CoqInterval: A Toolbox for Proving Non-linear Univariate Inequalities in Coq

Érik Martin-Dorel
erik.martin-dorel@irit.fr

Équipe ACADIE, Laboratoire IRIT
Université Toulouse III - Paul Sabatier

Joint work with Guillaume Melquiond, Inria

with special thanks to the members of the TaMaDi–CoqApprox project

12 January 2016
MAP 2016 – Effective Analysis: Foundations, Implementations, Certification
CIRM, Luminy

Agenda

1. **Motivation:** Formal proof of approximation errors
2. The **Coq** proof assistant: computation and proof reflection
3. **CoqInterval:** Methodology, Architecture, and Examples
4. **Related works:** Comparison with existing tools
5. Conclusion and perspectives

Accuracy of floating-point elementary functions

- elementary functions ($\exp$, $\cos$, etc.) are ubiquitous in today's software
- the IEEE 754–2008 std for floating-point arithmetic gives recommendation on their accuracy
- it is crucial that `libm`s (libraries of mathematical functions) document the accuracy of the computed values!

Example of correctness claim

- Proving the implementation of $\exp$ in CRlibm[1] relies on the claim:

$$\forall x \in \mathbb{R},\ |x| \leq 355 \cdot 2^{-22} \implies \left| \frac{x + 0.5 \cdot x^2 + c_3 x^3 + c_4 x^4 - \exp x + 1}{\exp x - 1} \right| \leq 2^{-62} \quad (1)$$

with $c_3 = 6004799504235417 \cdot 2^{-55}$ and $c_4 = 1501199876148417 \cdot 2^{-55}$.

[1] http://lipforge.ens-lyon.fr/www/crlibm/

Example of correctness claim

- Proving the implementation of $\exp$ in CRlibm[1] relies on the claim:

$$\forall x \in \mathbb{R},\ |x| \leq 355 \cdot 2^{-22} \implies \left| \frac{x + 0.5 \cdot x^2 + c_3 x^3 + c_4 x^4 - \exp x + 1}{\exp x - 1} \right| \leq 2^{-62} \quad (1)$$

with $c_3 = 6004799504235417 \cdot 2^{-55}$ and $c_4 = 1501199876148417 \cdot 2^{-55}$.

- Tedious and error-prone to prove by hand!

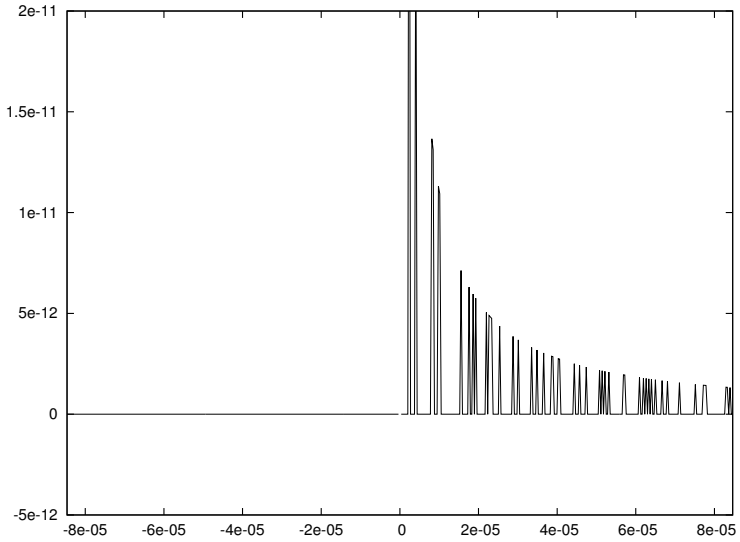
[1] http://lipforge.ens-lyon.fr/www/crlibm/

Example of correctness claim (continued)

- Attempt to verify (1) by plotting $f : x \mapsto \frac{x+0.5\cdot x^2+c_3x^3+c_4x^4-\exp x+1}{\exp x-1}$:

Example of correctness claim (continued)

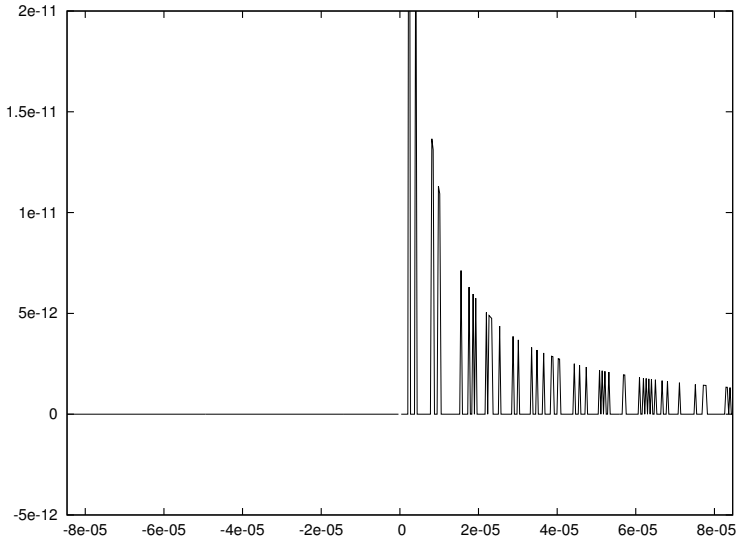
- Attempt to verify (1) by plotting $f : x \mapsto \frac{x+0.5\cdot x^2+c_3x^3+c_4x^4-\exp x+1}{\exp x-1}$:

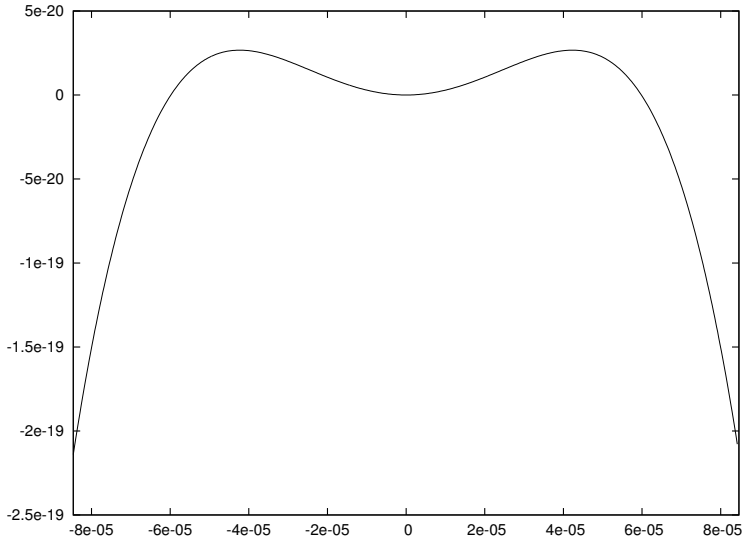


- On the left, the graph of f, as plotted by the Gnuplot tool.

