Formal proofs and certified computation in Coq for solving the Table Maker's Dilemma

Érik Martin-Dorel

Postdoc in the Toccata team, Inria Saclay – Île-de-France, LRI

RAIM 2013, Institut Henri Poincaré
19th November 2013

Acknowledgements

The TaMaDi project of the ANR

Project leader: Jean-Michel Muller

Project members: Jean-Claude Bajard, Nicolas Brisebarre, Florent de Dinechin, Pierre Fortin, Mourad Gouicem, Stef Graillat, Guillaume Hanrot, Thibault Hilaire, Mioara Joldeș, Christoph Lauter, Vincent Lefèvre, Érik Martin-Dorel, Micaela Mayero, Marc Mezzarobba, Jean-Michel Muller, Andy Novocin, Ioana Pașca, Laurence Rideau, Damien Stehlé, Laurent Théry, Serge Torres.

Acknowledgements

The TaMaDi project of the ANR

Project leader: Jean-Michel Muller

Project members: Jean-Claude Bajard, Nicolas Brisebarre, Florent de Dinechin, Pierre Fortin, Mourad Gouicem, Stef Graillat, Guillaume Hanrot, Thibault Hilaire, Mioara Joldeș, Christoph Lauter, Vincent Lefèvre, Érik Martin-Dorel, Micaela Mayero, Marc Mezzarobba, Jean-Michel Muller, Andy Novocin, Ioana Pașca, Laurence Rideau, Damien Stehlé, Laurent Théry, Serge Torres.

Correct rounding

The IEEE 754–2008 standard for floating-point (FP) arithmetic requires correct rounding for $+$, $-$, $\times$, $\div$, $\sqrt{\cdot}$

Correct rounding

The IEEE 754–2008 standard for floating-point (FP) arithmetic requires correct rounding for $+$, $-$, $\times$, $\div$, $\sqrt{\cdot}$

A correctly-rounded operation whose entries are FP numbers must return what we would get by infinitely-precise operation, followed by rounding.

Correct rounding

The IEEE 754–2008 standard for floating-point (FP) arithmetic requires correct rounding for $+$, $-$, $\times$, $\div$, $\sqrt{\cdot}$

A correctly-rounded operation whose entries are FP numbers must return what we would get by infinitely-precise operation, followed by rounding.

Advantages: greatly improves accuracy, portability, as well as provability: one can devise algorithms and proofs that use the specifications.

Correct rounding

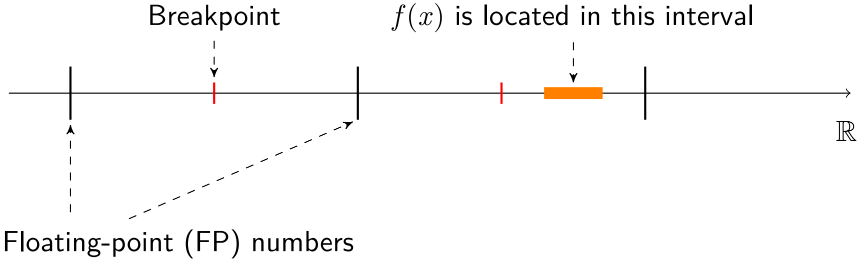
The IEEE 754–2008 standard for floating-point (FP) arithmetic requires correct rounding for $+$, $-$, $\times$, $\div$, $\sqrt{\cdot}$

A correctly-rounded operation whose entries are FP numbers must return what we would get by infinitely-precise operation, followed by rounding.

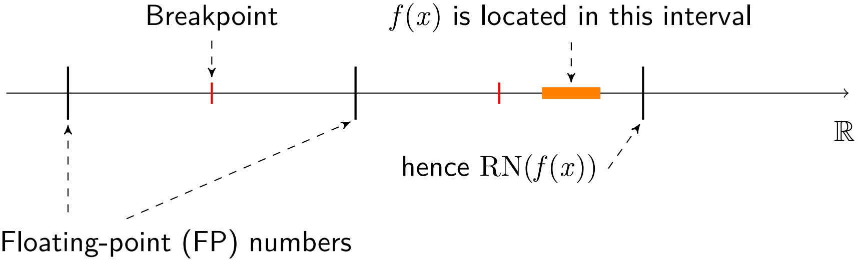
Advantages: greatly improves accuracy, portability, as well as provability: one can devise algorithms and proofs that use the specifications.

IEEE 754–2008 only recommends correct rounding for elementary functions $(\exp, \sin, \dots)$ $\Rightarrow$ solve the Table Maker's Dilemma for each function.

