Interval Arithmetic, Real Analysis, and Formal Proofs

Guillaume Melquiond

Inria & LRI, Université Paris Sud, CNRS

2016-11-17

Direct and Indirect Contributors

Y. Bertot, S. Boldo, O. Desmettre, G. Gonthier, M. Joldeș, C. Lelay, A. Mahboubi, M. Mayero, É. Martin-Dorel, I. Pașca, L. Rideau, T. Sibut-Pinote, A. Spiwack, L. Théry

and many other people I am presumably forgetting.

More than 200k lines of human-written formal proofs.

Motivation 1: Formal Verification of Math Libraries

Cody & Waite's algorithm (1980)

```
double cw_exp(double x)
{
  // exception handling and constants
  ...
  // argument reduction
  double k = nearbyint(x * InvLog2);
  double t = x - k * Log2h - k * Log2l;
  // polynomial approximation
  double t2 = t * t;
  double p = 0.25 + t2 * (p1 + t2 * p2);
  double q = 0.5  + t2 * (q1 + t2 * q2);
  double f = t * (p / (q - t * p)) + 0.5;
  // result reconstruction
  return ldexp(f, (int)k + 1);
}
```

This floating-point function accurately approximates exp.

Motivation 2: Numerical Integrals in Modern Math Proofs

Double bubbles minimize (2000)

The proof parameterizes the space of possible solutions by a two-dimensional rectangle [...]. This rectangle is subdivided into 15,016 smaller rectangles which are investigated by calculations involving a total of 51,256 numerical integrals.

Motivation 2: Numerical Integrals in Modern Math Proofs

Double bubbles minimize (2000)

The proof parameterizes the space of possible solutions by a two-dimensional rectangle [...]. This rectangle is subdivided into 15,016 smaller rectangles which are investigated by calculations involving a total of 51,256 numerical integrals.

Major arcs for Goldbach's problem (2013)

$$\int_{-\infty}^{\infty} \frac{(0.5 \cdot \log(\tau^2 + 2.25) + 4.1396 + \log \pi)^2}{0.25 + \tau^2} d\tau$$

We compute the last integral numerically (from -100,000 to 100,000).

mathoverflow

Questions Tags Users Badges Una

Rigorous numerical integration

9

2

I need to evaluate some (one-variable) integrals that neither SAGE nor Mathematica can do symbolically. As far as I can tell, I have two options:

(a) Use GSL (via SAGE), Maxima or Mathematica to do numerical integration. This is really a non-option, since, if I understand correctly, the "error bound" they give is not really a guarantee.

(b) Cobble together my own programs using the trapezoidal rule, Simpson's rule, etc., and get rigorous error bounds using bounds I have for the second (or fourth, or what have you) derivative of the function I am integrating. This is what I have been doing.

Is there a third option? Is there standard software that does (b) for me?

na.numerical-analysis

share cite improve this question

asked Mar 5 '13 at 23:03
H A Helfgott
3,141 • 17 • 61

Introduction

Objective

Formally verify inequalities on real-valued expressions.

Methodology

- define reliable yet efficient algorithms,
- formally prove that they are correct,
- execute them inside the Coq formal system.

Outline

Outline

1. Introduction
   - Motivations
   - The Coq proof assistant
   - The CoqInterval library
2. Formalizing the arithmetic
3. Fighting the dependency effect
4. Numerical integration
5. Conclusion

Coq: a Proof Assistant

Coq in a nutshell

- typed lambda-calculus with inductive types,
- proof verification using a "small" kernel,
- proof assistance using tactic-based backward reasoning.

Coq: a Proof Assistant

Coq in a nutshell

- typed lambda-calculus with inductive types,
- proof verification using a “small” kernel,
- proof assistance using tactic-based backward reasoning.