Example of correctness claim (continued)

- Attempt to verify (1) by plotting $f : x \mapsto \frac{x+0.5\cdot x^2+c_3x^3+c_4x^4-\exp x+1}{\exp x-1}$:

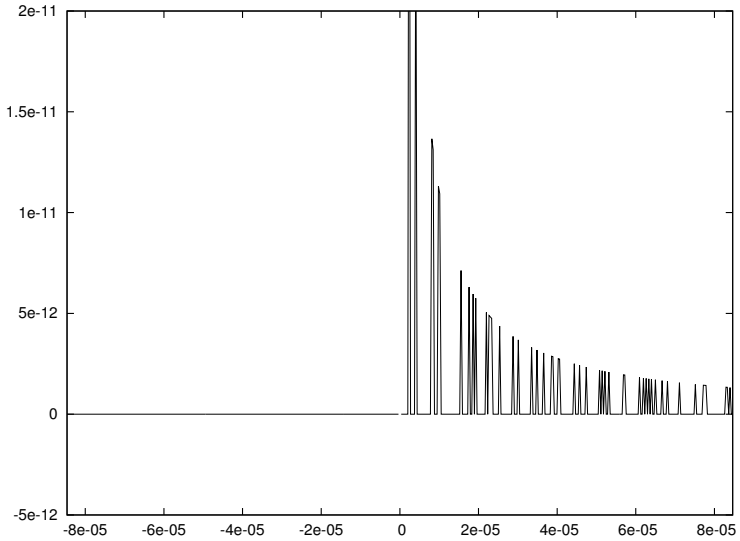


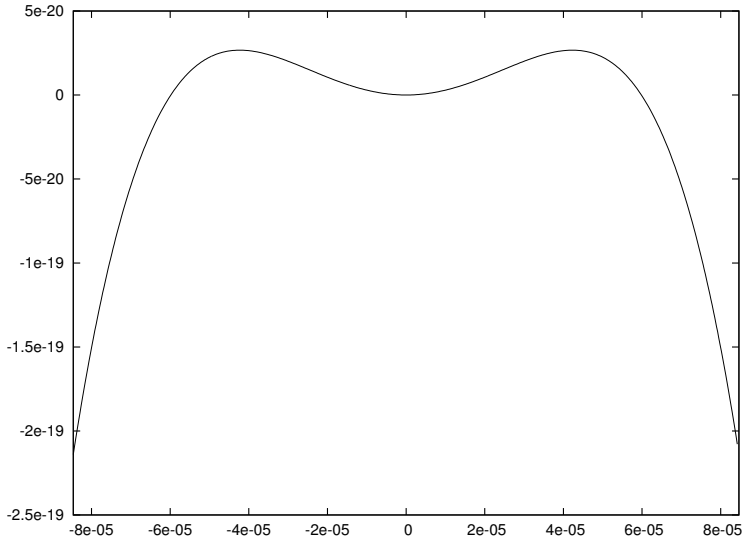


- On the left, the graph of f, as plotted by the Gnuplot tool.
- On the right, its actual graph, as plotted by Sollya.

Example of correctness claim (continued)

- Attempt to verify (1) by plotting $f : x \mapsto \frac{x+0.5\cdot x^2+c_3x^3+c_4x^4-\exp x+1}{\exp x-1}$:





- On the left, the graph of f, as plotted by the Gnuplot tool.
- On the right, its actual graph, as plotted by Sollya.

$\rightsquigarrow$ Need to use dedicated tools, e.g. proof assistants, to verify statements like (1) that are critical for the correctness of `libm`s' implementations

The Coq formal proof assistant

We use Coq for

- programming
  - pure functional language
  - specify algorithms and theorems
  - perform computations

The Coq formal proof assistant

We use Coq for

- programming
  - pure functional language
  - specify algorithms and theorems
  - perform computations
- proving
  - build proofs interactively
  - develop automatic tactics
  - use reflection
  - check proofs

The Coq formal proof assistant

We use Coq for

- programming
  - pure functional language
  - specify algorithms and theorems
  - perform computations
- proving
  - build proofs interactively
  - develop automatic tactics $\rightsquigarrow$ `Ltac`
  - use reflection
  - check proofs

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

So we can perform proofs by reflection:

- Suppose that we want to prove G.

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

So we can perform proofs by reflection:

- Suppose that we want to prove G.
- We reify G and automatically prove that $f(x_1, \ldots) = \text{true} \Rightarrow G$,

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

So we can perform proofs by reflection:

- Suppose that we want to prove G.
- We reify G and automatically prove that $f(x_1, \ldots) = \text{true} \Rightarrow G$,
  - by using a dedicated correctness lemma,

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

So we can perform proofs by reflection:

- Suppose that we want to prove G.
- We reify G and automatically prove that $f(x_1, \ldots) = \text{true} \Rightarrow G$,
  - by using a dedicated correctness lemma,
  - where f is a computable Boolean function f.

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

So we can perform proofs by reflection:

- Suppose that we want to prove G.
- We reify G and automatically prove that $f(x_1, \ldots) = \text{true} \Rightarrow G$,
  - by using a dedicated correctness lemma,
  - where f is a computable Boolean function f.
- Then we evaluate $f(x_1, \ldots)$.

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

So we can perform proofs by reflection:

- Suppose that we want to prove G.
- We reify G and automatically prove that $f(x_1, \ldots) = \text{true} \Rightarrow G$,
  - by using a dedicated correctness lemma,
  - where f is a computable Boolean function f.
- Then we evaluate $f(x_1, \ldots)$.
- If the computation yields true:

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

So we can perform proofs by reflection:

- Suppose that we want to prove G.
- We reify G and automatically prove that $f(x_1, \ldots) = \text{true} \Rightarrow G$,
  - by using a dedicated correctness lemma,
  - where f is a computable Boolean function f.
- Then we evaluate $f(x_1, \ldots)$.
- If the computation yields true:
  - we have proved that $\text{true} = \text{true} \Rightarrow G$,

Coq, computation, and proof by reflection

Coq comes with a primitive notion of computation, called conversion.

Key feature of Coq's logic: the convertibility rule

In environment E, if $p : A$ and if A and B are convertible, then $p : B$.

So we can perform proofs by reflection:

- Suppose that we want to prove G.
- We reify G and automatically prove that $f(x_1, \ldots) = \text{true} \Rightarrow G$,
  - by using a dedicated correctness lemma,
  - where f is a computable Boolean function f.
- Then we evaluate $f(x_1, \ldots)$.
- If the computation yields true:
  - we have proved that $\text{true} = \text{true} \Rightarrow G$,
  - which means that G holds.

Overview of the CoqInterval library — Issues and methods

- aim: (automatically) prove in Coq that the distance between $f(x)$ and some approximation $P(x)$ is bounded by some $\epsilon > 0$ for all $x \in \boldsymbol{I}$.

Overview of the CoqInterval library — Issues and methods

- aim: (automatically) prove in Coq that the distance between $f(x)$ and some approximation $P(x)$ is bounded by some $\epsilon > 0$ for all $x \in \boldsymbol{I}$.
- [G. Melquiond (2008): Proving bounds on real-valued functions with computations]

Overview of the CoqInterval library — Issues and methods

- aim: (automatically) prove in Coq that the distance between $f(x)$ and some approximation $P(x)$ is bounded by some $\epsilon > 0$ for all $x \in \boldsymbol{I}$.
- [G. Melquiond (2008): Proving bounds on real-valued functions with computations]
- main data-type: intervals with floating-point numbers bounds
  e.g., we'll consider an interval such as $[3.1415, 3.1416]$ in place of π