The Table Maker's Dilemma (TMD) in rounding-to-nearest



The Table Maker's Dilemma (TMD) in rounding-to-nearest



Breakpoint

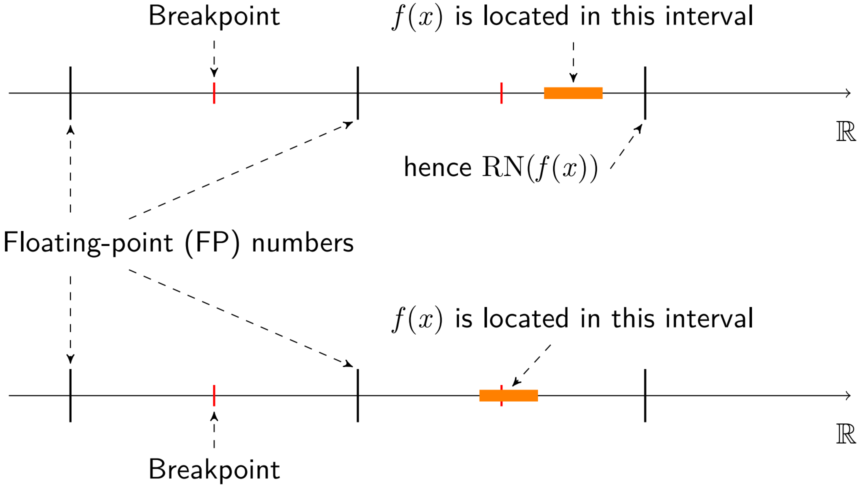
$f(x)$ is located in this interval

$\mathbb{R}$

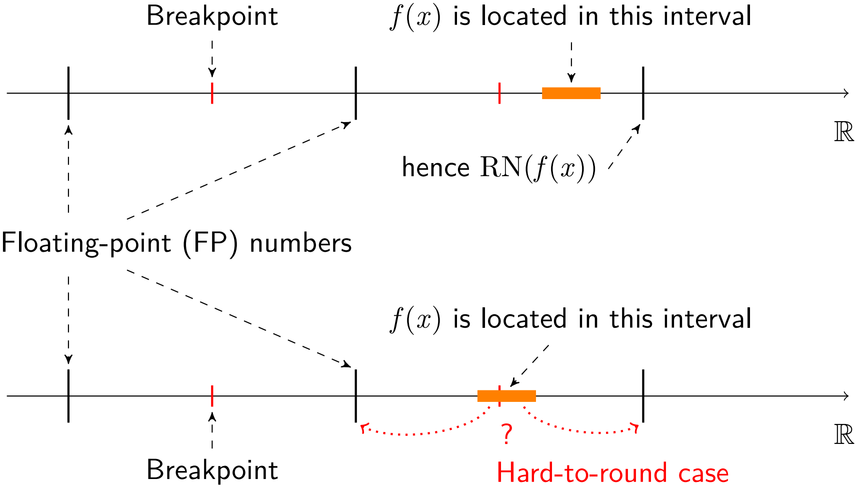
hence $\mathrm{RN}(f(x))$

Floating-point (FP) numbers

The Table Maker's Dilemma (TMD) in rounding-to-nearest



The Table Maker's Dilemma (TMD) in rounding-to-nearest



The Table Maker's Dilemma (TMD) (continued)

Solving the TMD $=$ find the hardest-to-round cases of f: the FP values x such that $f(x)$ is closest to a breakpoint without being a breakpoint.

The Table Maker's Dilemma (TMD) (continued)

Solving the TMD = find the hardest-to-round cases of f: the FP values x such that $f(x)$ is closest to a breakpoint without being a breakpoint.

The hardest-to-round case of $\exp$ for decimal64 and rounding-to-nearest is

$$x = 9.407822313572878 \times 10^{-2}$$

The Table Maker's Dilemma (TMD) (continued)

Solving the TMD $=$ find the hardest-to-round cases of f: the FP values x such that $f(x)$ is closest to a breakpoint without being a breakpoint.