Stating and proving $\frac{ab}{ac} = \frac{b}{c}$

```
Lemma Rdiv_compat_r : (* stating the theorem *)
  forall a b c : R,
  a <> 0 -> c <> 0 -> (a*b) / (a*c) = b/c.
Proof. (* building the proof using tactics *)
  intros.
  field.
  easy.
Qed. (* verifying the resulting proof *)
```

Automating Proofs using CoqInterval

CoqInterval in a nutshell

Formally-verified enclosures of real-valued expressions using

- basic arithmetic operators: $+$, $-$, $\times$, $\div$, $\sqrt{\cdot}$,
- elementary functions: cos, sin, tan, arctan, exp, log,
- univariate integrals.

Automating Proofs using CoqInterval

CoqInterval in a nutshell

Formally-verified enclosures of real-valued expressions using

- basic arithmetic operators: $+$, $-$, $\times$, $\div$, $\sqrt{\cdot}$,
- elementary functions: cos, sin, tan, arctan, exp, log,
- univariate integrals.

Stating and proving $\sqrt{\exp x} \le \pi \cdot \int_1^4 \log t \, dt$ when $x \le 3$

```
Lemma whatever :
  forall x : R, x <= 3 ->
  sqrt (exp x) <= PI * (RInt ln 1 4).
Proof.
  intros.
  interval.
Qed.
```

Formalization Scope

Components needed to get the `interval` tactic

- integer and real arithmetic — Stdlib

Formalization Scope

Components needed to get the `interval` tactic

- integer and real arithmetic — Stdlib
- floating-point arithmetic — Flocq, CoqInterval

Formalization Scope

Components needed to get the `interval` tactic

- integer and real arithmetic — Stdlib
- floating-point arithmetic — Flocq, CoqInterval
- real analysis — Stdlib, Coquelicot

Formalization Scope

Components needed to get the `interval` tactic

- integer and real arithmetic — Stdlib
- floating-point arithmetic — Flocq, CoqInterval
- real analysis — Stdlib, Coquelicot
- floating-point elementary functions — CoqInterval

Formalization Scope

Components needed to get the `interval` tactic

integer and real arithmetic	Stdlib
floating-point arithmetic	Flocq, CoqInterval
real analysis	Stdlib, Coquelicot
floating-point elementary functions	CoqInterval
interval arithmetic	CoqInterval

Formalization Scope

Components needed to get the `interval` tactic

• integer and real arithmetic	Stdlib
• floating-point arithmetic	Flocq, CoqInterval
• real analysis	Stdlib, Coquelicot
• floating-point elementary functions	CoqInterval
• interval arithmetic	CoqInterval
• automatic differentiation and Taylor models	CoqInterval

Formalization Scope

Components needed to get the `interval` tactic

- integer and real arithmetic — Stdlib
- floating-point arithmetic — Flocq, CoqInterval
- real analysis — Stdlib, Coquelicot
- floating-point elementary functions — CoqInterval
- interval arithmetic — CoqInterval
- automatic differentiation and Taylor models — CoqInterval
- fast integer arithmetic — Stdlib

Formalization Scope

Components needed to get the `interval` tactic

- integer and real arithmetic — Stdlib
- floating-point arithmetic — Flocq, CoqInterval
- real analysis — Stdlib, Coquelicot
- floating-point elementary functions — CoqInterval
- interval arithmetic — CoqInterval
- automatic differentiation and Taylor models — CoqInterval
- fast integer arithmetic — Stdlib

Everything is formalized in Coq logic.

Formalization Scope

Components needed to get the `interval` tactic

- integer and real arithmetic — Stdlib
- floating-point arithmetic — Flocq, CoqInterval
- real analysis — Stdlib, Coquelicot
- floating-point elementary functions — CoqInterval
- interval arithmetic — CoqInterval
- automatic differentiation and Taylor models — CoqInterval
- fast integer arithmetic — Stdlib

Everything is formalized in Coq logic.

Real arithmetic and analysis are not constructive
but we don't want to extract anything from the proofs anyway.

Outline

Arithmetic Datatypes

Positive integer

- list of ones (unary representation),
- list of bits (binary representation),
- balanced binary tree of fixed-size integers.