Overview of the CoqInterval library — Issues and methods

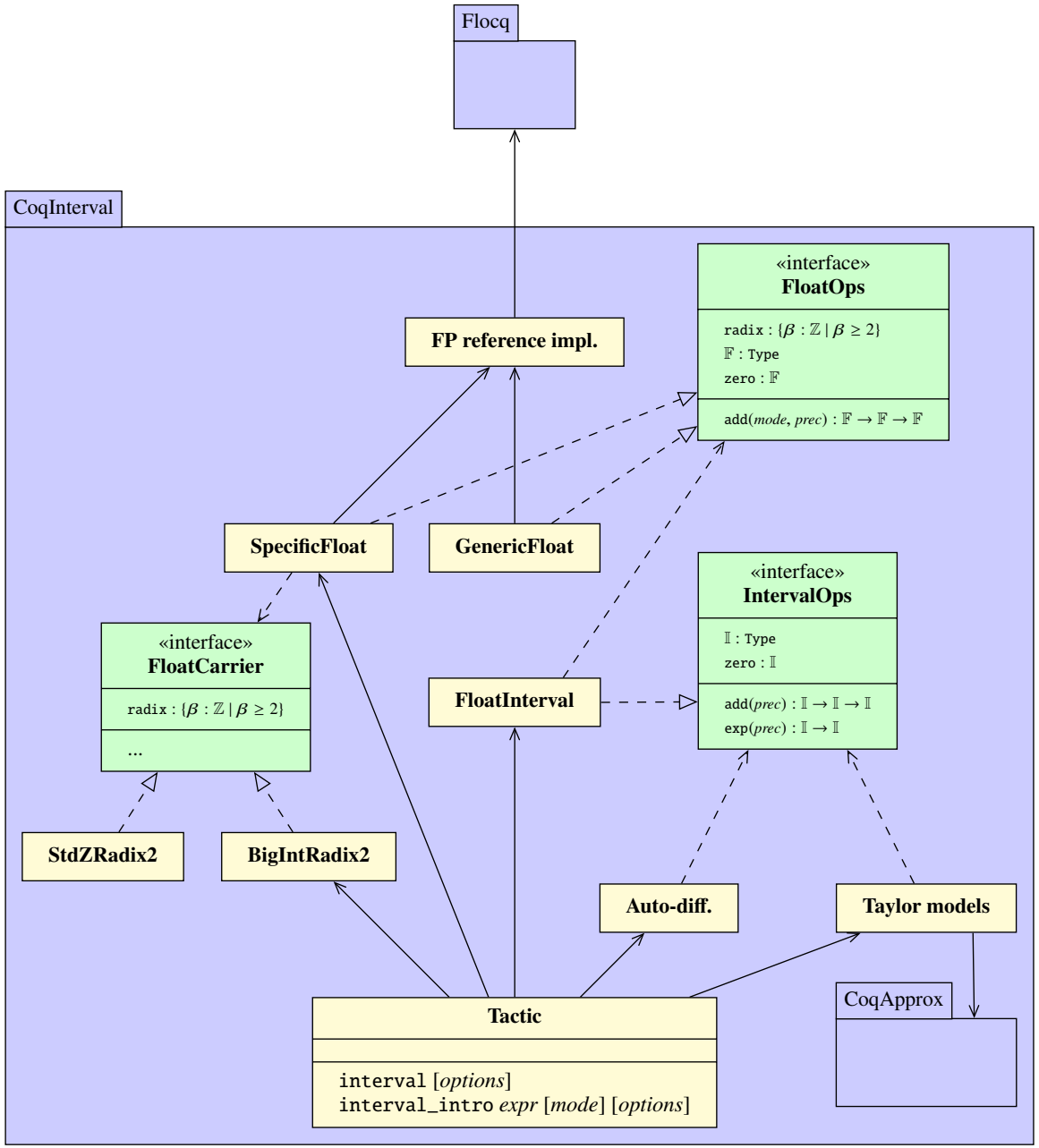
- aim: (automatically) prove in Coq that the distance between $f(x)$ and some approximation $P(x)$ is bounded by some $\epsilon > 0$ for all $x \in \boldsymbol{I}$.
- [G. Melquiond (2008): Proving bounds on real-valued functions with computations]
- main data-type: intervals with floating-point numbers bounds
  e.g., we'll consider an interval such as $[3.1415, 3.1416]$ in place of π
- dependency problem: when a variable occur several times, it typically leads to an overestimation of the range
  e.g., for $f(x) = x \cdot (1 - x)$ and $\boldsymbol{x} = [0, 1]$, we get $\mathrm{eval}_{\mathrm{IA}}(f, \boldsymbol{x}) = [0, 1]$, while the exact range is $f(\boldsymbol{x}) = [0, \frac{1}{4}]$

Overview of the CoqInterval library — Issues and methods

- aim: (automatically) prove in Coq that the distance between $f(x)$ and some approximation $P(x)$ is bounded by some $\epsilon > 0$ for all $x \in \boldsymbol{I}$.
- [G. Melquiond (2008): Proving bounds on real-valued functions with computations]
- main data-type: intervals with floating-point numbers bounds
  e.g., we'll consider an interval such as $[3.1415, 3.1416]$ in place of π
- dependency problem: when a variable occur several times, it typically leads to an overestimation of the range
  e.g., for $f(x) = x \cdot (1 - x)$ and $\boldsymbol{x} = [0, 1]$, we get $\mathrm{eval}_{\mathrm{IA}}(f, \boldsymbol{x}) = [0, 1]$, while the exact range is $f(\boldsymbol{x}) = [0, \frac{1}{4}]$
- solutions: bisection, automatic differentiation...

Overview of the CoqInterval library — Issues and methods

- aim: (automatically) prove in Coq that the distance between $f(x)$ and some approximation $P(x)$ is bounded by some $\epsilon > 0$ for all $x \in \boldsymbol{I}$.
- [G. Melquiond (2008): Proving bounds on real-valued functions with computations]
- main data-type: intervals with floating-point numbers bounds
  e.g., we'll consider an interval such as $[3.1415, 3.1416]$ in place of π
- dependency problem: when a variable occur several times, it typically leads to an overestimation of the range
  e.g., for $f(x) = x \cdot (1 - x)$ and $\boldsymbol{x} = [0, 1]$, we get $\mathrm{eval}_{\mathrm{IA}}(f, \boldsymbol{x}) = [0, 1]$, while the exact range is $f(\boldsymbol{x}) = [0, \frac{1}{4}]$
- solutions: bisection, automatic differentiation. . . or Taylor Models:
  [N. Brisebarre, M. Joldeș, EMD, M. Mayero, J-M. Muller, I. Pașca, L. Rideau, and L. Théry (2012): Rigorous Polynomial Approximation Using Taylor Models in Coq]

