Formalizing mathematics, in practice

Workshop in Honour of Thierry Coquand's 60th Birthday

Assia Mahboubi

August 26th 2022

Inria, LS2N, Université de Nantes, Vrije Universiteit Amsterdam

European Research Council
Established by the European Commission

Thank you

- Mathematics, Algorithms and Proofs (MAP) community
- TYPES Summer School, Göteborg 2005
- FORMATH European project 2010-2013
- Univalent Foundations of Mathematics IAS Princeton 2012-2013
- ...

International Congress of Mathematicians 2022

Welcome to the 2022 Virtual ICM!

I would like to warmly welcome all participants in this virtual ICM. Organizing this event in a short timeframe, with limited human resources, has been a very challenging task. I sincerely hope that this effort proves to be successful and that all of you can, as a result, enjoy learning about the latest developments in mathematics.

Carlos E. Kenig, IMU president

Good Morning and a very warm Welcome to the ICM2022!

Please find below the overview of today´s program. Click on „view session" to join your selected stream. All sessions are displayed in **CEST**.

A Discord server has been set up to discuss the vICM lectures as they are being given. We encourage all speakers to join this server to interact with the audience in a more direct and extensive way. Follow this link: https://discord.gg/QVxUEv6bRu

Program

06 JUL WED | 07 JUL THU | 08 JUL FRI | **09 JUL SAT** | 10 JUL SUN | 11 JUL MON | 12 JUL TUE | 13 JUL WED | 14 JUL THU

All | Room 1 | Room 2 | Room 3 | Room 4 | Room 5 | Room 6 | Room 7 | Room 8 | Room 9 | Room 10 | Room 11

09 JULY
SATURDAY

CEST
10:15 - 11:15

Special Plenary Lecture

Kevin Buzzard - The rise of formalism in mathematics

- Room 1
- Kevin Buzzard
- Martin Hairer

View Session

07 JULY
THURSDAY

CEST
14:15 - 15:00

Special Sectional Lecture

Georges Gonthier - Computer proofs: teaching computers mathematics, and conversely

- Room 4
- Irit Dinur
- Georges Gonthier

View Session

Constructive Mathematics

Coq's standard library: transitive closure of a relation

The transitive closure R^+ of R is the smallest transitive relation containing R.

Coq's standard library: transitive closure of a relation

The transitive closure R^+ of R is the smallest transitive relation containing R.

Constructed as:

$$R^+ = \bigcup_{i=1}^{+\infty} R^i \quad \text{with} \quad R_1 = R \quad R_{n+1} = R \circ R^n$$

Coq's standard library: transitive closure of a relation

The transitive closure R^+ of R is the smallest transitive relation containing R.

Constructed as:

$$R^+ = \bigcup_{i=1}^{+\infty} R^i \quad \text{with} \quad R_1 = R \quad R_{n+1} = R \circ R^n$$

Formalized as:

```
Inductive clos_trans (x : A) : A -> Prop :=
  | t_step (y : A) : R x y -> clos_trans x y
  | t_trans (y z : A) : clos_trans x y -> clos_trans y z -> clos_trans x z.
```

Equality up to permutation

In Coq's (or Agda's) standard library, for an arbitrary type `A`:

```
Inductive Permutation : list A -> list A -> Prop :=
| perm_nil: Permutation [] []
| perm_skip x l1 l2 : Permutation l1 l2 -> Permutation (x :: l1) (x :: l2)
| perm_swap x y l : Permutation (y :: x:: l) (x :: y :: l)
| perm_trans l l' l'' : Permutation l1 l2 -> Permutation l2 l3 -> Permutation l1 l3.
```

Equality up to permutation

In Coq's (or Agda's) standard library, for an arbitrary type `A`:

```
Inductive Permutation : list A -> list A -> Prop :=
| perm_nil: Permutation [] []
| perm_skip x l1 l2 : Permutation l1 l2 -> Permutation (x :: l1) (x :: l2)
| perm_swap x y l : Permutation (y :: x:: l) (x :: y :: l)
| perm_trans l l' l'' : Permutation l1 l2 -> Permutation l2 l3 -> Permutation l1 l3.
```

In Mathematical Components, for a type `A` with decidable equality:

```
(* _ == _ : nat -> nat -> bool *)
Definition perm_eq (l1 l2 : list A) : bool :=
  all [pred x | count_mem x l1 == count_mem x l2] (l1 ++ l2).
```

Equality up to permutation

In Coq's (or Agda's) standard library, for an arbitrary type `A`:

```
Inductive Permutation : list A -> list A -> Prop :=
| perm_nil: Permutation [] []
| perm_skip x l1 l2 : Permutation l1 l2 -> Permutation (x :: l1) (x :: l2)
| perm_swap x y l : Permutation (y :: x:: l) (x :: y :: l)
| perm_trans l l' l'' : Permutation l1 l2 -> Permutation l2 l3 -> Permutation l1 l3.
```

