3 where we explain how constructive reals can be used as interval bounds. In particular, a generalisation of Moore's sign-based interval multiplication is given. In section 4 Taylor models are presented according to [12]. We simplify it by giving a new technique for composing smooth functions with Taylor models, not requiring the manual insertion of an addition theorem (4.2). Besides, it is shown that the choice of reference points for the development of smooth functions can be improved (4.3). Finally, we give some examples of our implementation's performance (section 5) before reaching the conclusions (section 6).

2 Interval Arithmetic

As mentioned, interval arithmetic on computers has been developed by Moore in the early 1960s. We will briefly describe its traditional use for global optimisation, thereby summarising its elementary notions. While we are not going to follow this approach here, Taylor models themselves make careful use of interval arithmetic.

Definition 1. *Given a set of bounds $B \subseteq \mathbb{R}$ the set of associated* intervals *is defined as $\mathbb{I}_B := \{[a;b] \mid a,b \in B\}$ where $[a;b] := \{x \in R \mid a \leq x \leq b\}$. We also note the set of n-dimensional* boxes *as $\mathbb{I}_{\mathbb{R}}^n := \{X_1 \times \ldots \times X_n \mid X_1, \ldots, X_n \in \mathbb{I}\}$, and $[x_1, \ldots, x_k]$ the smallest interval containing all of $x_1, \ldots, x_k$, each one of which is either a bound or an interval.*

Definition 2. *For any $B \subseteq \mathbb{R}$ the function $\hat{f} : \mathbb{I}_B^n \to \mathbb{I}_B$ is a* B-interval extension *of $f : \mathbb{R}^n \to \mathbb{R}$ iff:*

$$\forall X \in \mathbb{I}_B^n.\ \hat{f}\, X \supseteq \{f\, x \mid x \in X\}$$

If $\supseteq$ can be replaced by $=$ in the preceding line, $\hat{f}$ is called B-sharp.

We will first develop interval arithmetic on $\mathbb{I}_{\mathbb{R}}$. The problems to which other choices for the bounds lead will be discussed in section 2.3. However, in section 3 we will show that, in spite of tradition, a choice different from $\mathbb{R}$ is not mandatory for an implementation.

2.1 Operations

Interval extensions of some basic real functions are given in table 1. They are all sharp, as can be easily verified. Extensions of more complicated functions can be obtained by structural recursion on the term describing them. The result is referred to as the *natural interval extension*. Note that it is in general not sharp.

$$\begin{aligned}
[a;b] \hat{+} [c;d] &:= [a+c; b+d] \\
\hat{-}\,[a;b] &:= [-b; -a] \\
[a;b] \hat{\cdot} [c;d] &:= [\min\{ac, ad, bc, cd\}; \max\{ac, ad, bc, cd\}] \\
1\hat{/}\,[a;b] &:= [1/b; 1/a] \quad \text{if } 0 \notin [a;b] \\
\hat{\arctan}[a;b] &= [\arctan a; \arctan b] \\
\hat{\sqrt{}}[a;b] &= [\sqrt{a}; \sqrt{b}] \quad \text{if } 0 \leq a
\end{aligned}$$

Table 1. Interval extensions of some real functions

The extension of multiplication is inefficient if implemented as suggested by the formula given in table 1, because all of ac, ad, bc, and bd are computed before determining their minimum and maximum. In fact this can be accelerated [17] by looking at the signs of a, b, c, and d, using the fact that $a \leq b$ and $c \leq d$.

Definition 3. *An interval* $[a;b]$ *has* sign $+$ *if* $a > 0$, sign $-$ *if* $b < 0$ *and* sign $\pm$ *if* $a \leq 0 \leq b$.

Given the signs of the two intervals $[a;b]$ and $[c;d]$ we can compute the extension of multiplication as described in the following table:

$[a;b]$	$[c;d]$	$\min\{ac, ad, bc, bd\}$	$\max\{ac, ad, bc, bd\}$
$+$	$+$	ac	bd
$+$	$\pm$	bc	bd
$\pm$	$\pm$	$\min\{ad, bc\}$	$\max\{ac, bd\}$

The other six cases can be reduced to these by the two equations:

$$[a;b] \hat{\cdot} [c;d] = [c;d] \hat{\cdot} [a;b] = -([-b; -a] \hat{\cdot} [c;d])$$

2.2 Global Optimisation with Interval Arithmetic

In order to obtain bounds for a given function f on a domain X, it is now sufficient to construct an interval extension $\hat{f}$ of f (e.g. the natural one). Then,

by the extension property, computing the interval $\hat{f}\ X$ provides bounds for f on X.