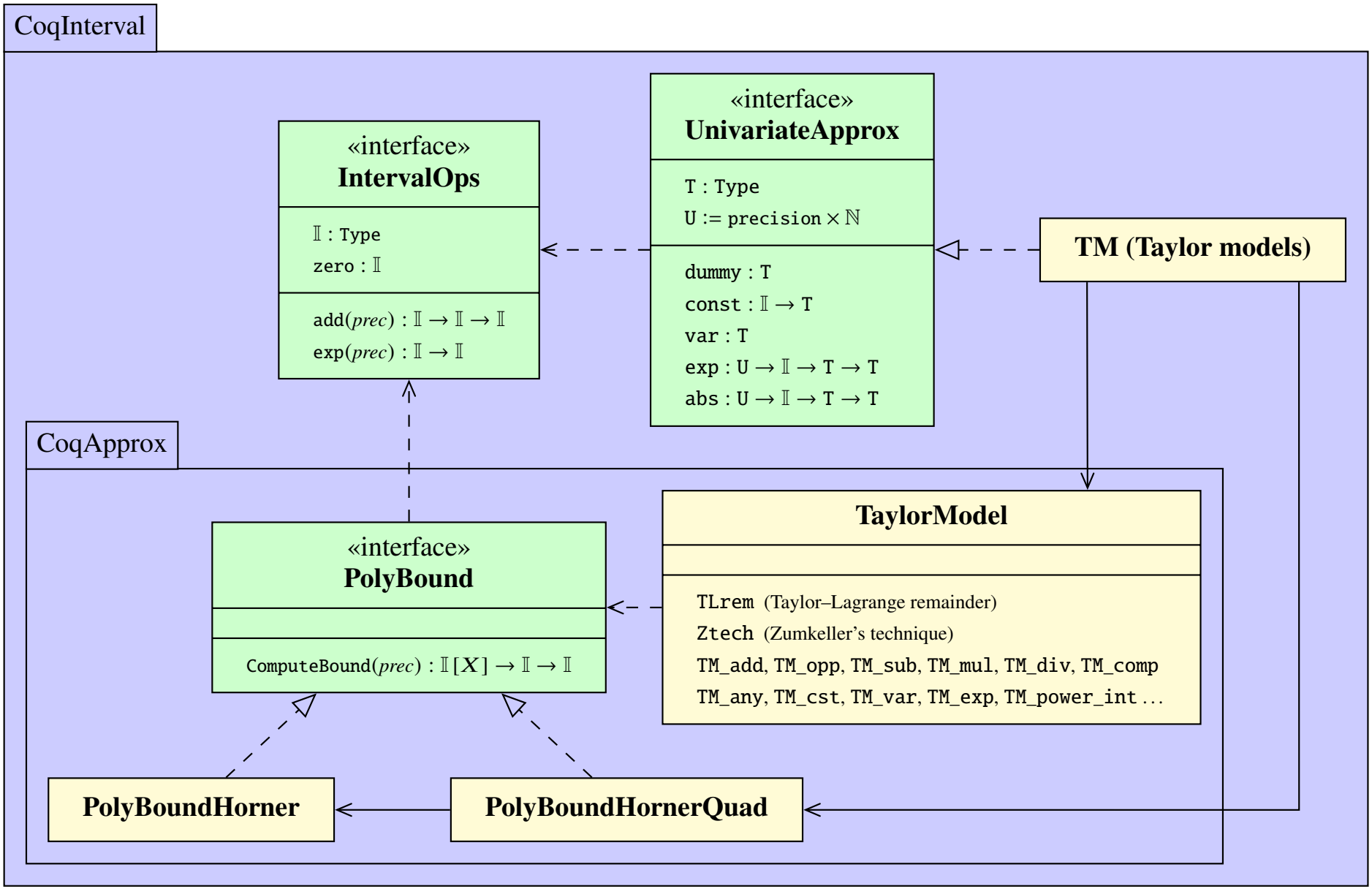


Caption:

A - - - - - - -> I
if the module A is parameterized by a module implementing I

C - - - - - - ▷ I
if the module C implements the interface I

M ——————> C
if the module M uses the module C


CoqInterval
«interface»
IntervalOps
𝕀 : Type
zero : 𝕀
add(prec) : 𝕀 → 𝕀 → 𝕀
exp(prec) : 𝕀 → 𝕀
«interface»
UnivariateApprox
T : Type
U := precision × ℕ
dummy : T
const : 𝕀 → T
var : T
exp : U → 𝕀 → T → T
abs : U → 𝕀 → T → T
TM (Taylor models)
CoqApprox
«interface»
PolyBound
ComputeBound(prec) : 𝕀[X] → 𝕀 → 𝕀
TaylorModel
TLrem (Taylor–Lagrange remainder)
Ztech (Zumkeller's technique)
TM_add, TM_opp, TM_sub, TM_mul, TM_div, TM_comp
TM_any, TM_cst, TM_var, TM_exp, TM_power_int ...
PolyBoundHorner
PolyBoundHornerQuad

The interval and interval_intro tactics

Syntax:

- `interval` *options*. (* decision procedure *)
- `interval_intro (`*expr*`)` *options* `as [H1 H2].` (* forward chaining *)
- `interval_intro (`*expr*`) lower` *options* `as H1.`
- `interval_intro (`*expr*`) upper` *options* `as H2.`

The `interval` and `interval_intro` tactics

Syntax:

- `interval` *`options`*`.` (* decision procedure *)
- `interval_intro (`*`expr`*`)` *`options`* `as [H1 H2].` (* forward chaining *)
- `interval_intro (`*`expr`*`) lower` *`options`* `as H1.`
- `interval_intro (`*`expr`*`) upper` *`options`* `as H2.`

`options` ::= [`with (`*`option`*$_1$`,`*`option`*$_2$`,...)`] chosen among the following:

- `i_prec` p: precision of radix-2 FP computations (30 bits by default)

The `interval` and `interval_intro` tactics

Syntax:

- `interval` *`options`*`.` (* decision procedure *)
- `interval_intro (`*`expr`*`)` *`options`* `as [H1 H2].` (* forward chaining *)
- `interval_intro (`*`expr`*`) lower` *`options`* `as H1.`
- `interval_intro (`*`expr`*`) upper` *`options`* `as H2.`

`options` ::= [`with (`*`option`*$_1$`,`*`option`*$_2$`,...)`] chosen among the following:

- `i_prec` p: precision of radix-2 FP computations (30 bits by default)
- `i_depth` n: maximum depth of bisection

The interval and interval_intro tactics

Syntax:

- `interval` *options*. (* decision procedure *)
- `interval_intro` (*expr*) *options* `as [H1 H2].` (* forward chaining *)
- `interval_intro` (*expr*) `lower` *options* `as H1.`
- `interval_intro` (*expr*) `upper` *options* `as H2.`

options ::= [`with` (*option*$_1$,*option*$_2$,...)] chosen among the following:

- `i_prec` p: precision of radix-2 FP computations (30 bits by default)
- `i_depth` n: maximum depth of bisection
- `i_bisect` x: do a bisection along variable x

The interval and interval_intro tactics

Syntax:

- `interval` *options*. (* decision procedure *)
- `interval_intro (`*expr*`)` *options* `as [H1 H2].` (* forward chaining *)
- `interval_intro (`*expr*`) lower` *options* `as H1.`
- `interval_intro (`*expr*`) upper` *options* `as H2.`

options ::= [`with (`$option_1$`,`$option_2$`,...)`] chosen among the following:

- `i_prec` p: precision of radix-2 FP computations (30 bits by default)
- `i_depth` n: maximum depth of bisection
- `i_bisect` x: do a bisection along variable x
- `i_bisect_diff` x: do a bisection and automatic differentiation w.r.t. x

The `interval` and `interval_intro` tactics

Syntax:

- `interval` *`options`*. (* decision procedure *)
- `interval_intro (`*`expr`*`)` *`options`* `as [H1 H2].` (* forward chaining *)
- `interval_intro (`*`expr`*`) lower` *`options`* `as H1.`
- `interval_intro (`*`expr`*`) upper` *`options`* `as H2.`

`options` ::= [`with (`*`option`*$_1$`,`*`option`*$_2$`,...)`] chosen among the following:

- `i_prec` p: precision of radix-2 FP computations (30 bits by default)
- `i_depth` n: maximum depth of bisection
- `i_bisect` x: do a bisection along variable x
- `i_bisect_diff` x: do a bisection and automatic differentiation w.r.t. x
- `i_bisect_taylor` x d: do a bisection along variable x while computing degree-d univariate Taylor models

Overview of the CoqInterval library — Proof example #1

Example taken from [John Harrison (1997): Verifying the Accuracy of Polynomial Approximations in HOL]

```
Require Import Reals Interval_tactic.
Local Open Scope R_scope.
```

```
Theorem Harrison97 : ∀x : ℝ, -10831/1000000 ≤ x ≤ 10831/1000000 ⟹
  |(e^x - 1) - (x + 8388676/2^24 x^2 + 11184876/2^26 x^3)| ≤ 23/27 × 1/2^33 .
```

Overview of the CoqInterval library — Proof example #1

Example taken from [John Harrison (1997): Verifying the Accuracy of Polynomial Approximations in HOL]