Arithmetic Datatypes

Positive integer

- list of ones (unary representation),
- list of bits (binary representation),
- balanced binary tree of fixed-size integers.

Floating-point number: $\mathbb{F} = \mathbb{Z}^2 \cup \{\bot_\mathbb{F}\}$

no overflow nor underflow, arbitrary precision;
(m, e) is interpreted as the real number $m \cdot \beta^e$.

Arithmetic Datatypes

Positive integer

- list of ones (unary representation),
- list of bits (binary representation),
- balanced binary tree of fixed-size integers.

Floating-point number: $\mathbb{F} = \mathbb{Z}^2 \cup \{\bot_\mathbb{F}\}$

no overflow nor underflow, arbitrary precision;
(m, e) is interpreted as the real number $m \cdot \beta^e$.

Interval: $\mathbb{I} = \mathbb{F}^2 \cup \{\bot_\mathbb{I}\}$

inf-sup representation; a bound $\bot_\mathbb{F}$ is interpreted as infinite.

Arithmetic Datatypes

Positive integer

- list of ones (unary representation),
- list of bits (binary representation),
- balanced binary tree of fixed-size integers.

Floating-point number: $\mathbb{F} = \mathbb{Z}^2 \cup \{\perp_\mathbb{F}\}$

no overflow nor underflow, arbitrary precision;
(m, e) is interpreted as the real number $m \cdot \beta^e$.

Interval: $\mathbb{I} = \mathbb{F}^2 \cup \{\perp_\mathbb{I}\}$

inf-sup representation; a bound $\perp_\mathbb{F}$ is interpreted as infinite.

Real number

$\mathbb{R}$ is an abstract type.

Operations and Specifications

Floating-point arithmetic operations

$\mathbb{F}\texttt{sqrt} : \mathit{mode}, \mathit{prec}, \mathbb{F} \to \mathbb{F}.$

$$\forall x \in \mathbb{F},\ \mathbb{F}\texttt{to}\mathbb{R}(\mathbb{F}\texttt{sqrt}(m, p, x)) = \texttt{round}(m, p, \sqrt{\mathbb{F}\texttt{to}\mathbb{R}(x)}).$$

Operations and Specifications

Floating-point arithmetic operations

$\mathbb{F}\texttt{sqrt} : \mathit{mode}, \mathit{prec}, \mathbb{F} \to \mathbb{F}.$

$$\forall x \in \mathbb{F},\ \mathbb{F}\texttt{to}\mathbb{R}(\mathbb{F}\texttt{sqrt}(m, p, x)) = \texttt{round}(m, p, \sqrt{\mathbb{F}\texttt{to}\mathbb{R}(x)}).$$

Floating-point elementary functions

$\mathbb{F}\texttt{log} : \mathit{prec}, \mathbb{F} \to \mathbb{I}.$

$$\forall x \in \mathbb{F},\ \log(\mathbb{F}\texttt{to}\mathbb{R}(x)) \in \mathbb{F}\texttt{log}(p, x).$$

Operations and Specifications

Floating-point arithmetic operations

$\mathbb{F}\texttt{sqrt} : \mathit{mode}, \mathit{prec}, \mathbb{F} \to \mathbb{F}$.

$$\forall x \in \mathbb{F},\ \mathbb{F}\texttt{to}\mathbb{R}(\mathbb{F}\texttt{sqrt}(m, p, x)) = \texttt{round}(m, p, \sqrt{\mathbb{F}\texttt{to}\mathbb{R}(x)}).$$

Floating-point elementary functions

$\mathbb{F}\texttt{log} : \mathit{prec}, \mathbb{F} \to \mathbb{I}$.

$$\forall x \in \mathbb{F},\ \log(\mathbb{F}\texttt{to}\mathbb{R}(x)) \in \mathbb{F}\texttt{log}(p, x).$$

Interval operations

$\mathbb{I}\texttt{sin} : \mathit{prec}, \mathbb{I} \to \mathbb{I}$.