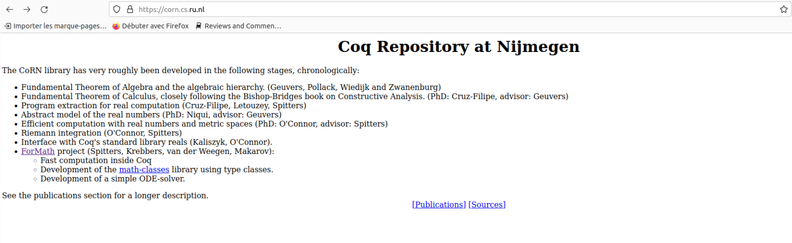
In Mathematical Components, for a type `A` with decidable equality:

```
(* _ == _ : nat -> nat -> bool *)
Definition perm_eq (l1 l2 : list A) : bool :=
  all [pred x | count_mem x l1 == count_mem x l2] (l1 ++ l2).
```

```
Lemma perm_cat2l l1 l2 l3 : perm_eq (l1 ++ l2) (l1 ++ l3) = perm_eq l2 l3.
Proof.
apply/permP/permP=> eq23 a; apply/eqP;
  by move/(_ a)/eqP: eq23; rewrite !count_cat eqn_add2l.
Qed.
```

C-CoRN: Coq Repository at Nijmegen

A library for constructive algebra and analysis, started circa 2000.



https://corn.cs.ru.nl

Importer les marque-pages... Débuter avec Firefox Reviews and Commen...

Coq Repository at Nijmegen

The CoRN library has very roughly been developed in the following stages, chronologically:

- Fundamental Theorem of Algebra and the algebraic hierarchy. (Geuvers, Pollack, Wiedijk and Zwanenburg)
- Fundamental Theorem of Calculus, closely following the Bishop-Bridges book on Constructive Analysis. (PhD: Cruz-Filipe, advisor: Geuvers)
- Program extraction for real computation (Cruz-Filipe, Letouzey, Spitters)
- Abstract model of the real numbers (PhD: Niqui, advisor: Geuvers)
- Efficient computation with real numbers and metric spaces (PhD: O'Connor, advisor: Spitters)
- Riemann integration (O'Connor, Spitters)
- Interface with Coq's standard library reals (Kaliszyk, O'Connor).
- ForMath project (Spitters, Krebbers, van der Weegen, Makarov):
  - Fast computation inside Coq
  - Development of the math-classes library using type classes.
  - Development of a simple ODE-solver.

See the publications section for a longer description.

[Publications] [Sources]

- Every non-constant single-variable polynomial with complex coefficients has at least one complex root. (1806)
- The integral of a function provides one of its antiderivatives. (circa 1700)
- . . .

Structures, setoids, hierarchies

Structures as dependent pairs:

$$(T, p_T) : \Sigma(x : \mathrm{Type}) S\ x$$

Or rather, tuples:

```
Record invType := {sort : Type; inv : sort -> sort; idem : involution inv}
```

[Telescopic mappings in typed lambda calculus, N. G. de Bruijn (1974, 1991), link]

[Dependently typed records for representing mathematical structure, R. Pollack (2000) link]

Structures, setoids, hierarchies

Quotients as constructive setoids:

```
Record CSetoid : Type := {
cs_crr : Type;
cs_eq : relation cs_crr; (* equality x ~ y *)
cs_ap : relation cs_crr; (* apartness x # y *)
cs_proof : is_CSetoid cs_crr cs_eq cs_ap} (* constructive setoid axioms *)
```

Constructive setoid axioms, about apartness:

- irreflexivity: $\neg(x \sharp x)$
- symmetry: $(x \sharp y) \Rightarrow (y \sharp x)$
- co-transitivity: $(x \sharp y) \Rightarrow (x \sharp z) \vee (z \sharp y)$
- tightness: $\neg(x \sharp y) \Leftrightarrow (x \sim y)$

Structures, setoids, hierarchies

Quotients as constructive setoids:

```
Record CSetoid : Type := {
cs_crr : Type;
cs_eq : relation cs_crr; (* equality x ~ y *)
cs_ap : relation cs_crr; (* apartness x # y *)
cs_proof : is_CSetoid cs_crr cs_eq cs_ap} (* constructive setoid axioms *)
```

Constructive setoid axioms, about apartness:

- irreflexivity: $\neg(x \sharp x)$
- symmetry: $(x \sharp y) \Rightarrow (y \sharp x)$
- co-transitivity: $(x \sharp y) \Rightarrow (x \sharp z) \vee (z \sharp y)$
- tightness: $\neg(x \sharp y) \Leftrightarrow (x \sim y)$?

"However ... we wanted the notion of constructive setoid to be a refinement of the notion of setoid."

[A Constructive Algebraic Hierarchy in Coq, H. Geuvers, R. Pollack, F. Wiedijk, J. Zwanenburg, JSC (2002), link]

Coercion (explicit subtyping) based inheritance:

```
Record CRing : Type :=
{ cr_crr :> CGroup;
  cr_one : cr_crr;
  cr_zero : cr_crr;
  cr_mult: CSetoid_bin_opp cr_crr;
  cr_proof : is_CRing cr_crr cr_one cr_mult}
```

where `cr_crr : CRing -> CGroup` is a coercion.

Structures, setoids, hierarchies

Coercion (explicit subtyping) based inheritance:

```
Record CRing : Type :=
{ cr_crr :> CGroup;
  cr_one : cr_crr;
  cr_zero : cr_crr;
  cr_mult: CSetoid_bin_opp cr_crr;
  cr_proof : is_CRing cr_crr cr_one cr_mult}
```

where `cr_crr : CRing -> CGroup` is a coercion.