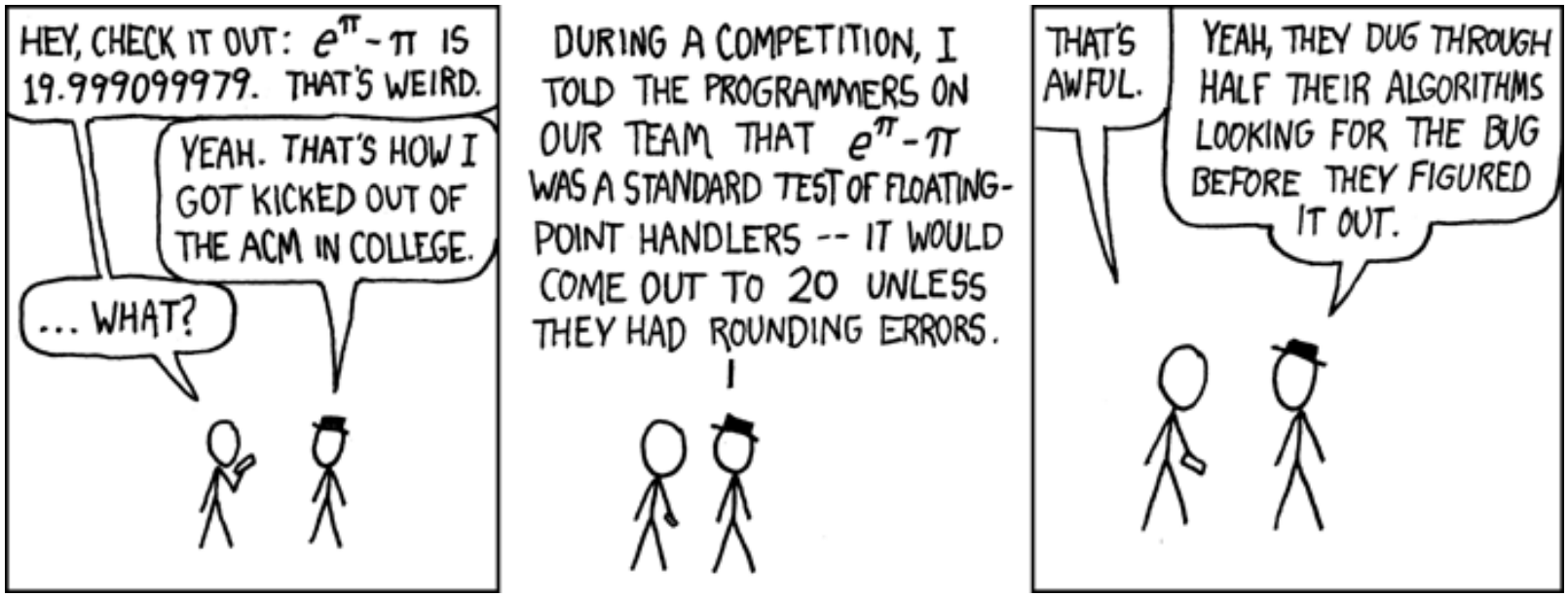
```
Require Import Reals Interval_tactic.
Local Open Scope R_scope.

Theorem Harrison97 : ∀x : ℝ, -10831/1000000 ≤ x ≤ 10831/1000000 ⟹
    |(e^x - 1) - (x + 8388676/2^24 x^2 + 11184876/2^26 x^3)| ≤ 23/27 × 1/2^33 .
Proof.
intros x H.
interval with (i_bisect_diff x, i_prec 50, i_depth 16). (* 35s *)
Qed.
```

Overview of the CoqInterval library — Proof example #1

Example taken from [John Harrison (1997): Verifying the Accuracy of Polynomial Approximations in HOL]

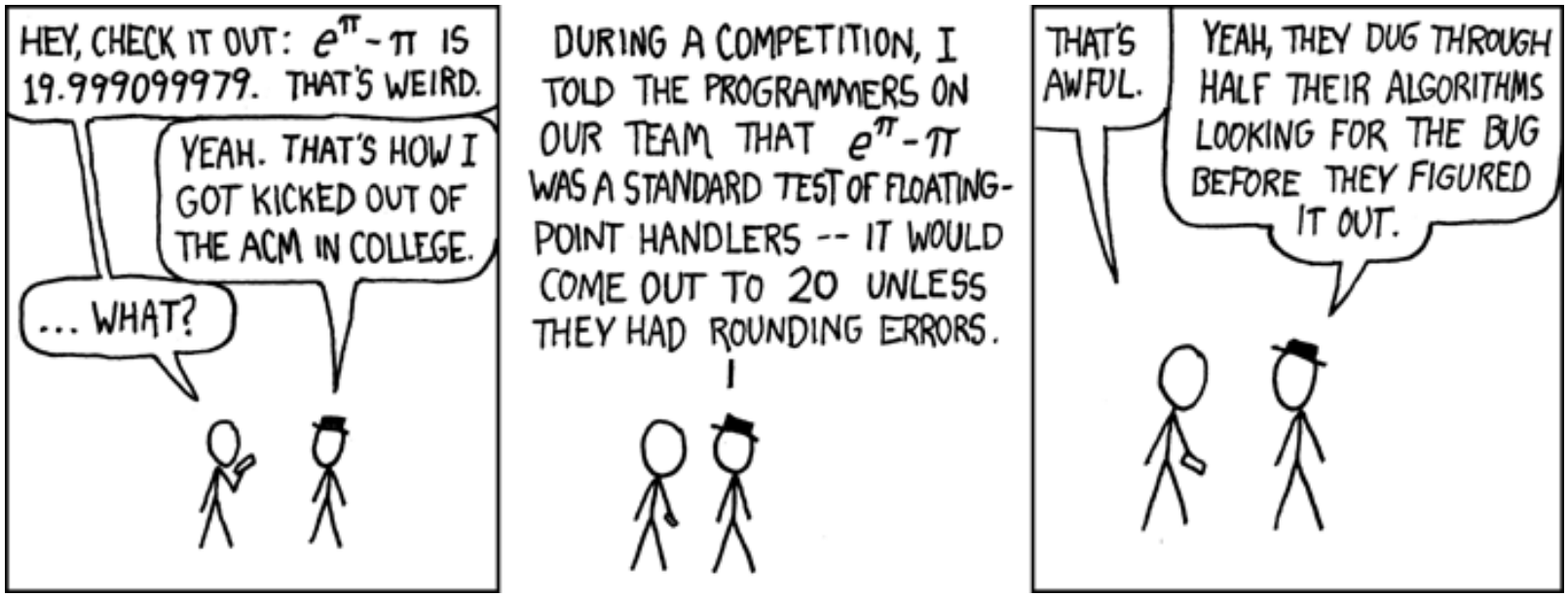
```
Require Import Reals Interval_tactic.
Local Open Scope R_scope.

Theorem Harrison97 : ∀x : ℝ, -10831/1000000 ≤ x ≤ 10831/1000000 ⟹
    |(e^x - 1) - (x + 8388676/2^24 x^2 + 11184876/2^26 x^3)| ≤ 23/27 × 1/2^33 .
Proof.
intros x H.
interval with (i_bisect_taylor x 3, i_prec 50). (* 0.50s *)
Qed.
```

Overview of the CoqInterval library — Proof example #2



(xkcd.com/217)

Overview of the CoqInterval library — Proof example #2



(xkcd.com/217)

```
Require Import Reals Interval_tactic.
Local Open Scope R_scope.

Lemma xkcd217 : exp PI - PI <> 20.
Proof.
interval. (* 0.05s *)
Qed.
```

Bisection

- Idea: Split $\boldsymbol{x}$ into sub-intervals $\boldsymbol{x} = \boldsymbol{a} \cup \boldsymbol{b}$, so we get $f(\boldsymbol{x}) \subset \boldsymbol{f}(\boldsymbol{a}) \cup \boldsymbol{f}(\boldsymbol{b})$ (which is a tighter inclusion than $f(\boldsymbol{x}) \subset \boldsymbol{f}(\boldsymbol{x})$)
- Then: Iterate the process recursively on $\boldsymbol{a}$ and $\boldsymbol{b}$.

- Drawback: Proving something like $\forall x \in [0,1],\ |x - x| \leq 2^{-40}$ with this technique alone yields a huge number of sub-intervals
- And it will not succeed in proving $\forall x \in [0,1],\ x - x = 0$.