$$\forall \mathbf{x} \in \mathbb{I},\ \forall x \in \mathbb{R},\ x \in \mathbf{x} \Rightarrow \sin(x) \in \mathbb{I}\texttt{sin}(p, \mathbf{x}).$$

Implementation of an Elementary Function

Implementation of $\mathbb{F}\texttt{log}$

- If $x < 1$, use $\log x = -\log(x^{-1})$.
- While $x > 1 + 2^{-8}$, use $\log x = 2 \log \sqrt{x}$.
- Evaluate the alternated series with interval operations

$$\log(1 + t) = t - t^2/2 + t^3/3 - \dots$$

until the remainder satisfies the target accuracy.

Implementation of an Elementary Function

Implementation of $\mathbb{F}\texttt{log}$

- If $x < 1$, use $\log x = -\log(x^{-1})$.
- While $x > 1 + 2^{-8}$, use $\log x = 2 \log \sqrt{x}$.
- Evaluate the alternated series with interval operations

$$\log(1+t) = t - t^2/2 + t^3/3 - \ldots$$

until the remainder satisfies the target accuracy.

This is a poor way of approximating log,
but at least $\log x \in \mathbb{F}\texttt{log}(p, x)$ is formally proved.

Outline

Fighting the Dependency Effect

Dependency effect

Interval arithmetic might compute overestimated enclosures if there are multiple occurrences of variables:

$\forall x \in \mathbf{x} = [-1; 1],\ \sin x - x \in [-0.2; 0.2]$,
yet $\sin \mathbf{x} - \mathbf{x} \subseteq [-1.9; 1.9]$.

Fighting the Dependency Effect

Dependency effect

Interval arithmetic might compute overestimated enclosures if there are multiple occurrences of variables:

$\forall x \in \mathbf{x} = [-1; 1],\ \sin x - x \in [-0.2; 0.2]$,
yet $\sin \mathbf{x} - \mathbf{x} \subseteq [-1.9; 1.9]$.

Definition (Polynomial enclosure)

$(P, \mathbf{\Delta}) \in \mathbb{R}[X] \times \mathbb{I}$ encloses f on $\mathbf{x} \ni x_0$ if

$$\forall x \in \mathbf{x},\ f(x) - P(x - x_0) \in \mathbf{\Delta}.$$

Fighting the Dependency Effect

Dependency effect

Interval arithmetic might compute overestimated enclosures if there are multiple occurrences of variables:

$\forall x \in \mathbf{x} = [-1; 1],\ \sin x - x \in [-0.2; 0.2]$,
yet $\sin \mathbf{x} - \mathbf{x} \subseteq [-1.9; 1.9]$.

Definition (Polynomial enclosure)

$(P, \mathbf{\Delta}) \in \mathbb{R}[X] \times \mathbb{I}$ encloses f on $\mathbf{x} \ni x_0$ if

$$\forall x \in \mathbf{x},\ f(x) - P(x - x_0) \in \mathbf{\Delta}.$$

$(X^3/6, [-0.01; 0.01])$ encloses $\sin x - x$ on $[-1; 1] \ni 0$,
so $\sin x - x \in (\mathbf{x} - 0)^3/6 + [-0.01; 0.01] \subseteq [-0.2; 0.2]$.

Fighting the Dependency Effect

Definition (Polynomial enclosure)

$(P, \mathbf{\Delta}) \in \mathbb{R}[X] \times \mathbb{I}$ encloses f on $\mathbf{x} \ni x_0$ if

$$\forall x \in \mathbf{x},\ f(x) - P(x - x_0) \in \mathbf{\Delta}.$$

Enclosure of arithmetic operations

If $f \in (P_f, \mathbf{\Delta}_f)$ and $g \in (P_g, \mathbf{\Delta}_g)$ on $\mathbf{x} \ni x_0$,
then $f + g \in (P_f + P_g, \mathbf{\Delta}_f + \mathbf{\Delta}_g)$.