Later improved by a type class based hierarchy (MathClasses).

[Type Classes for Mathematics in Type Theory, B. Spitters, E. van der Weegen, MSCS (2011), link]

[Type classes for efficient exact real arithmetic in Coq, R. Krebbers, B. Spitters, LMCS (2013), link]

Computable real numbers

- Computable real numbers à la Bishop - Bridges

  [A monadic, functional implementation of real numbers, R. O'Connor, 2007, MSCS, link]

- Connection with Coq's standard library for classical reals

  [A monadic, functional implementation of real numbers, C. Kaliszyk, R. O'Connor, 2009, JFR. link]

- Speed up using machine integers, expanded and better structured

  [Type classes for efficient exact real arithmetic in Coq, R. Krebbers, B. Spitters, 2013, LMCS, link]

Computable real numbers

- Computable real numbers à la Bishop - Bridges

  [A monadic, functional implementation of real numbers, R. O'Connor, 2007, MSCS, link]

- Connection with Coq's standard library for classical reals

  [A monadic, functional implementation of real numbers, C. Kaliszyk, R. O'Connor, 2009, JFR. link]

- Speed up using machine integers, expanded and better structured

  [Type classes for efficient exact real arithmetic in Coq, R. Krebbers, B. Spitters, 2013, LMCS, link]

```
Lemma ground_ineq : 0.41078129 < sin E. Proof. <immediate>. Qed.
```

Computable real numbers

- Computable real numbers à la Bishop - Bridges

  [A monadic, functional implementation of real numbers, R. O'Connor, 2007, MSCS, link]

- Connection with Coq's standard library for classical reals

  [A monadic, functional implementation of real numbers, C. Kaliszyk, R. O'Connor, 2009, JFR. link]

- Speed up using machine integers, expanded and better structured

  [Type classes for efficient exact real arithmetic in Coq, R. Krebbers, B. Spitters, 2013, LMCS, link]

```
Lemma ground_ineq : 0.41078129 < sin E. Proof. <immediate>. Qed.
```

Computed 25 decimals of sine(e) in 0.1s, 500 decimals in 1.9s.

Computational Mathematics

2006: Verified four color theorem

- First (computer-aided) proof: W. Appel and K. Haken, 1976
- Formally verified proof: G. Gonthier, with B. Werner, 2006
- Uses (optimized) computation inside logic

2006: Verified four color theorem

- First (computer-aided) proof: W. Appel and K. Haken, 1976
- Formally verified proof: G. Gonthier, with B. Werner, 2006
- Uses (optimized) computation inside logic

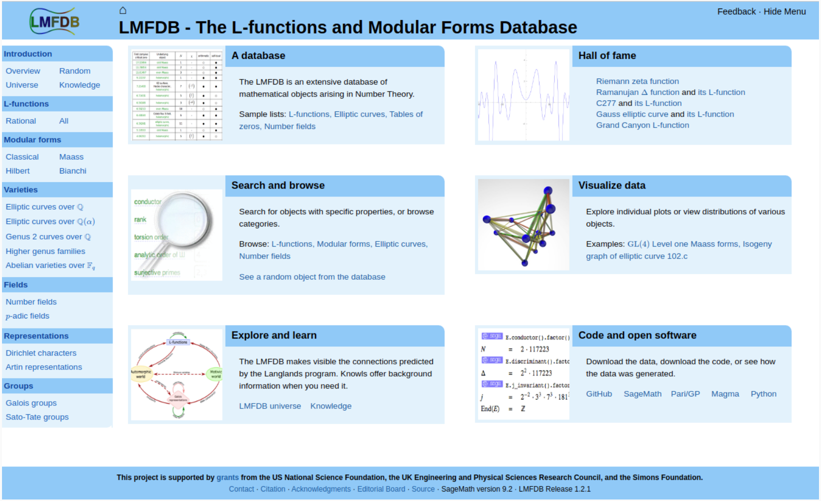
```
Variable (m : map ℝ).

Theorem four_color_finite : finite_simple_map m -> colorable_with 4 m.
Proof.
intros fin_m.
pose proof (discretize.discretize_to_hypermap fin_m) as [G planarG colG].
exact (colG (combinatorial4ct.four_color_hypermap planarG)).
Qed.


Theorem four_color : simple_map m -> colorable_with 4 m.
Proof. exact (finitize.compactness_extension four_color_finite). Qed.
```

[Formal Proof—The Four-Color Theorem, G. Gonthier (2008) link]

2006: Verified four color theorem

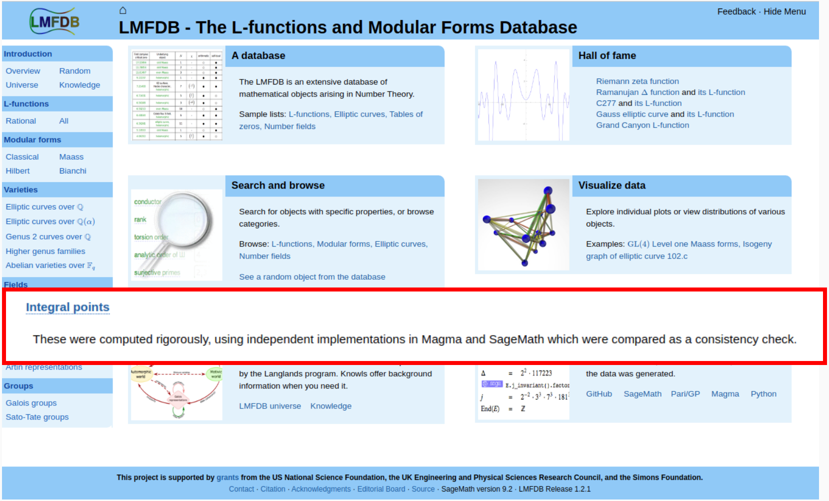
- Verification of the non-trivial computational part of the proof
- Formalization of a corpus of modern combinatorics
- Formal proof engineering methodology
- Novel/rediscovered mathematics