- Advantage: Can be combined with other approaches to reduce the dependency effect (cf. `i_bisect_diff` and `i_bisect_taylor`)

Automatic differentiation

- Based on the interval version of Taylor-Lagrange's formula at order 0,

$$\forall x \in \boldsymbol{x},\ \exists \xi \in \boldsymbol{x}, \quad f(x) = f(x_0) + (x - x_0) \cdot f'(\xi),$$
$$\forall x \in \boldsymbol{x}, \quad f(x) \in \boldsymbol{f}([x_0, x_0]) + (\boldsymbol{x} - [x_0, x_0]) \cdot \boldsymbol{f'}(\boldsymbol{x}).$$

- Rely on automatic differentiation to compute $\boldsymbol{f'}(\boldsymbol{x})$
  - Work with pairs of intervals $\underbrace{(\boldsymbol{u}}_{\substack{\text{enclosure} \\ \text{of } f(x)}}, \underbrace{\boldsymbol{u'})}_{\substack{\text{enclosure} \\ \text{of } f'(x)}}$
  - Example of rule: $(\boldsymbol{u}, \boldsymbol{u'}) \times (\boldsymbol{v}, \boldsymbol{v'}) = (\boldsymbol{uv}, \boldsymbol{u'v} + \boldsymbol{uv'})$
- For the toy example $f(x) = x - x$ over $\boldsymbol{x} = [0, 1]$ (cf. previous slide), we get $\boldsymbol{f'}(\boldsymbol{x}) = [0, 0]$, so f is a constant function $f \equiv f(x_0) = 0$. QED.

CoqApprox: certified library of Taylor models

A Taylor model is a pair (polynom, error interval) and we will say that $(P, \boldsymbol{\Delta})$ represents a function f over $\boldsymbol{I}$ if we have $\forall x \in \boldsymbol{I},\ f(x) - P(x) \in \boldsymbol{\Delta}$

CoqApprox: certified library of Taylor models

A Taylor model is a pair (polynom, error interval) and we will say that $(P, \boldsymbol{\Delta})$ represents a function f over $\boldsymbol{I}$ if we have $\forall x \in \boldsymbol{I},\ f(x) - P(x) \in \boldsymbol{\Delta}$

Goal : find some $\boldsymbol{\Delta}$ as small as possible.

CoqApprox: certified library of Taylor models

A Taylor model is a pair (polynom, error interval) and we will say that $(P, \boldsymbol{\Delta})$ represents a function f over $\boldsymbol{I}$ if we have $\forall x \in \boldsymbol{I},\ f(x) - P(x) \in \boldsymbol{\Delta}$

Goal : find some $\boldsymbol{\Delta}$ as small as possible.

Methodology in 2 steps

1. For "basic functions", compute an enclosure of the Taylor–Lagrange remainder at order n;
2. For "composite functions", use a dedicated algorithm for addition, multiplication, composition, and division.

CoqApprox: certified library of Taylor models

A Taylor model is a pair (polynom, error interval) and we will say that $(P, \boldsymbol{\Delta})$ represents a function f over $\boldsymbol{I}$ if we have $\forall x \in \boldsymbol{I},\ f(x) - P(x) \in \boldsymbol{\Delta}$

Goal : find some $\boldsymbol{\Delta}$ as small as possible.

Methodology in 2 steps

1. For "basic functions", compute an enclosure of the Taylor–Lagrange remainder at order n;
2. For "composite functions", use a dedicated algorithm for addition, multiplication, composition, and division.

Within CoqApprox: certified computation of Taylor models for functions $\sqrt{\cdot}$, $\frac{1}{\sqrt{\cdot}}$, $x \mapsto x^n$ $(n \in \mathbb{Z})$, $\exp$, $\sin$, $\cos$, $\ln$ and operations $+$, $-$, $\times$, $\div$, $\circ$.
Support for $\tan$ and $\arctan$ TMs in the upcoming release of CoqInterval!

CoqApprox' technicalities – what's new for next release?

- Add support for $\tan$ and $\arctan$ (formalizing Sollya's algorithm)

CoqApprox' technicalities – what's new for next release?

- Add support for $\tan$ and $\arctan$ (formalizing Sollya's algorithm)
- Depend on the `Coquelicot` library

CoqApprox' technicalities – what's new for next release?

- Add support for $\tan$ and $\arctan$ (formalizing Sollya's algorithm)
- Depend on the `Coquelicot` library
- **From** polynomials over $\mathbb{R} \cup \{\text{NaN}\}$ **to** polynomials over $\mathbb{R}$ and separated proofs for NaN propagation ($+$ changes in `IntervalOps`)

CoqApprox' technicalities – what's new for next release?

- Add support for $\tan$ and $\arctan$ (formalizing Sollya's algorithm)
- Depend on the `Coquelicot` library
- **From** polynomials over $\mathbb{R} \cup \{\text{NaN}\}$ **to** polynomials over $\mathbb{R}$ and separated proofs for NaN propagation ($+$ changes in `IntervalOps`)
- No more need to give explicit formulas of n^{th} derivatives over $\mathbb{R} \cup \{\text{NaN}\}$ and verify them, instead we just need to instantiate:
  `Hypothesis Hder_n : ∀n r, der r → ex_derive_n f n r.`

CoqApprox' technicalities – what's new for next release?

- Add support for $\tan$ and $\arctan$ (formalizing Sollya's algorithm)
- Depend on the `Coquelicot` library
- **From** polynomials over $\mathbb{R} \cup \{\text{NaN}\}$ **to** polynomials over $\mathbb{R}$ and separated proofs for NaN propagation ($+$ changes in `IntervalOps`)
- No more need to give explicit formulas of n^{th} derivatives over $\mathbb{R} \cup \{\text{NaN}\}$ and verify them, instead we just need to instantiate:
  `Hypothesis Hder_n : ∀n r, der r → ex_derive_n f n r.`
- Remove the dependency on the excluded-middle axiom

CoqApprox' technicalities – what's new for next release?

- Add support for $\tan$ and $\arctan$ (formalizing Sollya's algorithm)
- Depend on the `Coquelicot` library
- **From** polynomials over $\mathbb{R} \cup \{\text{NaN}\}$ **to** polynomials over $\mathbb{R}$ and separated proofs for NaN propagation ($+$ changes in `IntervalOps`)
- No more need to give explicit formulas of n^{th} derivatives over $\mathbb{R} \cup \{\text{NaN}\}$ and verify them, instead we just need to instantiate: `Hypothesis Hder_n : ∀n r, der r → ex_derive_n f n r.`
- Remove the dependency on the excluded-middle axiom
- Remove degree constraints in `TM_add_correct`, `TM_mul_correct`, and so on $\rightsquigarrow$ no more side-conditions nor padding ($\rightsquigarrow$ better perf. expected for multiplying TMs with heterogeneous sizes)

CoqApprox' technicalities – what's new for next release?

- Add support for $\tan$ and $\arctan$ (formalizing Sollya's algorithm)
- Depend on the `Coquelicot` library
- **From** polynomials over $\mathbb{R} \cup \{\text{NaN}\}$ **to** polynomials over $\mathbb{R}$ and separated proofs for NaN propagation ($+$ changes in `IntervalOps`)
- No more need to give explicit formulas of n^{th} derivatives over $\mathbb{R} \cup \{\text{NaN}\}$ and verify them, instead we just need to instantiate: `Hypothesis Hder_n : ∀n r, der r → ex_derive_n f n r.`
- Remove the dependency on the excluded-middle axiom
- Remove degree constraints in `TM_add_correct`, `TM_mul_correct`, and so on $\rightsquigarrow$ no more side-conditions nor padding ($\rightsquigarrow$ better perf. expected for multiplying TMs with heterogeneous sizes)
- Architecture: Remove/merge several `Modules`

CoqApprox' technicalities – what's new for next release?