However, and this is why the story does not end here, these bounds are in general quite crude. Optimising $x \mapsto x - x$ on $[a; b]$ by this method yields $[a - b; b - a]$, although it is easy to see that $[0; 0]$ is a bound here. One might object that $x - x$ could easily be recognised and rewritten to 0. This is correct (and even helpful), but there are many other similar cases, e.g. $(\sin x)^2 + (\cos x)^2$. No general procedure is known to cover all of them.

It might seem surprising that the interval extension of the subtraction given in table 1 is sharp, while the result that it yields on the above example is not optimal. This point merits some study. To begin with, the extension property for $\hat{-}$ states that for any $X, Y \in \mathbb{I}$:

$$\forall x \in X, y \in Y.\ x - y \in X \mathbin{\hat{-}} Y$$

Furthermore, sharpness asserts that for any X, Y the interval $X \hat{-} Y$ is the narrowest one satisfying this statement. By setting Y to X we obtain:

$$\forall x, y \in X.\ x - y \in X \mathbin{\hat{-}} X$$

This shows the problem clearly: An argument to an interval extension does not contain any reference to the variable it represents. The term $X \mathbin{\hat{-}} X$ represents not only $x - y$, but of course also the less general $x - x$. It cannot be evaluated to $[0; 0]$ because the information that both its arguments are the same has been lost. This phenomenon is often referred to as the *dependency problem.*

A simple strategy to improve the quality of the bounds is based on interval-splitting. In one dimension the *united extension* $\hat{f}_n$ of f is given by:

$$\hat{f}_n\,[a; b] = \bigcup_{i=1}^{n} \hat{f}\ \left[a + (i-1)\frac{b-a}{n}; a + i\frac{b-a}{n}\right]$$

Moore showed in his thesis [18] that $[a; b] \mapsto \lim_{n\to\infty} \hat{f}_n[a; b]$ is a sharp extension of f. This can – with exponential complexity – easily be generalised to the multi-dimensional case. Although important, this technique is not yet sufficient for most applications, which is why numerous other refinements [11,19] have been developed. In section 4 we are going to see how Taylor models address the dependency problem more directly and are able to overcome it in part.

2.3 The Set of Interval Bounds

Traditionally some set of "machine-representable" numbers $B \subset \mathbb{R}$ has been used to implement interval bounds. However, note that even when we are able to obtain a basic set of B-sharp functions (such as in table 1), the B-sharpness of the united extension can be lost, as shows the following example:

Example 1. Let $B = \{\frac{k}{1000} \mid k \in \mathbb{N}\}$. Denote $\lfloor x \rfloor = \max\{b \in B \mid b \leq x\}$ and $\lceil x \rceil = \min\{b \in B \mid x \leq b\}$. We then have the B-sharp extensions:

$$[a;b] \mathbin{\hat{-}_B} [c;d] = [a-d; b-c]$$
$$\sqrt{[a;b]}_B = \left[\lfloor\sqrt{a}\rfloor ; \left\lceil\sqrt{b}\right\rceil\right]$$

Now consider the function $f = x \mapsto \sqrt{x} - \sqrt{x}$. Its natural B-interval extension is $\hat{f} = [a;b] \mapsto \left[\ \lfloor\sqrt{a}\rfloor - \left\lceil\sqrt{b}\right\rceil ; \left\lceil\sqrt{b}\right\rceil - \lfloor\sqrt{a}\rfloor\right]$. For its united extension we have

$$\hat{f}_n\,[0;1] = \bigcup_{i=1}^{n} [-\delta_{i,n}; \delta_{i,n}] = [-\delta_{1,n}; \delta_{1,n}] \text{ where } \delta_{i,n} = \left\lceil\sqrt{\frac{i}{n}}\right\rceil - \left\lfloor\sqrt{\frac{i-1}{n}}\right\rfloor$$

Note that $\delta_{1,n} = \left\lceil\sqrt{\frac{1}{n}}\right\rceil \geq \frac{1}{1000}$ for any n. Thus $\lim_{n\to\infty} \hat{f}_n\,[0;1] = [0; \frac{1}{1000}] \neq [0;0]$, which means that $\hat{f}_n$ is not B-sharp. □

The same phenomenon occurs with floating-point numbers. As an alternative, taking rational numbers allows us to have sharp interval extensions of basic arithmetic to $\mathbb{I}_{\mathbb{Q}}$. However, for the irrational functions (e.g. square root) there are no sharp extensions. In order to refine bounds for f obtained by computing $\hat{f}_n$ for some n we thus have two options:

- Recompute $\hat{f}_n$ for a larger n.
- Use a more precise set of bounds $B \subset \mathbb{R}$. In practice this can mean to increase some precision parameter.