Handbooks



LMFDB
LMFDB - The L-functions and Modular Forms Database
Feedback · Hide Menu
Introduction
Overview Random
Universe Knowledge
L-functions
Rational All
Modular forms
Classical Maass
Hilbert Bianchi
Varieties
Elliptic curves over $\mathbb{Q}$
Elliptic curves over $\mathbb{Q}(\alpha)$
Genus 2 curves over $\mathbb{Q}$
Higher genus families
Abelian varieties over $\mathbb{F}_q$
Fields
Number fields
p-adic fields
Representations
Dirichlet characters
Artin representations
Groups
Galois groups
Sato-Tate groups
A database
The LMFDB is an extensive database of mathematical objects arising in Number Theory.
Sample lists: L-functions, Elliptic curves, Tables of zeros, Number fields
Hall of fame
Riemann zeta function
Ramanujan Δ function and its L-function
C277 and its L-function
Gauss elliptic curve and its L-function
Grand Canyon L-function
Search and browse
conductor
rank
torsion order
analytic order of Ш
surjective primes
Search for objects with specific properties, or browse categories.
Browse: L-functions, Modular forms, Elliptic curves, Number fields
See a random object from the database
Visualize data
Explore individual plots or view distributions of various objects.
Examples: $\mathrm{GL}(4)$ Level one Maass forms, Isogeny graph of elliptic curve 102.c
Explore and learn
L-functions
Automorphic world
Motivic world
The LMFDB makes visible the connections predicted by the Langlands program. Knowls offer background information when you need it.
LMFDB universe Knowledge
Code and open software
E.conductor().factor()
$N = 2 \cdot 117223$
E.discriminant().facto
$\Delta = 2^2 \cdot 117223$
E.j_invariant().factor
$j = 2^{-2} \cdot 3^3 \cdot 7^3 \cdot 181$
$\mathrm{End}(E) = \mathbb{Z}$
Download the data, download the code, or see how the data was generated.
GitHub SageMath Pari/GP Magma Python
This project is supported by grants from the US National Science Foundation, the UK Engineering and Physical Sciences Research Council, and the Simons Foundation.
Contact · Citation · Acknowledgments · Editorial Board · Source · SageMath version 9.2 · LMFDB Release 1.2.1

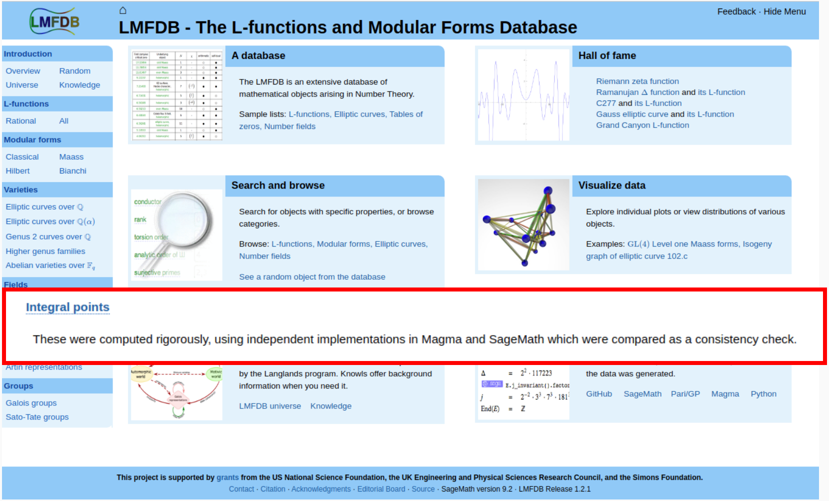
Handbooks



Integral points

These were computed rigorously, using independent implementations in Magma and SageMath which were compared as a consistency check.

Handbooks



Fact

Today, no explicit policy exist for auditing software that produce proof steps.

Cross-verification is not enough

In SymPy 1.5.1 [1], compare

```
>>> simplify(hyper([n],[m],x).subs({m:-1, n:-1, x:1}))
2
```

with

```
>>> simplify(hyper([n],[m],x).subs(m, n)).subs({n:-1, x:1})
E
```

[1] Example suggested by F. Johansson.

Cross-verification is not enough

In SymPy 1.5.1 [1], compare

```
>>> simplify(hyper([n],[m],x).subs({m:-1, n:-1, x:1}))
                                                            2
```

with

```
>>> simplify(hyper([n],[m],x).subs(m, n)).subs({n:-1, x:1})
                                                           E
```

⇒ Post-hoc verification techniques cannot apply.

[1] Example suggested by F. Johansson.