- Add support for $\tan$ and $\arctan$ (formalizing Sollya's algorithm)
- Depend on the `Coquelicot` library
- **From** polynomials over $\mathbb{R} \cup \{\text{NaN}\}$ **to** polynomials over $\mathbb{R}$ and separated proofs for NaN propagation ($+$ changes in `IntervalOps`)
- No more need to give explicit formulas of n^{th} derivatives over $\mathbb{R} \cup \{\text{NaN}\}$ and verify them, instead we just need to instantiate: `Hypothesis Hder_n : ∀n r, der r → ex_derive_n f n r.`
- Remove the dependency on the excluded-middle axiom
- Remove degree constraints in `TM_add_correct`, `TM_mul_correct`, and so on $\rightsquigarrow$ no more side-conditions nor padding ($\rightsquigarrow$ better perf. expected for multiplying TMs with heterogeneous sizes)
- Architecture: Remove/merge several `Modules`
- Misc improvements (new helper tactics, naming convention, ...)

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)
- MetiTarski: standalone tool = axioms for approximating elementary functions + decision procedure for multivariate polynomial inequalities

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)
- MetiTarski: standalone tool = axioms for approximating elementary functions + decision procedure for multivariate polynomial inequalities
- HOL Light/`REAL_SOS`: decision procedure for multivariate polynomial inequalities (SOS certificates)

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)
- MetiTarski: standalone tool = axioms for approximating elementary functions + decision procedure for multivariate polynomial inequalities
- HOL Light/`REAL_SOS`: decision procedure for multivariate polynomial inequalities (SOS certificates)
- HOL Light/`verify_ineq`: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (order-1 TL)

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)
- MetiTarski: standalone tool = axioms for approximating elementary functions + decision procedure for multivariate polynomial inequalities
- HOL Light/`REAL_SOS`: decision procedure for multivariate polynomial inequalities (SOS certificates)
- HOL Light/`verify_ineq`: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (order-1 TL)
- NLCertify: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (quadratic-forms approx + SDP)

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)
- MetiTarski: standalone tool = axioms for approximating elementary functions + decision procedure for multivariate polynomial inequalities
- HOL Light/`REAL_SOS`: decision procedure for multivariate polynomial inequalities (SOS certificates)
- HOL Light/`verify_ineq`: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (order-1 TL)
- NLCertify: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (quadratic-forms approx + SDP)
- PVS/Bernstein: decision procedure for multivariate polynomial inequalities (Bernstein polynomials + global optimization)

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)
- MetiTarski: standalone tool = axioms for approximating elementary functions + decision procedure for multivariate polynomial inequalities
- HOL Light/`REAL_SOS`: decision procedure for multivariate polynomial inequalities (SOS certificates)
- HOL Light/`verify_ineq`: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (order-1 TL)
- NLCertify: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (quadratic-forms approx + SDP)
- PVS/Bernstein: decision procedure for multivariate polynomial inequalities (Bernstein polynomials + global optimization)
- PVS/interval: decision procedure for multivariate ineqs with elementary functions (Interval Arithmetic + Branch & Bound)

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)
- MetiTarski: standalone tool = axioms for approximating elementary functions + decision procedure for multivariate polynomial inequalities
- HOL Light/`REAL_SOS`: decision procedure for multivariate polynomial inequalities (SOS certificates)
- HOL Light/`verify_ineq`: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (order-1 TL)
- NLCertify: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (quadratic-forms approx + SDP)
- PVS/Bernstein: decision procedure for multivariate polynomial inequalities (Bernstein polynomials + global optimization)
- PVS/interval: decision procedure for multivariate ineqs with elementary functions (Interval Arithmetic + Branch & Bound)

Panorama of existing tools

- Sollya: rigorous computing toolbox for the `libm` developer, relying on MPFI. Considered function: `supnorm` (otherwise `checkinfnorm`)
- MetiTarski: standalone tool = axioms for approximating elementary functions + decision procedure for multivariate polynomial inequalities
- HOL Light/`REAL_SOS`: decision procedure for multivariate polynomial inequalities (SOS certificates)
- HOL Light/`verify_ineq`: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (order-1 TL)
- NLCertify: born in Flyspeck, decision procedure for multivariate ineqs with elementary functions (quadratic-forms approx + SDP)
- PVS/Bernstein: decision procedure for multivariate polynomial inequalities (Bernstein polynomials + global optimization)
- PVS/interval: decision procedure for multivariate ineqs with elementary functions (Interval Arithmetic + Branch & Bound)

Design of a multi-prover test-suite

- approximation problems: CRlibm's $\exp$ ($|x| \geq 2^{-20}$), a Remez of $\sqrt{\cdot}$, a degree-5 approx of $\arctan$, Earth's radius of curvature, Tang's $\exp$

Design of a multi-prover test-suite

- approximation problems: CRlibm's $\exp$ ($|x| \geq 2^{-20}$), a Remez of $\sqrt{\cdot}$, a degree-5 approx of $\arctan$, Earth's radius of curvature, Tang's $\exp$
- degree-2 to degree-8 approximations problems of $x \mapsto \cos(1.5 \cdot \cos x)$ with binary32 coefficients

Design of a multi-prover test-suite

- approximation problems: CRlibm's $\exp$ ($|x| \geq 2^{-20}$), a Remez of $\sqrt{\cdot}$, a degree-5 approx of $\arctan$, Earth's radius of curvature, Tang's $\exp$
- degree-2 to degree-8 approximations problems of $x \mapsto \cos(1.5 \cdot \cos x)$ with binary32 coefficients
- 25 problems from MetiTarski's test-suite selected to be compatible with all provers' input

Design of a multi-prover test-suite

- approximation problems: CRlibm's $\exp$ ($|x| \geq 2^{-20}$), a Remez of $\sqrt{\cdot}$, a degree-5 approx of $\arctan$, Earth's radius of curvature, Tang's $\exp$
- degree-2 to degree-8 approximations problems of $x \mapsto \cos(1.5 \cdot \cos x)$ with binary32 coefficients
- 25 problems from MetiTarski's test-suite selected to be compatible with all provers' input
- 4 typical multivariate polynomial inequalities: RD, adaptiveLV, butcher, magnetism

Design of a multi-prover test-suite

- approximation problems: CRlibm's $\exp$ ($|x| \geq 2^{-20}$), a Remez of $\sqrt{\cdot}$, a degree-5 approx of $\arctan$, Earth's radius of curvature, Tang's $\exp$
- degree-2 to degree-8 approximations problems of $x \mapsto \cos(1.5 \cdot \cos x)$ with binary32 coefficients
- 25 problems from MetiTarski's test-suite selected to be compatible with all provers' input
- 4 typical multivariate polynomial inequalities: RD, adaptiveLV, butcher, magnetism
- System: Ubuntu 14.04.2 LTS on Intel Core i5-4460S CPU @ 2.90 GHz

Design of a multi-prover test-suite

- approximation problems: CRlibm's $\exp$ ($|x| \geq 2^{-20}$), a Remez of $\sqrt{\cdot}$, a degree-5 approx of $\arctan$, Earth's radius of curvature, Tang's $\exp$
- degree-2 to degree-8 approximations problems of $x \mapsto \cos(1.5 \cdot \cos x)$ with binary32 coefficients
- 25 problems from MetiTarski's test-suite selected to be compatible with all provers' input
- 4 typical multivariate polynomial inequalities: RD, adaptiveLV, butcher, magnetism
- System: Ubuntu 14.04.2 LTS on Intel Core i5-4460S CPU @ 2.90 GHz
- Output: total time in s | Failed ($\Leftrightarrow$ error) | Timeout ($\Leftrightarrow > 180\,s$) | -