The first option would be a good choice for $f\,x = x - x$, the second one for $f\,x = \sqrt{x}$ on $[0;2]$. However, it is not easy to say which choice is better in general. Making the wrong one will lead to unnecessary computations.

3 Constructive Real Numbers

No solution has been given to the loss of sharpness for the united extension occurring when using numbers that are "machine-representable" in the traditional sense. We will explain how constructive analysis can be used to represent the whole of (constructively defined) $\mathbb{R}$ on a machine. This approach allows us to regain sharpness of the united extension.

Real numbers can be seen as equivalence classes of Cauchy sequences. $x : \mathbb{N} \to \mathbb{Q}$ is a Cauchy sequence if:

$$\forall \varepsilon > 0.\ \exists n.\ \forall k_1, k_2 \geq n.\ \left|\, x\,k_1 - x\,k_2 \,\right| < \varepsilon$$

By the Curry-Howard-isomorphism constructively proving this property for a given x amounts to providing a function $m : \mathbb{Q} \to \mathbb{N}$ (referred to as the *modulus*) such that

$$\forall \varepsilon > 0.\ \forall k_1, k_2 \geq m\,\varepsilon.\ \left|\, x\,k_1 - x\,k_2 \,\right| < \varepsilon$$

A real number can thus be defined as a pair $(x, m) \in (\mathbb{N} \to \mathbb{Q}) \times (\mathbb{Q} \to \mathbb{N})$ verifying this property. They are equipped with two families of ε-indexed relations:

$$(x_1, m_1) =_\varepsilon (x_2, m_2) :\Leftrightarrow \left| (x_1 \circ m_1)\,\varepsilon - (x_2 \circ m_2)\,\varepsilon \right| \leq 2\varepsilon$$
$$(x_1, m_1) <_\varepsilon (x_2, m_2) :\Leftrightarrow (x_1 \circ m_1)\,\varepsilon + 2\varepsilon \leq (x_2 \circ m_2)\,\varepsilon$$

We also note:

$$a < b :\Leftrightarrow \exists \varepsilon.\ a <_\varepsilon b \qquad \text{and} \qquad a = b :\Leftrightarrow \forall \varepsilon.\ a =_\varepsilon b$$

Note that $=_\varepsilon$ and $<_\varepsilon$ are decidable for a given ε, whereas $=$ and $<$ are not. We give here only a few examples of functions on constructive reals:

$$(x_1, m_1) + (x_2, m_2) := \left(k \mapsto x_1\,k + x_2\,k, \varepsilon \mapsto \max\left\{m_1\left(\frac{\varepsilon}{2}\right), m_2\left(\frac{\varepsilon}{2}\right)\right\}\right)$$
$$\min\,(x_1, m_1)\,(x_2, m_2) := (k \mapsto \min\,(x_1\,k)\,(x_2\,k), \varepsilon \mapsto \max\,(m_1\,\varepsilon)\,(m_2\,\varepsilon))$$
$$\mathrm{sgn}_\varepsilon\,(x, m) := \begin{cases} 1 & \text{if } x\,\varepsilon < -\varepsilon \\ -1 & \text{if } \varepsilon < x\,\varepsilon \\ 0 & \text{otherwise} \end{cases}$$

More details on constructive real numbers and their implementations can be found in [25,21,23,2,15]. Irrational functions (square root, trigonometry etc.) can be computed without rounding. This means that in order to obtain an approximation for a larger formula we have to provide only one precision argument.

Real numbers defined in this way are termed "constructive" because they serve as the basis of constructive analysis, in which every proof induces an algorithm. However, when one does not care for algorithms, it is of course acceptable to use the excluded middle to reason about them.

3.1 Interval Multiplication

When using constructive real numbers as interval bounds, the undecidability of their sign is not entirely harmless. For the multiplication of two intervals Moore's procedure (section 2.1) cannot be applied since it requires sign information for all four bounds involved. If sgn_ε yields 0 for one of the two intervals, we would have to fall back to the more inefficient version of table 1.

In order to avoid this, we propose a generalisation of Moore's efficient interval multiplication to intervals with constructive real numbers as bounds. It performs a finer case analysis, able to eliminate candidates among $\{ac, ad, bc, cd\}$ based on only partial sign information.