Cross-verification is not enough

In SymPy 1.5.1 [1], compare

```
>>> simplify(hyper([n],[m],x).subs({m:-1, n:-1, x:1}))
                                                2
```

with

```
>>> simplify(hyper([n],[m],x).subs(m, n)).subs({n:-1, x:1})
                                                E
```

⇒ Post-hoc verification techniques cannot apply.

Wolfram Language (Mathematica) exhibit the exact same phenomenon.

⇒ Cross-verification is not enough.

[1] Example suggested by F. Johansson.

Formally verified rigorous computations

Ternary Goldbach conjecture is true (H. Helfgott, 2013)

Every odd integer greater than 5 is the sum of three primes.

Formally verified rigorous computations

Ternary Goldbach conjecture is true (H. Helfgott, 2013)

Every odd integer greater than 5 is the sum of three primes.

MAJOR ARCS FOR GOLDBACH'S PROBLEM 35

By Cauchy-Schwarz, this is at most

$$\sqrt{\frac{1}{2\pi}\int_{-\frac{1}{2}-i\infty}^{-\frac{1}{2}+i\infty}\left|\frac{L'(s,\chi)}{L(s,\chi)}\cdot\frac{1}{s}\right|^2|ds|}\cdot\sqrt{\frac{1}{2\pi}\int_{-\frac{1}{2}-i\infty}^{-\frac{1}{2}+i\infty}|G_\delta(s)s|^2\,|ds|}$$

By (4.12),

$$\begin{aligned}\sqrt{\int_{-\frac{1}{2}-i\infty}^{-\frac{1}{2}+i\infty}\left|\frac{L'(s,\chi)}{L(s,\chi)}\cdot\frac{1}{s}\right|^2|ds|} &\leq \sqrt{\int_{-\frac{1}{2}-i\infty}^{-\frac{1}{2}+i\infty}\left|\frac{\log q}{s}\right|^2|ds|}\\ &\quad+\sqrt{\int_{-\infty}^{\infty}\frac{\left|\frac{1}{2}\log\left(\tau^2+\frac{9}{4}\right)+4.1396+\log\pi\right|^2}{\frac{1}{4}+\tau^2}d\tau}\\ &\leq\sqrt{2\pi}\log q+\sqrt{226.844},\end{aligned}$$

where we compute the last integral numerically.[4]

[4] By a rigorous integration from $\tau = -100000$ to $\tau = 100000$ using VNODE-LP [Ned06], which runs on the PROFIL/BIAS interval arithmetic package [Knü99].

Formally verified rigorous computations

Ternary Goldbach conjecture is true (H. Helfgott, 2013)

Every odd integer greater than 5 is the sum of three primes.

By Cauchy-Schwarz, this is at most

$$\sqrt{\frac{1}{2\pi}\int_{-\frac{1}{2}-i\infty}^{-\frac{1}{2}+i\infty}\left|\frac{L'(s,\chi)}{L(s,\chi)}\cdot\frac{1}{s}\right|^2|ds|}\cdot\sqrt{\frac{1}{2\pi}\int_{-\frac{1}{2}-i\infty}^{-\frac{1}{2}+i\infty}|G_\delta(s)s|^2\,|ds|}$$

By (4.12),

$$\begin{aligned}\sqrt{\int_{-\frac{1}{2}-i\infty}^{-\frac{1}{2}+i\infty}\left|\frac{L'(s,\chi)}{L(s,\chi)}\cdot\frac{1}{s}\right|^2|ds|} &\leq \sqrt{\int_{-\frac{1}{2}-i\infty}^{-\frac{1}{2}+i\infty}\left|\frac{\log q}{s}\right|^2|ds|}\\ &\quad+\sqrt{\int_{-\infty}^{\infty}\frac{\left|\frac{1}{2}\log\left(\tau^2+\frac{9}{4}\right)+4.1396+\log\pi\right|^2}{\frac{1}{4}+\tau^2}d\tau}\\ &\leq \sqrt{2\pi}\log q+\sqrt{226.844},\end{aligned}$$

where we compute the last integral numerically.[4]

[4] By a rigorous integration from $\tau=-100000$ to $\tau=100000$ using VNODE-LP [Ned06], which runs on the PROFIL/BIAS interval arithmetic package [Knü99].

- This estimation is wrong (although the proof can be repaired).

[Formally Verified Approximations of Definite Integrals - A. Mahboubi, G. Melquiond, Th. Sibut-Pinote, JAR 2018]

Catalog of univariate elementary functions

Described by an abstract syntax:

$$
\begin{array}{rcl}
\mathcal{E} & := & x \mid \mathbb{F} \mid \pi \mid \\
& & \mathcal{E} + \mathcal{E} \mid \mathcal{E} - \mathcal{E} \mid \mathcal{E} \times \mathcal{E} \mid \mathcal{E} \div \mathcal{E} \mid -\mathcal{E} \mid \|\mathcal{E}\| \mid \\
& & \sqrt{\mathcal{E}} \mid \mathcal{E}^k \mid \\
& & \cos(\mathcal{E}) \mid \sin(\mathcal{E}) \mid \tan(\mathcal{E}) \mid \operatorname{atan}(\mathcal{E}) \mid \\
& & \exp(\mathcal{E}) \mid \ln(\mathcal{E})
\end{array}
$$

Specification of interval extensions

The library implements interval extensions for each elementary function:

- $[e]_{\mathbb{R}_\perp} \quad : \quad \mathbb{R}_\perp \to \mathbb{R}_\perp$
- $[e]_{\mathbb{I}_\perp} \quad : \quad \mathbb{I}_\perp \to \mathbb{I}_\perp$

Specification of interval extensions

The library implements interval extensions for each elementary function:

- $[e]_{\mathbb{R}_\perp} \quad : \quad \mathbb{R}_\perp \to \mathbb{R}_\perp$
- $[e]_{\mathbb{I}_\perp} \quad : \quad \mathbb{I}_\perp \to \mathbb{I}_\perp$

Example:

$$\forall \mathrm{i} \in \mathbb{I}_\perp, \forall x \in \boldsymbol{i}, \quad \pi + cos(x) \in \boldsymbol{\pi} + \boldsymbol{cos}(\boldsymbol{i})$$

Specification of interval extensions

The library implements interval extensions for each elementary function:

- $[e]_{\mathbb{R}_\perp} \quad : \quad \mathbb{R}_\perp \to \mathbb{R}_\perp$
- $[e]_{\mathbb{I}_\perp} \quad : \quad \mathbb{I}_\perp \to \mathbb{I}_\perp$

Example:

$$\forall \mathrm{i} \in \mathbb{I}_\perp, \forall x \in \boldsymbol{i}, \quad \pi + cos(x) \in \boldsymbol{\pi} + \boldsymbol{cos}(\boldsymbol{i})$$

Correctness theorem of interval extensions:

$$\forall e \in \mathcal{E}, \forall \boldsymbol{i} \in \mathbb{I}_\perp, \forall x \in \boldsymbol{i}, \quad [e]_{\mathbb{R}_\perp}(x) \in [e]_{\mathbb{I}_\perp}(\boldsymbol{i})$$

Formally verified approximations

Initial problem:

$$\int_a^b f(x)dx \in [m, M] \quad ?$$

Formally verified approximations

Entry in the Catalog:

$$\int_{[e_a]_{\mathbb{R}}}^{[e_b]_{\mathbb{R}}} [e_f]_{\mathbb{R}} dx \in [m, M] \quad ?$$

Formally verified approximations

Verified computation:

$$\int_{[e_a]_{\mathbb{R}}}^{[e_b]_{\mathbb{R}}} [e_f]_{\mathbb{R}}\, dx$$

Formally verified approximations

Verified computation:

$$\int_{[e_a]_{\mathbb{R}}}^{[e_b]_{\mathbb{R}}} [e_f]_{\mathbb{R}}\, dx \in \int_{[e_a]_{\mathbb{I}}}^{[e_b]_{\mathbb{I}}} [e_f]_{\mathbb{I}}\, dx$$

Formally verified approximations

Verified computation:

$$\int_{[e_a]_{\mathbb{R}}}^{[e_b]_{\mathbb{R}}} [e_f]_{\mathbb{R}} dx \in \int_{[e_a]_{\mathbb{I}}}^{[e_b]_{\mathbb{I}}} [e_f]_{\mathbb{I}} dx \subseteq [m, M]$$

[Formally Verified Approximations of Definite Integrals, A. Mahboubi, G. Melquiond, Th. Sibut-Pinote, JAR 2018]

Formally verified approximations

Verified computation, using rigorous polynomial approximations:

$$\int_{[e_a]_{\mathbb{R}}}^{[e_b]_{\mathbb{R}}} [e_f]_{\mathbb{R}}\, dx \in \int_{[e_a]_{\mathbb{TM}}}^{[e_b]_{\mathbb{TM}}} [e_f]_{\mathbb{TM}}\, dx \subseteq [m, M]$$

[Formally Verified Approximations of Definite Integrals, A. Mahboubi, G. Melquiond, Th. Sibut-Pinote, JAR 2018]

Formally verified rigorous computations

$$\int_0^1 |(x^4 + 10x^3 + 19x^2 - 6x - 6e^x|dx \simeq 11.14731055005714$$

Formally verified rigorous computations

$$\int_0^1 |(x^4 + 10x^3 + 19x^2 - 6x - 6e^x| dx \simeq 11.14731055005714$$

- Octave's `quad`/`quadgk`: only 10/9 correct digits;
- INTLAB verifyquad: false answer without warning;
- VNODE-LP: cannot be used because of the absolute value.

Formally verified rigorous computations

$$\int_0^1 |(x^4 + 10x^3 + 19x^2 - 6x - 6e^x| dx \simeq 11.14731055005714$$

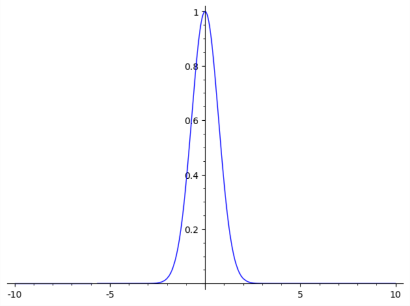
- Octave's `quad`/`quadgk`: only 10/9 correct digits;
- INTLAB verifyquad: false answer without warning;
- VNODE-LP: cannot be used because of the absolute value.