The hardest-to-round case of $\exp$ for decimal64 and rounding-to-nearest is

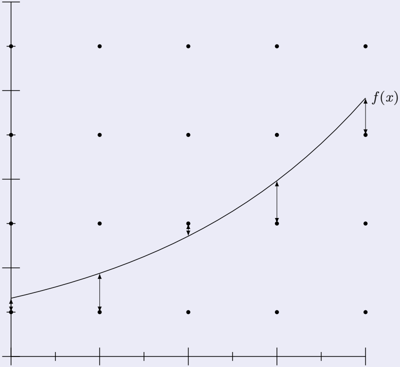
$$x = 9.407822313572878 \times 10^{-2}$$

$$\exp(x) = 1.098645682066338\ 5\ {\color{red}0000000000000000}\ 278\ldots$$

Computation of lists of hard-to-round cases

Finding all the hard-to-round cases of f over $\boldsymbol{I}$ with respect to $\epsilon > 0$

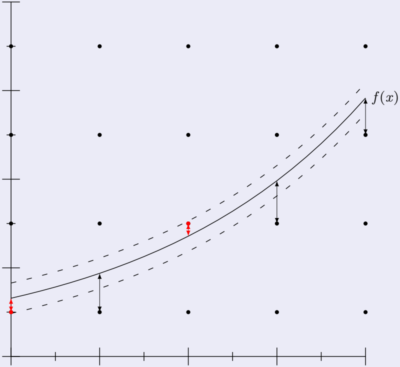
$$\left\{x \in \mathbb{F} \cap \boldsymbol{I} \;\middle/\; \left|\left(\frac{f(x)}{\mathrm{ulp}(f(x))} - \frac{1}{2}\right) \operatorname{cmod} 1\right| \leqslant \epsilon\right\}.$$



Computation of lists of hard-to-round cases

Finding all the hard-to-round cases of f over $\boldsymbol{I}$ with respect to $\epsilon > 0$

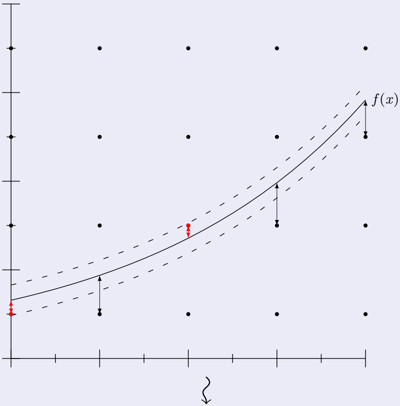
$$\left\{x \in \mathbb{F} \cap \boldsymbol{I} \;\Big/\; \left|\left(\frac{f(x)}{\mathrm{ulp}(f(x))} - \frac{1}{2}\right) \mathrm{cmod}\ 1\right| \leqslant \epsilon\right\}.$$



Computation of lists of hard-to-round cases

Finding all the hard-to-round cases of f over $\boldsymbol{I}$ with respect to $\epsilon > 0$

$$\left\{x \in \mathbb{F} \cap \boldsymbol{I} \;\middle/\; \left|\left(\frac{f(x)}{\mathrm{ulp}(f(x))} - \frac{1}{2}\right) \operatorname{cmod} 1\right| \leqslant \epsilon\right\}.$$



Integer Small Value Pbm $(P \in \mathbb{Z}[X], A, B, M \in \mathbb{Z}) = \{x \in [\![-A, A]\!] \,/\, |P(x) \operatorname{cmod} M| \leqslant B\}$.

The Stehlé–Lefèvre–Zimmermann (SLZ) algorithm

SLZ = Polynomial Approximation + Coppersmith's technique

+ Bivariate Hensel lifting

The Stehlé–Lefèvre–Zimmermann (SLZ) algorithm

Critical part of the algorithm

SLZ = Polynomial Approximation + Coppersmith's technique
+ Bivariate Hensel lifting

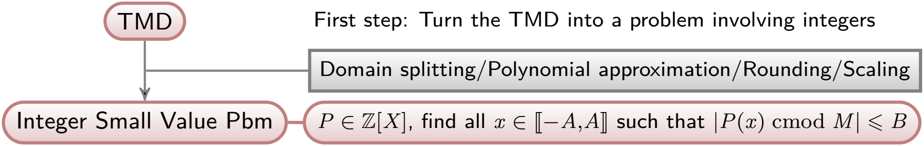
- Sophisticated algorithms with highly optimized implementations
- Rely on many tools and libraries (SAGE, MPIR, FLINT, fpLLL, . . .)
- Very long calculations (several years of CPU time)

The SLZ algorithm (continued)

TMD

First step: Turn the TMD into a problem involving integers

The SLZ algorithm (continued)