Fighting the Dependency Effect

Definition (Polynomial enclosure)

$(P, \mathbf{\Delta}) \in \mathbb{R}[X] \times \mathbb{I}$ encloses f on $\mathbf{x} \ni x_0$ if

$$\forall x \in \mathbf{x},\ f(x) - P(x - x_0) \in \mathbf{\Delta}.$$

Enclosure of arithmetic operations

If $f \in (P_f, \mathbf{\Delta}_f)$ and $g \in (P_g, \mathbf{\Delta}_g)$ on $\mathbf{x} \ni x_0$,
then $f + g \in (P_f + P_g, \mathbf{\Delta}_f + \mathbf{\Delta}_g)$.

Enclosure of elementary functions

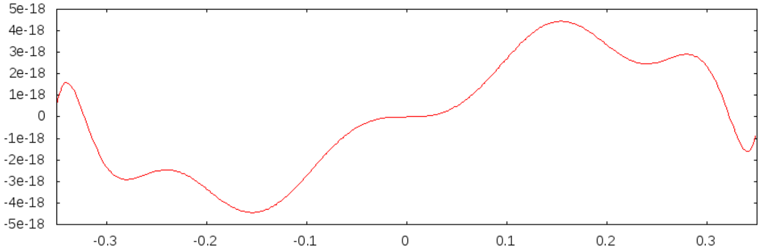
$$f(x) - \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k \in \frac{f^{(n+1)}(\mathbf{x})}{(n+1)!}(\mathbf{x} - x_0)^{n+1}.$$

Derivatives are obtained using the linear differential equation of f.

Bounding Approximation Errors

Example (Method error for Cody & Waite's algorithm)

```
Lemma method_error : forall t : R,
  let t2 := t * t in
  let p := p0 + t2 * (p1 + t2 * p2) in
  let q := q0 + t2 * (q1 + t2 * q2) in
  let f := 2 * (t * (p / (q - t * p)) + 1/2) in
  Rabs t <= 355 / 1024 ->
  Rabs ((f - exp t) / exp t) <= 23 * pow2 (-62).
```



Bounding Approximation Errors

Example (Method error for Cody & Waite's algorithm)

```
Lemma method_error : forall t : R,
  let t2 := t * t in
  let p := p0 + t2 * (p1 + t2 * p2) in
  let q := q0 + t2 * (q1 + t2 * q2) in
  let f := 2 * (t * (p / (q - t * p)) + 1/2) in
  Rabs t <= 355 / 1024 ->
  Rabs ((f - exp t) / exp t) <= 23 * pow2 (-62).
Proof.
  intros t t2 p q f Ht.
  unfold f, q, p, t2, p0, p1, p2, q0, q1, q2 ; simpl ;
  interval with (i_bisect_taylor t 9, i_prec 70).
Qed.
```

Proof completes in about 5 seconds
using degree-9 polynomials and 70-bit FP arithmetic.

Outline

1. Introduction
2. Formalizing the arithmetic
3. Fighting the dependency effect
4. Numerical integration
   - Integrating polynomial enclosures
   - Example: Helfgott's integral on MathOverflow
   - Example: improper integrals
5. Conclusion

Polynomial Integral Enclosure

Lemma (Polynomial integral enclosure)

Suppose f *is approximated on* $[u, v]$ *by* $p \in \mathbb{R}[X]$ *and* $\mathbf{\Delta} \in \mathbb{I}$ *in the sense that* $\forall x \in [u, v],\ f(x) - p(x) \in \mathbf{\Delta}$*.*
Then for any primitive P *of* p

$$\int_u^v f(t)\,dt \in P(\mathbf{v}) - P(\mathbf{u}) + (\mathbf{v} - \mathbf{u}) \cdot \mathbf{\Delta}.$$

Polynomial Integral Enclosure

Lemma (Polynomial integral enclosure)

Suppose f is approximated on $[u, v]$ by $p \in \mathbb{R}[X]$ and $\mathbf{\Delta} \in \mathbb{I}$ in the sense that $\forall x \in [u, v],\ f(x) - p(x) \in \mathbf{\Delta}$.
Then for any primitive P of p

$$\int_u^v f(t)\,dt \in P(\mathbf{v}) - P(\mathbf{u}) + (\mathbf{v} - \mathbf{u}) \cdot \mathbf{\Delta}.$$

Adaptive splitting

Integration domain is recursively split into two sub-domains until the target accuracy is reached.