We note $(x_{-1}, x_1, y_{-1}, y_1) := (a, b, c, d)$. Under the assumptions $x_{-1} < x_1$ and $y_{-1} < y_1$ (which happens to be an invariant of all interval operations) we have for any ε:

$$\min\{x_i y_j \mid i, j \in \{-1, 1\}\} = \min\{x_i y_j \mid i, j \in \{-1, 1\} \wedge \neg(0 <_\varepsilon j x_i \vee 0 <_\varepsilon i y_j)\}$$

Proof. We have to show that for the smallest element $x_i y_j$ the property $\neg(0 <_\varepsilon jx_i \vee 0 <_\varepsilon iy_j)$ holds. If $0 <_\varepsilon jx_i$ then $x_i y_{-j} < x_i y_j$. Symmetrically, if $0 <_\varepsilon iy_i$ then $x_{-i} y_j < x_i y_j$. Both conclusions contradict the assumption that $x_i y_j$ is minimal. □

What ε should be chosen? Note that this choice does not affect the result of the given procedure but only its performance. If ε is chosen too small, the cost of determining $0 <_\varepsilon jx_i$ and $0 <_\varepsilon iy_j$ becomes high, if chosen too large the set which the min function is applied to is more likely to contain more than one element. Experiments show that this choice is in most cases not critical.

3.2 Partial Functions

Looking at table 1 we notice that the extension of the multiplicative inverse has $0 \notin [a; b]$ as a side condition. If it is not satisfied then the inverse function in our implementation returns the special interval $[-\infty; \infty]$, conveying the information that nothing can be pe proved about the result.

Because the condition is undecidable on constructive reals we can only give an ε-indexed family of interval extensions:

$$\hat{1/}_\varepsilon[a; b] := \begin{cases} [1/b; 1/a] & \text{if } 0 <_\varepsilon a \vee b <_\varepsilon 0 \\ [-\infty; \infty] & \text{otherwise} \end{cases}$$

Once a function has returned the interval $[-\infty; \infty]$ this result is propagated (e.g. $[-\infty; \infty] + [a; b] = [-\infty; \infty]$) and the computation thus aborted. Except for this error case the sharpness of the united extension remains ensured.

4 Taylor Models

The dependency problem described in section 2.2 gave rise to many refinements of interval arithmetic, such as variable centring, domain-splitting, domain projections, or gradient checks [11,19]. All of these provide a certain remedy, but they do not actually solve the problem. It has often been observed that Taylor expansions of the function to optimise can be used to obtain better results with interval arithmetic. Taylor *models* [12] exploit this fact systematically and combine it with methods for the efficient computation of derivatives.

As we have seen, the dependency problem stems from the fact that the information that $x - x$ is different from $x - y$ is lost on the interval level. This is because interval arithmetic is a *calculus of number sets*, unable to represent this kind of information. In contrast to this, Taylor models provide a *calculus of function sets*.

We note $\mathbb{R}[\mathcal{X}_1, \ldots, \mathcal{X}_k]$ the set of k-variate polynomials with real coefficients in the variables $\mathcal{X}_1, \ldots, \mathcal{X}_k$. Their addition and multiplication are assumed.

Definition 4. *The set of k-variate* Taylor models *is defined as* $\mathbb{T} := \mathbb{I}^k \times \mathbb{R}[\mathcal{X}_1, \ldots, \mathcal{X}_k] \times \mathbb{I}$. *For a Taylor model $(X, P, \Delta) \in \mathbb{T}$ the box X is called its domain, P its polynomial, and Δ its error bound. The set of all Taylor models over some given domain X is denoted by $\mathbb{T}_X$.*

A Taylor model represents a set of functions:

$$[\![(X, P, \Delta)]\!] = \{f : X \to \mathbb{R} \mid \forall x.\, f\, x - P\, x \in \Delta\}$$

4.1 Arithmetic on Taylor Models

A polynomial bounder $B : \mathbb{R}[\mathcal{X}_1, \ldots, \mathcal{X}_k] \to \mathbb{I}^k \to \mathbb{I}$ is assumed to be available, e.g. by a Horner evaluation in interval arithmetic. $(P)_{\leq n}$ denotes the polynomial P up to the nth coefficient, and $(P)_{>n}$ the remaining part.

A binary operation on Taylor models $\odot_{\mathbb{T}}$ correctly reflects $\odot_{\mathbb{R}}$ on reals iff:

$$[\![T_1 \odot_{\mathbb{T}} T_2]\!] \supseteq \{x \mapsto f\, x \odot_{\mathbb{R}} g\, x \mid f \in [\![T_1]\!], g \in [\![T_2]\!]\}$$

This condition is satisfied by the following definitions, valid for two Taylor models with identical domain [12]:

$$\begin{aligned}(X, P_1, \Delta_1) \tilde{+} (X, P_2, \Delta_2) &= (X, P_1 + P_2, \Delta_1 \hat{+} \Delta_2)\\ (X, P_1, \Delta_1) \tilde{\cdot} (X, P_2, \Delta_2) &= (X, (P_1 \cdot P_2)_{\leq n}, B\,(P_1 \cdot P_2)_{>n}\, X \hat{+}\\ &\qquad B\, P_1\, X \hat{\cdot} \Delta_2 \hat{+} \Delta_1 \hat{\cdot} B\, P_2\, X \hat{+} \Delta_1 \hat{\cdot} \Delta_2)\end{aligned}$$

For multiplication the degree n can be arbitrarily chosen. Higher values will lead to better accuracy, but also to a higher computational cost.