TMD

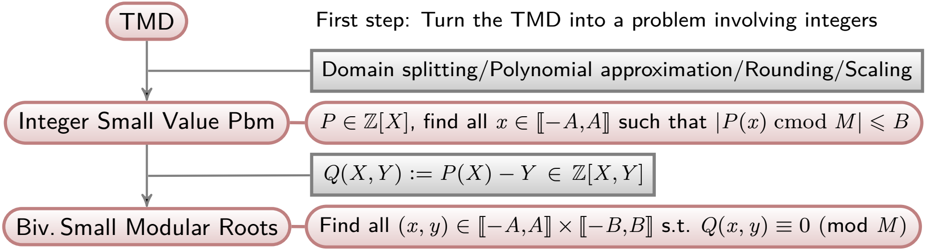
First step: Turn the TMD into a problem involving integers

Domain splitting/Polynomial approximation/Rounding/Scaling

Integer Small Value Pbm

$P \in \mathbb{Z}[X]$, find all $x \in [\![-A, A]\!]$ such that $|P(x) \operatorname{cmod} M| \leqslant B$

The SLZ algorithm (continued)



TMD

First step: Turn the TMD into a problem involving integers

Domain splitting/Polynomial approximation/Rounding/Scaling

Integer Small Value Pbm

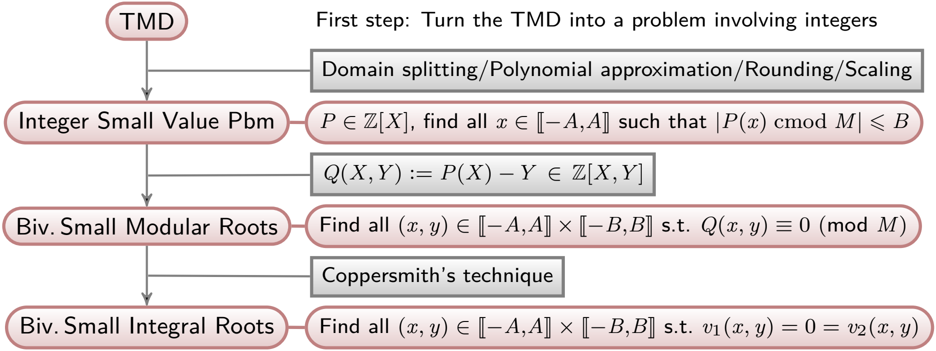
$P \in \mathbb{Z}[X]$, find all $x \in \llbracket -A, A \rrbracket$ such that $|P(x) \operatorname{cmod} M| \leqslant B$

$Q(X,Y) := P(X) - Y \ \in \mathbb{Z}[X,Y]$

Biv. Small Modular Roots

Find all $(x,y) \in \llbracket -A, A \rrbracket \times \llbracket -B, B \rrbracket$ s.t. $Q(x,y) \equiv 0 \pmod{M}$

The SLZ algorithm (continued)



TMD

First step: Turn the TMD into a problem involving integers

Domain splitting/Polynomial approximation/Rounding/Scaling

Integer Small Value Pbm

$P \in \mathbb{Z}[X]$, find all $x \in \llbracket -A,A \rrbracket$ such that $|P(x) \operatorname{cmod} M| \leqslant B$

$Q(X,Y) := P(X) - Y \in \mathbb{Z}[X,Y]$

Biv. Small Modular Roots

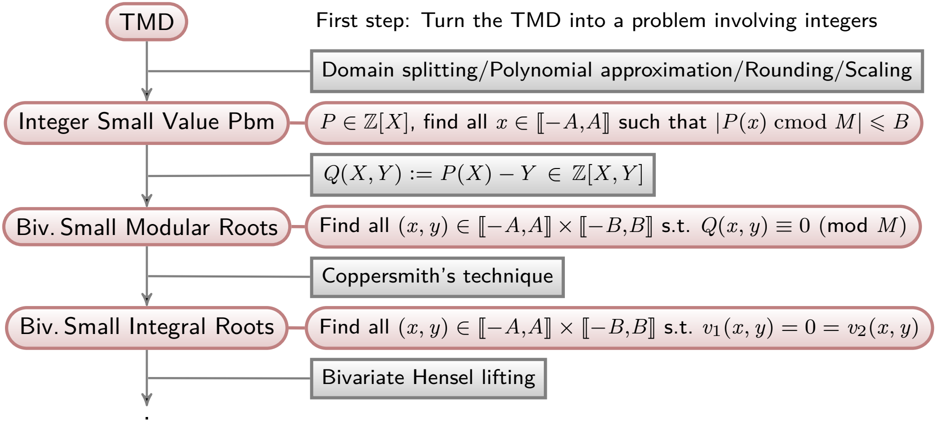
Find all $(x,y) \in \llbracket -A,A \rrbracket \times \llbracket -B,B \rrbracket$ s.t. $Q(x,y) \equiv 0 \pmod{M}$