Design of a multi-prover test-suite

- approximation problems: CRlibm's $\exp$ ($|x| \geq 2^{-20}$), a Remez of $\sqrt{\cdot}$, a degree-5 approx of $\arctan$, Earth's radius of curvature, Tang's $\exp$
- degree-2 to degree-8 approximations problems of $x \mapsto \cos(1.5 \cdot \cos x)$ with binary32 coefficients
- 25 problems from MetiTarski's test-suite selected to be compatible with all provers' input
- 4 typical multivariate polynomial inequalities: RD, adaptiveLV, butcher, magnetism
- System: Ubuntu 14.04.2 LTS on Intel Core i5-4460S CPU @ 2.90 GHz
- Output: total time in s | Failed ($\Leftrightarrow$ error) | Timeout ($\Leftrightarrow > 180\,s$) | -

Forge: https://gforge.inria.fr/scm/browser.php?group_id=6316&extra=bench-ineqs

Experimental Results (univariate approximation problems)

Problems	CoqInterval 2.0	Sollya	MetiTarski	NLCertify (not verified)	NLCertify (verified polys)	PVS/interval	HOL Light/ `verify_ineq`	PVS/Bernstein	HOL Light/ `REAL_SOS`
crlibm_exp	0.83*	0.02	Failed	-	-	Failed	-	-	-
remez_sqrt	0.45	0.02	0.05	15.28*	Timeout	Failed	3.60*	-	-
abs_err_atan	0.45	0.01	0.07	Failed	Failed	Timeout	2.36*	-	-
rel_err_geo	3.10	2.24	Timeout	Timeout	Timeout	Failed	229.54*	-	-
harrison97	0.42	0.01	0.10	-	-	Failed	-	-	-
cos_cos_d2	0.71	0.05	Timeout	Timeout	Timeout	20.64	5.82*	-	-
cos_cos_d3	0.79	0.05	Timeout	Timeout	Timeout	48.87	6.28*	-	-
cos_cos_d4	0.91	0.06	Timeout	Timeout	Timeout	Timeout	8.83*	-	-
cos_cos_d5	1.44	0.06	Timeout	Timeout	Timeout	Timeout	15.70*	-	-
cos_cos_d6	1.54	0.07	Timeout	Timeout	Timeout	Timeout	20.92*	-	-
cos_cos_d7	2.21	0.07	Timeout	Timeout	Timeout	Timeout	41.88*	-	-
cos_cos_d8	2.79	0.08	Timeout	Timeout	Timeout	Timeout	87.78*	-	-

Experimental Results (MetiTarski 1/2)

Problems	CoqInterval 2.0	Sollya	MetiTarski	NLCertify (not verified)	NLCertify (verified polys)	PVS/interval	HOL Light/ `verify_ineq`	PVS/Bernstein	HOL Light/ `REAL_SOS`
MT1	0.53	-	0.13	-	-	Failed	-	-	-
MT2	1.56	-	0.06	9.99⋆	Timeout	Failed	-	-	-
MT3	0.18	-	0.18	-	-	1.14	-	-	-
MT4	0.23	-	0.17	1.31⋆	18.95⋆	1.19	-	-	-
MT5	0.11⋆	-	0.05	-	-	1.24	-	-	-
MT6	0.15⋆	-	0.07	-	-	1.23	-	-	-
MT7	0.04	-	0.04	-	-	0.69	-	-	-
MT8	0.33	-	0.15	-	-	Timeout	-	-	-
MT9	0.52	-	0.46	-	-	Timeout	-	-	-
MT10	0.19	-	0.04	0.96	14.86	Failed	-	-	-
MT11	0.10	-	0.22	0.40	6.73	1.72	-	-	-
MT12	2.84	-	0.07	Timeout	Timeout	Timeout	-	-	-
MT13	0.98	-	0.07	11.82	137.91	Failed	-	-	-
MT14	0.07	-	0.06	-	-	0.89	-	-	-
MT15	0.15	-	0.07	-	-	0.98	-	-	-

Experimental Results (MT 2/2 + multivariate problems)

Problems	CoqInterval 2.0	Sollya	MetiTarski	NLCertify (not verified)	NLCertify (verified polys)	PVS/interval	HOL Light/ `verify_ineq`	PVS/Bernstein	HOL Light/ `REAL_SOS`
MT16	0.13	-	0.02	0.58*	8.23*	3.23	0.57*	-	-
MT17	0.11	-	0.06	0.22	4.06	1.27	0.23	-	-
MT18	0.16	-	0.02	0.21	2.46	0.69	0.75	-	-
MT19	0.52	-	Failed	5.09	74.55	Failed	1.92	-	-
MT20	3.09	-	0.05	2.63	44.21	Timeout	15.54	-	-
MT21	0.33	-	0.38	3.69	51.94	Failed	1.37	-	-
MT22	0.69	-	0.06	Timeout	Timeout	Failed	113.74	-	-
MT23	1.17	-	0.12	Failed	Failed	Failed	86.90	-	-
MT24	0.10	-	0.36	0.17	2.38	Failed	0.24	-	-
MT25	0.29	-	0.17	-	-	1.78	-	-	-
RD	0.25	-	0.02	1.88	66.01	1.67	0.48	3.26	Timeout
adaptiveLV	0.16	-	0.04	0.23	3.18	1.00	1.26	4.02	3.78
butcher	0.42	-	0.05	0.73	11.08	19.99	2.21	18.23	Timeout
magnetism	0.17	-	0.05	1.35	20.60	Timeout	313.75	Timeout	0.24

Conclusion

- Coq tactics to automatically and formally prove numerical bounds on real-valued expressions

Conclusion

- Coq tactics to automatically and formally prove numerical bounds on real-valued expressions
- All computations performed in Coq's logic, using interval arithmetic

Conclusion

- Coq tactics to automatically and formally prove numerical bounds on real-valued expressions
- All computations performed in Coq's logic, using interval arithmetic
- Implements bisection, automatic differentiation and Taylor models techniques to reduce the dependency effect

Conclusion

- Coq tactics to automatically and formally prove numerical bounds on real-valued expressions
- All computations performed in Coq's logic, using interval arithmetic
- Implements bisection, automatic differentiation and Taylor models techniques to reduce the dependency effect
- Regarding performance, CoqInterval is competitive w.r.t state-of-the-art inequality provers

Conclusion

- Coq tactics to automatically and formally prove numerical bounds on real-valued expressions
- All computations performed in Coq's logic, using interval arithmetic
- Implements bisection, automatic differentiation and Taylor models techniques to reduce the dependency effect
- Regarding performance, CoqInterval is competitive w.r.t state-of-the-art inequality provers
- Reference:
  http://www.irit.fr/publis/ACADIE/CoqInterval-JAR.pdf

Some future directions

- Bottleneck: Horner evaluation.
  Formalize alternative schemes that are amenable to formal methods?
- Certifying algorithms: Check polynomial approximations for special functions, by using certificates generated by Sollya?
- Reals/Coquelicot/CoqInterval/...: Increase automation for developing formal libraries of elementary functions more easily
- Symbolic-numeric methods: formally verify numerical solutions of differential equations $\rightsquigarrow$ on-going works (Thomas Sibut-Pinote)

Thanks for your attention!

How to install CoqInterval?

Browse http://coq-interval.gforge.inria.fr/ ; or follow the steps below.

```
# 1. Install OPAM (http://opam.ocaml.org/doc/Install.html)
$ opam init # create ~/.opam and propose to update the ~/.bashrc
$ opam switch install 4.02.3 # setup a recent version of OCaml
$ eval $(opam config env)

# 2. Enable the OPAM repository for stable Coq packages
$ opam repo add coq-released https://coq.inria.fr/opam/released

# 3. Install Coq + SSReflect + Flocq + Coquelicot + CoqInterval
$ opam install --jobs=2 coq.8.4.6 coq-interval
```