4.2 Composing Smooth Functions with Taylor Models

After having defined addition and multiplication in the previous section we are able to evaluate any polynomial in $\mathbb{T}_X$. The natural next step is to construct Taylor model versions for the square root or trigonometric functions (the multiplicative inverse will be treated along with them).

We show how, given any smooth function $g : Y \to R$ (where $Y \subseteq \mathbb{R}$) and a Taylor model $F \in \mathbb{T}_X$ (where $X \in \mathbb{I}^k$) we can construct a new Taylor model $H \in \mathbb{T}_X$ such that:

$$\{g \circ f \mid f \in [\![F]\!]\} =: g \circ [\![F]\!] \subseteq H$$

The proposed solution of this problem is not going to use the fact that F is represented by a Taylor model. It would work as well for any other representation of function sets.

The idea is to apply Taylor's theorem to develop g around a freely chosen reference point $y_0 \in Y$. We thus have for any $f \in F$:

$$\forall x \in X.\, g\,(f\, x) \in \sum_{k=0}^{n} \frac{g^{(k)}\, y_0}{k!}(f\, x - y_0)^k + \frac{g^{(n+1)}[y_0, f\, x]}{(n+1)!}(f\, x - y_0)^{(n+1)} \qquad (1)$$

The left summand can be written in notation of functions arithmetic, while we make the right summand an interval independent of x (by taking the union of all possibles values of f on X). (1) thus implies:

$$g \circ f \in \sum_{k=0}^{n} \frac{g^{(k)}\, y_0}{k!}(f - y_0)^k + \frac{g^{(n+1)}[y_0, f\, X]}{(n+1)!}(f\, X - y_0)^{(n+1)}$$

The last step is to include all possible choices within F and to bound it on X:

$$g \circ F \subseteq \sum_{k=0}^{n} \frac{g^{(k)}\, y_0}{k!} (F - y_0)^k + \frac{g^{(n+1)}[y_0, B\,F\,X]}{(n+1)!} (B\,F\,X - y_0)^{(n+1)} =: H \quad (2)$$

We're now done with the construction of H. It can actually be implemented: The left summand is a polynomial of Taylor models (or any other structure representing function sets), for which we have already defined the arithmetic operations. The right summand relies on the bounding of F and interval arithmetic. It contributes to the resulting Taylor model's error interval.

This construction is inspired by the strategy for composing smooth functions with Taylor models given in [12]: Note c the constant part of F (i.e. the first coefficient of its polynomial) and $\bar{F} := F - c$. Use an addition theorem for g to split $g \circ F = g \circ (c \tilde{+} \bar{F})$ into two parts. Then apply Taylor's theorem with reference point 0 to the part including $\bar{F}$. For the logarithm this strategy gives us:

$$\log \circ F = \log \circ (c + \bar{F}) = \log \circ \left\{ c \cdot \left(1 + \frac{\bar{F}}{c} \right) \right\} = \log c + \log \circ \left(1 + \frac{\bar{F}}{c} \right) \quad (3)$$

$$\in \log c + \sum_{k=1}^{n} \frac{(-1)^{k-1}}{k} \left(\frac{\bar{F}}{c} \right)^k + \frac{(-1)^n \left(\frac{B\,\bar{F}\,X}{c} \right)^{n+1}}{(n+1) \left(1 + \left[0, \frac{B\,\bar{F}\,X}{c} \right] \right)^{n+1}} \quad (4)$$

This kind of reasoning does of course not represent any difficulty for the working mathematician. However, finding an appropriate addition theorem for a given g requires a certain amount of creativity, which can only be provided by a human. This is why in [14] this strategy has been applied to many different functions manually: $x \mapsto \frac{1}{x}$, sin, cos, arctan, log etc., so that they could be implemented. In contrast, our construction (2) can entirely be carried out by a machine and is still general enough to cover all these functions. The idea was to apply Taylor's theorem immediately (i.e. without invocation of an addition theorem) with some carefully chosen y_0 (instead of 0) as reference point. For the logarithm (2) yields:

$$\log \circ F \subseteq \log y_0 + \sum_{k=1}^{n} \frac{(-1)^{k-1}}{k y_0^k} (F - y_0)^k + \frac{(-1)^n (B\,F\,X - y_0)^{n+1}}{(n+1)[y_0, B\,F\,X]^{n+1}}$$

Choosing $y_0 = c$ makes this equivalent to (4). The "creative part" of applying a function-dependent addition theorem, done in step (3), has been made superfluous, so a machine can entirely perform the task. Our Coq implementation actually contains a generic function that provides a Taylor model extension given only a function's Taylor coefficients as arguments.