Integrating a Non-Smooth Integrand

Example (Helfgott's integral on MathOverflow)

$$\int_0^1 \left|\left(x^4 + 10x^3 + 19x^2 - 6x - 6\right) \exp x\right| dx$$

Integrating a Non-Smooth Integrand

Example (Helfgott's integral on MathOverflow)

$$\int_0^1 \left|\left(x^4 + 10x^3 + 19x^2 - 6x - 6\right) \exp x\right| dx$$

Results when asked for 15 correct digits

- Matlab `quadv`, `quadcc`, `quadl`: correct answer. ✓
- Matlab `quad`, `quadgk`: only 10 correct digits, no warning. ✗
- Intlab `verifyquad`: absolute values not supported. ✧
- VNODE-LP: absolute value not supported. ✧

Integrating a Non-Smooth Integrand

Example (Helfgott's integral on MathOverflow)

$$\int_0^1 \left|\left(x^4 + 10x^3 + 19x^2 - 6x - 6\right)\, \exp x\right|\, dx$$

Results using CoqInterval

Target	Time	Degree	Depth	Prec
10^{-3}	0.7	5	8	30
10^{-6}	0.9	6	13	40
10^{-9}	1.3	8	18	50
10^{-12}	1.9	10	22	60
10^{-15}	2.7	12	28	70

Improper Integrals of the First Kind

Example (Major arcs for Goldbach's problem)

The paper states that

$$\int_{-\infty}^{\infty} \frac{(0.5 \cdot \log(\tau^2 + 2.25) + 4.1396 + \log \pi)^2}{0.25 + \tau^2} d\tau \leq 226.844.$$

Improper Integrals of the First Kind

Example (Major arcs for Goldbach's problem)

The paper states that

$$\int_{-\infty}^{\infty} \frac{(0.5 \cdot \log(\tau^2 + 2.25) + 4.1396 + \log \pi)^2}{0.25 + \tau^2} d\tau \leq 226.844.$$

CoqInterval proves $\ldots \in [226.849; 226.850]$.

Proof completes in about 30 seconds
using degree-10 polynomials and 40-bit FP arithmetic.

Note: Infinite endpoints are handled by manually factoring the integrand into Bertrand's form.

Outline

Conclusion

Contributions and limitations

- formally guaranteed bounds on real-valued expressions,
- support for (improper) integrals,
- simple algorithms yet efficient enough in practice,
- poor support for multivariate expressions.

Conclusion

Contributions and limitations

- formally guaranteed bounds on real-valued expressions,
- support for (improper) integrals,
- simple algorithms yet efficient enough in practice,
- poor support for multivariate expressions.

`http://coq-interval.gforge.inria.fr/`

Conclusion

Contributions and limitations

- formally guaranteed bounds on real-valued expressions,
- support for (improper) integrals,
- simple algorithms yet efficient enough in practice,
- poor support for multivariate expressions.

`http://coq-interval.gforge.inria.fr/`

Need more?

- Can't stand Coq? Extract and compile as an external tool.
- Need more speed? Realize integer operations with GMP.

What about Exact Reals in Coq?

- N. Julien (2008). Certified exact real arithmetic using coinduction in arbitrary integer base.
- I. Paşca (2008). A formal verification for Kantorovitch's theorem.
- R. O'Connor (2008). Certified exact transcendental real number computation in Coq.
- R. Krebbers, B. Spitters (2011). Computer certified efficient exact reals in Coq.