Coppersmith's technique

Biv. Small Integral Roots

Find all $(x,y) \in \llbracket -A,A \rrbracket \times \llbracket -B,B \rrbracket$ s.t. $v_1(x,y) = 0 = v_2(x,y)$

The SLZ algorithm (continued)



First step: Turn the TMD into a problem involving integers

TMD

Domain splitting/Polynomial approximation/Rounding/Scaling

Integer Small Value Pbm

$P \in \mathbb{Z}[X]$, find all $x \in [\![-A,A]\!]$ such that $|P(x) \operatorname{cmod} M| \leqslant B$

$Q(X,Y) := P(X) - Y \in \mathbb{Z}[X,Y]$

Biv. Small Modular Roots

Find all $(x,y) \in [\![-A,A]\!] \times [\![-B,B]\!]$ s.t. $Q(x,y) \equiv 0 \pmod{M}$

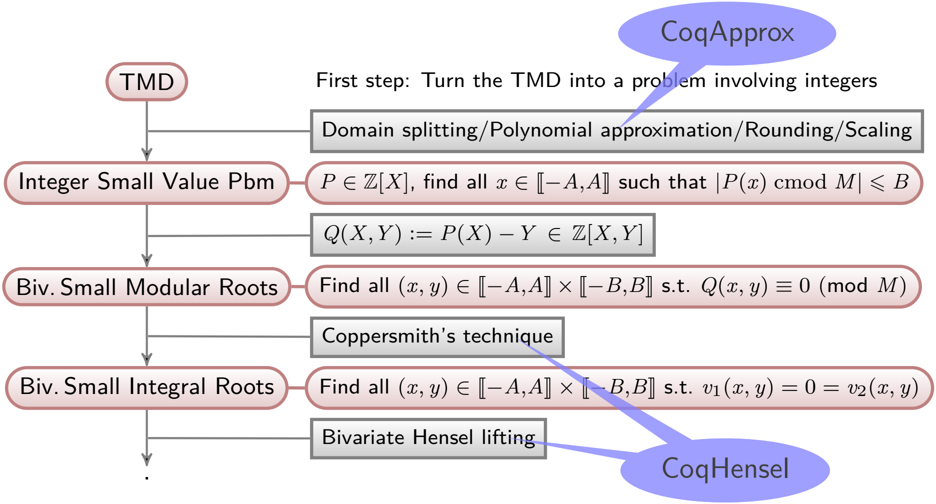
Coppersmith's technique

Biv. Small Integral Roots

Find all $(x,y) \in [\![-A,A]\!] \times [\![-B,B]\!]$ s.t. $v_1(x,y) = 0 = v_2(x,y)$

Bivariate Hensel lifting

The SLZ algorithm (continued)



First step: Turn the TMD into a problem involving integers

TMD

Domain splitting/Polynomial approximation/Rounding/Scaling

Integer Small Value Pbm

$P \in \mathbb{Z}[X]$, find all $x \in [\![-A,A]\!]$ such that $|P(x) \operatorname{cmod} M| \leqslant B$

$Q(X,Y) := P(X) - Y \in \mathbb{Z}[X,Y]$

Biv. Small Modular Roots

Find all $(x,y) \in [\![-A,A]\!] \times [\![-B,B]\!]$ s.t. $Q(x,y) \equiv 0 \pmod{M}$

Coppersmith's technique

Biv. Small Integral Roots

Find all $(x,y) \in [\![-A,A]\!] \times [\![-B,B]\!]$ s.t. $v_1(x,y) = 0 = v_2(x,y)$

Bivariate Hensel lifting

Outline

CoqApprox: Context and motivations

Goal

Compute polynomial approximations of univariate functions along with certified error bounds: for a given function f over an interval[a] $\boldsymbol{I}$, compute P, ϵ and formally prove that $\forall x \in \boldsymbol{I},\ |f(x) - P(x)| \leqslant \epsilon$

---

[a] Intervals are printed in bold.

CoqApprox: Context and motivations

Goal

Compute polynomial approximations of univariate functions along with certified error bounds: for a given function f over an interval[a] $\boldsymbol{I}$, compute P, ϵ and formally prove that $\forall x \in \boldsymbol{I},\ |f(x) - P(x)| \leqslant \epsilon$

---

[a] Intervals are printed in bold.