4.3 What Reference Point to Choose?

With this procedure established, a point that merits some more study is the choice to be made for the reference point y_0. As mentioned, the strategy described

in [13] is equivalent to setting $y_0 = c$. However Taylor's theorem can be applied with any $y_0 \in Y$ as reference point. A good choice is one that minimises the width of the resulting Taylor model's error interval. In fact, there are cases where a better choices for y_0 than the Taylor model's constant part can be made. We illustrate this by the following example:

Example 2.

$$\frac{1}{(1+\mathcal{X}^2,[0;0])} \subseteq \sum_{k=0}^{n} \frac{(-1)^k}{y_0^{k+1}}(1+\mathcal{X}^2,[0;0])^k + \frac{(-1)^{n+1}}{[y_0,1+X^2]^{n+2}}(1+X^2-y_0)^{n+1}$$

At order $n = 2$ we obtain (assuming $y_0 \in 1 + X^2$):

$$\left(\frac{1-3y_0+3y_0^2}{y_0^3} + \frac{2-3y_0}{y_0^3}\mathcal{X}^2, \frac{X^4}{y_0^3} - \frac{(1+X^2-y_0)^3}{(1+X^2)^4}\right)$$

Let us further fix $X := [-\frac{1}{2}; \frac{1}{2}]$. For this example the optimal (i.e. minimising the width of the error interval) choice for the reference point is *not* $y_0 = c = 1$. For $y_0 = 1$ the error interval has the width 0.078125, and for $y_0 = \frac{5}{4}$ the width is 0.047625. Careful study shows that the latter is optimal. □

This example is limited to the multiplicative inverse and also to the case where the error part of the Taylor model given as an argument is zero. It is not obvious how to obtain an optimal value for y_0 in general. However, there are cases where better choices than c can be made. It would be interesting to see if a general procedure can be derived.

4.4 Implementation of Taylor Models

In Coq we represent a Taylor model as a record of two fields:

```
Record TaylorModel (degree : nat) (X : list intvl) : Type := TM {
  approx : Poly R (length X);
  error : intvl
}
```

This type is parameterised by the `degree` at which Taylor operations will be carried out (section 4.1) and the domain `X`. The field `approx` contains a polynomial with real coefficients in `length X` (the dimension of the domain) variables. The field `error` is the Taylor model's error interval.

Polynomials An $(n+1)$-variate polynomial can be represented as a polynomial with n-variate polynomials as coefficients. This is justified by the canonical polynomial isomorphism:

$$\mathbb{R}[\mathcal{X}_1, \ldots, \mathcal{X}_{n+1}] \simeq \mathbb{R}[\mathcal{X}_1, \ldots, \mathcal{X}_n][\mathcal{X}_{n+1}]$$

This can be translated to Coq as follows [9]:

```
Fixpoint PolyN (n:nat) struct n : Type :=
match n with
| 0 => C
| S m => list (PolyN m)
end.
```

The coefficients of the Taylor models' polynomials are represented by constructive real numbers because they can become irrational. Other choices have different consequences:

- Floating-point numbers: Their usage as coefficients will only yield approximations without error bounds, which is unacceptable for formal proof. An explicit treatment of rounding errors is possible, as has been shown for addition and multiplication [24]. However this approach can be expected to be much more difficult for the Taylor development of general smooth functions.
- Intervals: Using intervals with rational or floating-point number bounds as coefficients is feasible, but they add to the complexity of proofs. Furthermore, they force the user to give a precision for the enclosure of irrational values, thereby affecting sharpness in a way difficult to control.

Bounding Taylor models make use of polynomial bounding for multiplication (section 4.1) and composition with smooth functions (section 4.2). A simple way to bound polynomials is to evaluate their interval extension. However there are much better strategies yielding narrower bounds. For example it is well-known that a univariate quadratic function can be rewritten by:

$$c_2x^2 + c_1x + c_0 = c_2\left(x + \frac{c_1}{2c_2}\right)^2 - \frac{c_1^2}{4c_2} + c_0$$

The natural interval extension of the right hand side will then yield sharp bounds. There are much more sophisticated techniques for multi-variate polynomials of several degrees, as described in section 5.4.3 of [12]. Our implementation does not include them yet.