Example

For $f(x) = \sin x$ over $\boldsymbol{I} = [-1, 1]$ and a target accuracy of 2^{-400}, we can compute an order-80 Taylor expansion of f around 0 for the polynomial P and take $\epsilon = 1.79 \times 2^{-402}$.

CoqApprox: Mathematical setup

Data structure

A rigorous polynomial approximation (RPA) is a pair $(P, \mathbf{\Delta})$ where P is a polynomial in a given basis, and $\mathbf{\Delta}$ an interval. Typical examples of RPAs are Taylor Models (TMs) and Chebyshev Models (CMs).

CoqApprox: Mathematical setup

Data structure

A rigorous polynomial approximation (RPA) is a pair $(P, \mathbf{\Delta})$ where P is a polynomial in a given basis, and $\mathbf{\Delta}$ an interval. Typical examples of RPAs are Taylor Models (TMs) and Chebyshev Models (CMs).

Methodology

1. For basic[a] functions: rely on the Taylor–Lagrange formula.
2. For composite functions, we define some "arithmetic rules" for addition, multiplication, composition, and division.
   E.g.: if $(P_1, \mathbf{\Delta}_1)$ is a TM of f_1 and $(P_2, \mathbf{\Delta}_2)$ is a TM of f_2, then $(P_1, \mathbf{\Delta}_1) \oplus (P_2, \mathbf{\Delta}_2) := (P_1 + P_2, \mathbf{\Delta}_1 + \mathbf{\Delta}_2)$ is a TM for $f_1 + f_2$.

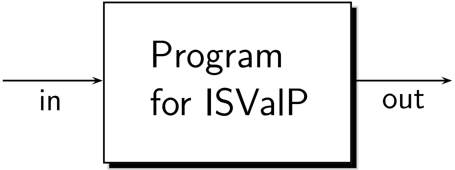
---

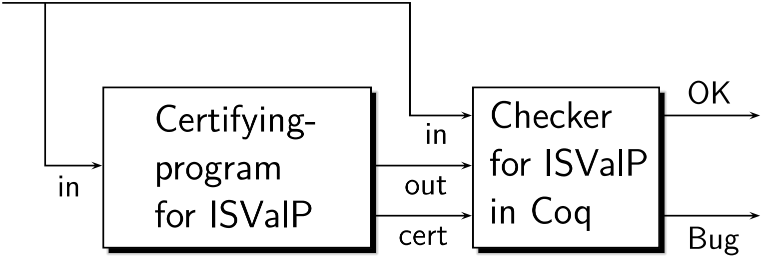
[a] We focus on D-finite (*aka* holonomic) functions, i.e., solutions of homogeneous linear ordinary differential equations with polynomial coefficients.

CoqApprox: Formalization and machine-checked proofs

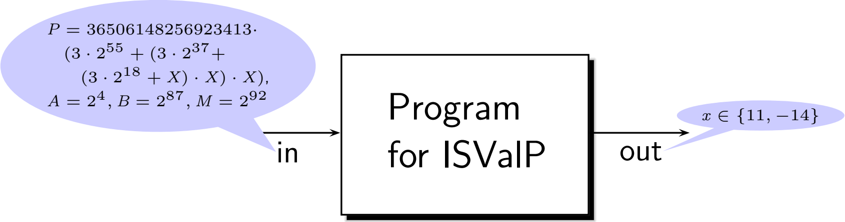
- Libraries used: Ssreflect [MathComponents], CoqInterval [Melquiond]
- Efficiency: on the whole, the timings of the Coq implementation have the same order of magnitude as that of the C implementation provided in Sollya [Chevillard, Joldeș, Lauter]
- Sharp bounds: thanks to the implemented algorithm called Zumkeller's technique, the approximation of basic functions leads to sharp bounds in practice.

From SLZ to certificates for Integer Small Value Problems





From SLZ to certificates for Integer Small Value Problems



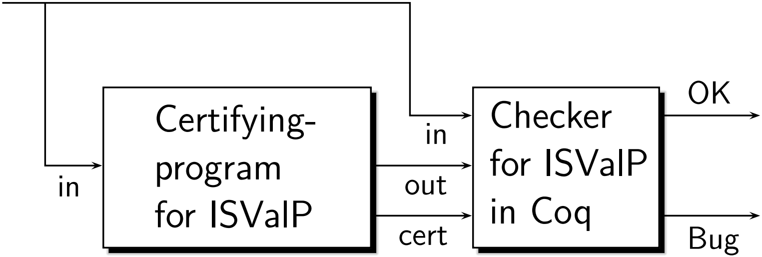
$P = 36506148256923413 \cdot (3 \cdot 2^{55} + (3 \cdot 2^{37} + (3 \cdot 2^{18} + X) \cdot X) \cdot X)$, $A = 2^4$, $B = 2^{87}$, $M = 2^{92}$

in

Program for ISValP

out

$x \in \{11, -14\}$



in

Certifying-program for ISValP

in

out

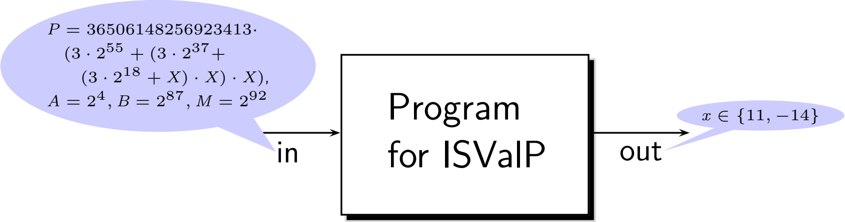
cert

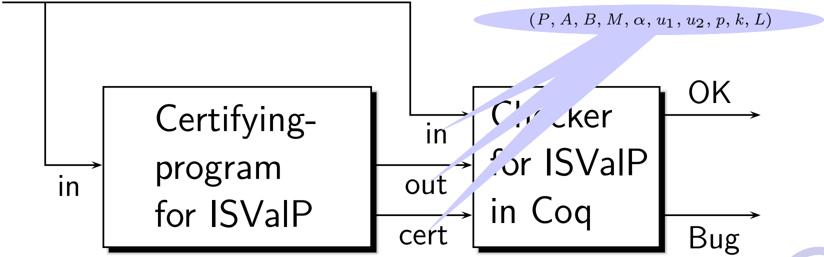
Checker for ISValP in Coq

OK

Bug

From SLZ to certificates for Integer Small Value Problems





A verified checker for the Integer Small Value Problem

Theorem

For any certificate $(P, A, B, M, \alpha, u_1, u_2, p, k, L)$ that is accepted, we have

$$\forall (x, y) \in [\![-A, A]\!] \times [\![-B, B]\!],\ P(x) \equiv y \pmod{M} \implies (x, y) \in L.$$

The Coq proof required the formalization of several mathematical notions:

- Taylor's theorem for bivariate polynomials,
- Hensel's lemma for pairs of bivariate polynomials,
- properties of the weighted norm-1 of a bivariate polynomial,

...

Challenges and methodology

Formalizing efficient computation in a proof assistant is often challenging.

- Approach by refinement: first, prove an abstract version of the algorithms, then refine them to effective implementations that are proved correct w.r.t. the abstract version.

Challenges and methodology

Formalizing efficient computation in a proof assistant is often challenging.

- Approach by refinement: first, prove an abstract version of the algorithms, then refine them to effective implementations that are proved correct w.r.t. the abstract version.
- Approach by certificates: rather than formally verifying the correctness of an optimized implementation of a complex algorithm such as LLL, generate some “logs” of its execution and check them independently.

Milestone

We can compute formally verified Taylor Models for the following D-finite functions: $x \mapsto \frac{1}{x}$, $\sqrt{\cdot}$, $\frac{1}{\sqrt{\cdot}}$, $\exp$, $\sin$, $\cos$,
and the algorithms to compute Taylor Models for composite functions (involving the operations $+$, $\times$, $\circ$, $\div$) have also been formally verified.

Milestone

We can compute formally verified Taylor Models for the following D-finite functions: $x \mapsto \frac{1}{x}$, $\sqrt{\cdot}$, $\frac{1}{\sqrt{\cdot}}$, $\exp$, $\sin$, $\cos$,
and the algorithms to compute Taylor Models for composite functions (involving the operations $+$, $\times$, $\circ$, $\div$) have also been formally verified.

Test-suite for CoqHensel:

- 4096 ISValP certificates to address an exponent of $\exp$ in binary64: $n := 53$, $n' := 300$, $\alpha := 13$, $\approx 100\,$MB of data. The generation of each certificate takes $\approx 140\,$s, and the verification in Coq takes $\approx 35\,$s (using `native_compute` with "`bigZ` $\times$ `bigN`" integers).