Computing Taylor Coefficients As we have seen, in order to apply a smooth function to a Taylor model it is necessary to compute the coefficients of its Taylor series. Doing this by symbolic derivation is prohibitively expensive, so we use combinatoric formulas to obtain the derivatives. For example:

$$\mathrm{inv}^{(k)}y = (-1)^k\frac{k!}{y^{k+1}} \qquad \log^{(k)}y = \mathrm{inv}^{(k-1)}y = (-1)^{(k-1)}\frac{(k-1)!}{y^k}$$

A perhaps less well-known formula is [1]:

$$\arctan^{(k)}y = \frac{n!}{(1+y^2)^n}\sum_{k=0,\ n+k\,\mathrm{odd}}^{n}(-1)^{\frac{n+k+1}{2}}\binom{n}{k}y^k$$

These formulas are evaluated both in real numbers and in the interval arithmetic, as required by equation 2.

5 Examples

Example 3. As a first example [13] we study the function $f\,x = \frac{1}{x} + x$ on the domain $[1.9; 2.1]$. Our implementation yields the following results (the last line of the table shows the actual bound):

order of Δ	width of Δ	bound interval
0	$5.0125313 \cdot 10^{-2}$	$[2.3761905; 2.6263158]$
1	$5.5401662 \cdot 10^{-2}$	$[2.3722992; 2.6277008]$
2	$1.4579385 \cdot 10^{-3}$	$[2.4250000; 2.5764579]$
3	$1.5346721 \cdot 10^{-4}$	$[2.4236733; 2.5763267]$
4	$4.0386107 \cdot 10^{-6}$	$[2.4236875; 2.5763165]$
∞	0	$[2.4263158; 2.5761905]$

All these results have been obtained in less than a second using Coq's compiler [8]. □

Example 4. Lemma I_751442360 in Thomas Hales' proof of the Kepler conjecture [10] states that

$$\frac{\begin{array}{c}-x_1x_3 - x_2x_4 + x_1x_5 + x_3x_6 - x_5x_6 + \\ x_2(-x_2 + x_1 + x_3 - x_4 + x_5 + x_6)\end{array}}{\sqrt{4x_2\begin{pmatrix}x_2x_4(-x_2 + x_1 + x_3 - x_4 + x_5 + x_6) + \\ x_1x_5(x_2 - x_1 + x_3 + x_4 - x_5 + x_6) + \\ x_3x_6(x_2 + x_1 - x_3 + x_4 + x_5 - x_6) \\ - x_1x_3x_4 - x_2x_3x_5 - x_2x_1x_6 - x_4x_5x_6\end{pmatrix}}} < \tan\left(\frac{\pi}{2} - 0.74\right) \approx 1.09518$$

for all $x \in X_{751442360}$, which is a 6-dimensional box given in the proof. We bound the left-hand side on a sub-domain of $X_{751442360}$ which is smaller (of width $\approx \frac{1}{10}$ in every dimension), but still contains the global maximum. The results are:

order of Δ	width of Δ	bound interval
0	∞	$[-\infty; \infty]$
1	1.3025929	$[0.36246433; 1.6650572]$
2	$1.6105738 \cdot 10^{-2}$	$[0.94291234; 1.0905563]$
∞	0	$[0.95253193; 1.0849205]$

The result for order 3 has been obtained in about ten minutes[1] . Further work needs to be done in order to improve this performance. However, it is already sufficiently tight for proving the required statement on the given small sub-domain. □

6 Conclusion

The global optimisation problems included in Thomas Hales' proof of the Kepler are the most complex to have been included in a mathematical proof so

[1] on an Intel Pentium 4 running at 2.80GHz

far. Having shown that they are not entirely out of reach for current proof assistants encourages us to further pursue our direction of work. Many paths can be followed to improve our implementation, two of which we find worth mentioning:

- In order to bound the multi-variate polynomials appearing in the Taylor models our current implementation evaluates them in interval arithmetic using the Horner-scheme induced by the canonical polynomial isomorphism. However, many better methods are available, which should considerably tighten the resulting bounds.
- Implementations of constructive real numbers in a style of pure functional programming suffer from an important performance problem: the cost of evaluating $x + x$ to precision ϵ is twice that of evaluating x to precision $\frac{\epsilon}{2}$. The same computation is actually carried out twice. This problem could be avoided by stocking previously computed results in a global cache.

It should be kept in mind that we have to pay a certain performance penalty for the maximal security achieved by implementing algorithms inside a proof assistant. However, the speed of machines has been increasing at an exponential rate for a long time. Besides, algorithms become more and more efficient. As a consequence, today we are able to treat problems in a formal setting that twenty years ago were only in reach for implementations on machine level.

In order to diminish this gap, work on proof assistants itself is necessary: a more efficient mechanism for computation has recently been added to Coq [8]. A second step in this direction would be the usage of machine numbers for computations inside proofs. With such tools at hand, our implementation could become a powerful system performing verified optimisation for a large class of functions.

Acknowledgements

I would like to thank Bas Spitters, Milad Niqui, and Russell O'Connor for many instructive discussions on constructive real numbers. I'm grateful to my supervisor, Benjamin Werner, for advice he gave to me at various stages of this work. Furthermore I would like to thank the anonymous referees as well as Nathalie Revol for their valuable comments on earlier versions of this article.

References

1. Mohammad K. Azarian. A076741. In N. J. A. Sloane, editor, *The On-Line Encyclopedia of Integer Sequences*. `www.research.att.com/~njas/sequences/`, 2002.
2. Yves Bertot. Calcul de formules affines et de séries entières en arithmétique exacte avec types co-inductifs. Technical report, 2005.
3. Nicolaas G. de Bruijn. The mathematical language AUTOMATH, its usage, and some of its extensions. In M. Laudet, editor, *Proceedings of the Symposium on Automatic Demonstration*, pages 29–61, Versailles, France, December 1968. Springer-Verlag LNM 125.

4. G. Dowek, A. Geser, and C. Muñoz. Tactical conflict detection and resolution in a 3-D airspace. In *Proceedings of the 4th USA/Europe Air Traffic Management R&DSeminar, ATM 2001*, Santa Fe, New Mexico, 2001.
5. Leonhard Euler. Institutiones calculi differentialis, 1755.
6. Georges Gonthier. A computer-checked proof of the Four Colour Theorem. Preprint, 2005.
7. B. Grégoire, L. Thery, and B. Werner. A computational approach to pocklington certificates in type theory. FLOP 2006, to appear in LNCS.
8. Benjamin Grégoire and Xavier Leroy. A compiled implementation of strong reduction. In Proceedings ICFP'02.
9. Benjamin Grégoire and Assia Mahboubi. Proving equalities in a commutative ring done right in coq. In Joe Hurd and Tom Melham, editors, *Proceedings of TPHOLs'05*, volume 3603 of *Lecture Notes in Computer Science*, pages 98–113, Oxford, UK, August 2005. Springer-Verlag.
10. Thomas C. Hales. A proof of the Kepler Conjecture. Manuscript, 2004.
11. E. Hansen. *Global Optimization Using Interval Analysis*. M. Dekker, 1992.
12. Kyoko Makino. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators*. PhD thesis, Michigan State University, East Lansing, MI, USA, 1998. Also MSUCL-1093.
13. Kyoko Makino and Martin Berz. Remainder differential algebras and their applications. In Martin Berz, Christian Bischof, George Corliss, and Andreas Griewank, editors, *Computational Differentiation: Techniques, Applications, and Tools*, pages 63–74. SIAM, Philadelphia, Penn., 1996.
14. Kyoko Makino and Martin Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.
15. Valérie Ménissier-Morain. *Arithmétique exacte: conception, algorithmique et performances d'une implémentation informatique en précision arbitraire*. Phd thesis, Université Paris 7, 1994.
16. Jean-Pierre Merlet. Solving the forward kinematics of a gough-type parallel manipulator with interval analysis. *International Journal of Robotics Research*, 23(3):221–236, 2004.
17. Ramon E. Moore. Automatic error analysis in digital computation. Technical Report LMSD-48421 Lockheed Missiles and Space Co, Palo Alto, CA, 1959.
18. Ramon E. Moore. *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. PhD thesis, Department of Computer Science, Stanford University, 1962.
19. Ramon E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs N. J., 1966.
20. Tobias Nipkow, Gertrud Bauer, and Paula Schultz. Flyspeck i: Tame graphs. In U. Furbach and N. Shankar, editors, *Int. Joint Conf. Automated Reasoning — IJCAR 2006*, volume ?, pages ?–?
21. Milad Niqui. *Formalising Exact Arithmetic: Representations, Algorithms and Proofs, Radboud University Nijmegen*. PhD thesis, Radboud University Nijmegen, September 2004.
22. Steven Obua. Proving bounds for real linear programs in isabelle/hol. In *TPHOLs*, pages 227–244, 2005.
23. Russell O'Connor. A monadic, functional implementation of real numbers. Preprint, 2005.
24. N. Revol, K. Makino, and M. Berz. Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in COSY.
25. Helmut Schwichtenberg. Constructive analysis with witnesses, 2005. Manuscript.