Milestone

We can compute formally verified Taylor Models for the following D-finite functions: $x \mapsto \frac{1}{x}$, $\sqrt{\cdot}$, $\frac{1}{\sqrt{\cdot}}$, $\exp$, $\sin$, $\cos$,
and the algorithms to compute Taylor Models for composite functions (involving the operations $+$, $\times$, $\circ$, $\div$) have also been formally verified.

Test-suite for CoqHensel:

- 4096 ISValP certificates to address an exponent of $\exp$ in binary64: $n := 53$, $n' := 300$, $\alpha := 13$, ≈ 100 MB of data. The generation of each certificate takes ≈ 140 s, and the verification in Coq takes ≈ 35 s (using `native_compute` with "`bigZ` × `bigN`" integers).
- one ISValP certificate for $\exp$ in binary128: $n := 113$, $n' := 3000$, $\alpha := 6$. Verified by Coq in $1\,041$ s (vs. 56 s in Maple 15).

Future work

Short-term perspectives:

- Implement faster algorithms for the operations of polynomials.
- Combine CoqHensel & CoqApprox to devise a complete certificate checker for the TMD.
- Implement and prove more functions in CoqApprox $(\cosh, \tan, \dots)$

Future work

Short-term perspectives:

- Implement faster algorithms for the operations of polynomials.
- Combine CoqHensel & CoqApprox to devise a complete certificate checker for the TMD.
- Implement and prove more functions in CoqApprox $(\cosh, \tan, \dots)$

Long-term perspectives:

- Combine TMs with some polynomial global optimization technique.

Future work

Short-term perspectives:

- Implement faster algorithms for the operations of polynomials.
- Combine CoqHensel & CoqApprox to devise a complete certificate checker for the TMD.
- Implement and prove more functions in CoqApprox $(\cosh, \tan, \dots)$

Long-term perspectives:

- Combine TMs with some polynomial global optimization technique.
- Implement Chebyshev Models $\rightsquigarrow$ tighter remainders.

Future work

Short-term perspectives:

- Implement faster algorithms for the operations of polynomials.
- Combine CoqHensel & CoqApprox to devise a complete certificate checker for the TMD.
- Implement and prove more functions in CoqApprox $(\cosh, \tan, \ldots)$

Long-term perspectives:

- Combine TMs with some polynomial global optimization technique.
- Implement Chebyshev Models $\rightsquigarrow$ tighter remainders.
- Consider the possible generalization to the multivariate case.

Future work

Short-term perspectives:

- Implement faster algorithms for the operations of polynomials.
- Combine CoqHensel & CoqApprox to devise a complete certificate checker for the TMD.
- Implement and prove more functions in CoqApprox $(\cosh, \tan, \ldots)$

Long-term perspectives:

- Combine TMs with some polynomial global optimization technique.
- Implement Chebyshev Models $\rightsquigarrow$ tighter remainders.
- Consider the possible generalization to the multivariate case.
- Consider alternative techniques for verifying error bounds
  - fixed-point theorems?
  - majorant series?

Future work

Short-term perspectives:

- Implement faster algorithms for the operations of polynomials.
- Combine CoqHensel & CoqApprox to devise a complete certificate checker for the TMD.
- Implement and prove more functions in CoqApprox $(\cosh, \tan, \dots)$

Long-term perspectives:

- Combine TMs with some polynomial global optimization technique.
- Implement Chebyshev Models $\leadsto$ tighter remainders.
- Consider the possible generalization to the multivariate case.
- Consider alternative techniques for verifying error bounds
  - fixed-point theorems?
  - majorant series?
- On-going works: formal proof of Lefèvre's algorithm.

End of the talk

Thank you for your attention!

The TaMaDi homepage:
https://tamadiwiki.ens-lyon.fr/

Certificates to address the Integer Small Value Problem

```
Record cert_ISValP :=
{ $P$ : {poly int}   (* hence $Q(X,Y) = P(Y) - X$ *)
; $A$ : nat          (* bound related to the TMD accuracy *)
; $B$ : nat          (* bound related to the domain range *)
; $M$ : nat          (* the modulo *)
; $\alpha$ : nat          (* the Coppersmith parameter *)
; $u_1$ : {bipoly int} (* in basis $M^{\alpha-i} \times Q^i(X,Y) \times Y^j$ *)
; $u_2$ : {bipoly int} (* in basis $M^{\alpha-i} \times Q^i(X,Y) \times Y^j$ *)
; $p$ : nat          (* prime used by Hensel lifting *)
; $k$ : nat          (* number of iterations *)
; $L$ : list (int * int * bool) (* list of solutions *)
}.

Definition check_ISValP : cert_ISValP -> bool.
```

Back