INTLAB bug report (2016) $\Rightarrow$ Removal of the support for the absolute value

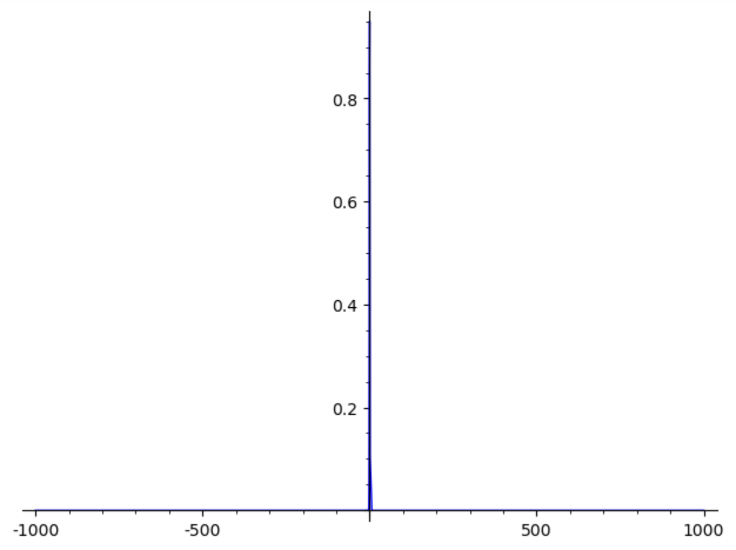
Formally verified rigorous computations

```
arb_sqrt(arb_t z, const arb_t x, slong prec)
{
    mag_t rx, zr;
    int inexact;

    if (mag_is_zero(arb_radref(x)))
    {
        arb_sqrt_arf(z, arb_midref(x), prec);
    }
    else if (arf_is_special(arb_midref(x)) ||
              arf_sgn(arb_midref(x)) < 0 || mag_is_inf(arb_radref(x)))
    {
        if (arf_is_pos_inf(arb_midref(x)) && mag_is_finite(arb_radref(x)))
            arb_sqrt_arf(z, arb_midref(x), prec);
        else
            arb_indeterminate(z);
    }
    else  /* now both mid and rad are non-special values, mid > 0 */
    {
        slong acc;

        acc = _fmpz_sub_small(ARF_EXPREF(arb_midref(x)), MAG_EXPREF(arb_radref(x)));
        acc = FLINT_MIN(acc, prec);
        prec = FLINT_MIN(prec, acc + MAG_BITS);
        prec = FLINT_MAX(prec, 2);

        if (acc < 0)
        {
            arb_indeterminate(z);
        }
        else if (acc <= 20)
        {
            mag_t t, u;

            mag_init(t);
            mag_init(u);

            arb_get_mag_lower(t, x);

            if (mag_is_zero(t) && arb_contains_negative(x))
            {
```

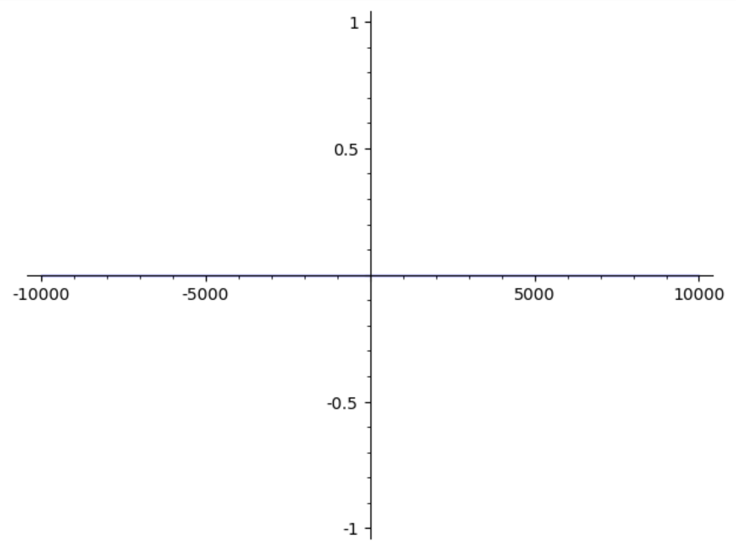
Plotting $exp(-x^2)$ with sagemath



Plotting $exp(-x^2)$ with sagemath

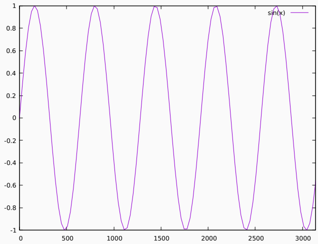


Plotting $exp(-x^2)$ with sagemath



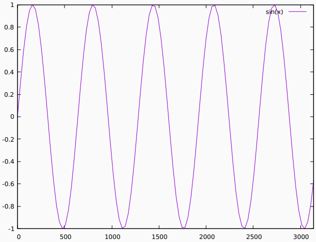
Plotting $sin(x)$ for $x \in [0, 3141]$

Plotting $sin(x)$ **for** $x \in [0, 3141]$

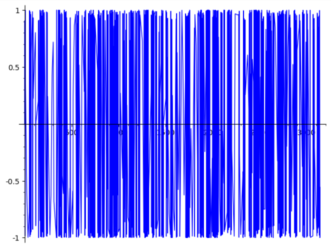


Gnuplot

Plotting $sin(x)$ for $x \in [0, 3141]$



Gnuplot



Sagemath

Faithful plotting is hard

Issues:

- Sampling
- Accuracy
- Bugs

Faithful plotting is hard

Issues:

- Sampling
- Accuracy
- Bugs

Desired properties:

- Correctness: blank pixels are not traversed by the function graph
- Completeness: filled pixels are traversed by the function graph

Faithful plotting is hard

Issues:

- Sampling
- Accuracy
- Bugs

Desired properties:

- Correctness: blank pixels are not traversed by the function graph
- Completeness: filled pixels are traversed by the function graph

$\Rightarrow$ Formally verified plots: guarantee correctness and strive for completeness

Generating formally verified plots

To obtain a verified plot for $f(x)$ for $x \in X$:

- Partition X in $(X_i)_{i=1\ldots n}$
- Produce a list $(\ell_i)_{i=1\ldots n}$ of intervals
- Ensure (with a formal proof) that for every $i = 1 \ldots n$:

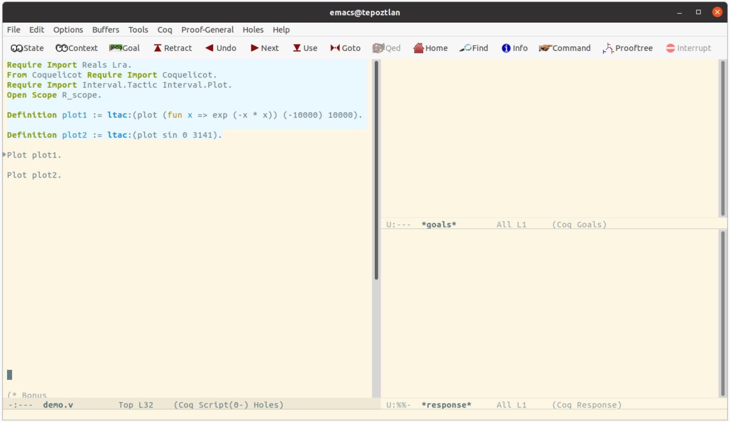
$$\forall x \in X_i, f(x) \in \ell_i$$

- Fill the corresponding pixels.

Rigorous polynomial approximation make computations efficient enough.

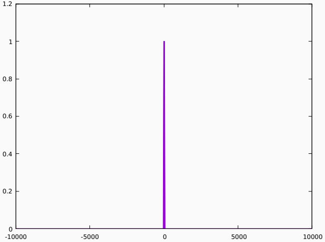
[Plotting in a formally verified way, G. Melquiond, F-IDE 2021]

Demo



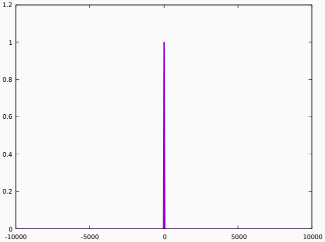
emacs@tepoztlan
File Edit Options Buffers Tools Coq Proof-General Holes Help
State Context Goal Retract Undo Next Use Goto Qed Home Find Info Command Prooftree Interrupt
Require Import Reals Lra.
From Coquelicot Require Import Coquelicot.
Require Import Interval.Tactic Interval.Plot.
Open Scope R_scope.
Definition plot1 := ltac:(plot (fun x => exp (-x * x)) (-10000) 10000).
Definition plot2 := ltac:(plot sin 0 3141).
Plot plot1.
Plot plot2.
(* Bonus
-:--- demo.v Top L32 (Coq Script(0-) Holes)
U:--- *goals* All L1 (Coq Goals)
U:%%- *response* All L1 (Coq Response)

Demo

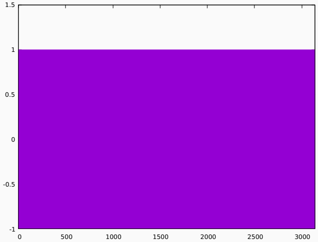


Verified plot of $exp(-x^2)$ for $x \in [-10000, 10000]$

Demo



Verified plot of $exp(-x^2)$ for $x \in [-10000, 10000]$



Verified plot of $sin(x)$ for $x \in [0, 3141]$

Contemporary Mathematics

2013: Odd order theorem formally verified

Theorem (W. Feit - J. G. Thompson, 1963)

Every finite group of odd order is solvable.

[A formal proof of the Odd Order theorem, Gonthier et al., Proc. of ITP 2013]

2013: Odd order theorem formally verified

Problems:

- Maintenance
- Readability of mathematical statements and proofs scripts
- Performance issues on the interactive prover side
- (Keep the proof constructive)

Mathematical notation

- Inference:

```
    Definition det (n : nat) (A : 'M_n[R]) : R := \sum_(s : S_n) (-1) ^+ s * \prod_i A i
        (s i)
```

$$det(A) = \sum_{\sigma \in S_n} (-1)^{\epsilon_\sigma} \prod_{i=1}^{n} a_{i\sigma(i)}$$

- Linguistics

```
Theorem third_isog (G H K : {group gT}) : H \subset K -> H <| G -> K <| G
    -> (G / H) / (K / H) \isog (G / K).
```

$$(G/H)/(K/H) \sim (G/K) \quad \text{when } H \subset K, H \lhd G, K \lhd G$$

From the Odd order theorem to the Mathematical Components library

- types with decidable equality and choice operator
- h-sets and Hedberg theorem
- type classes / unification hints hierarchies
- conversion / small scale reflexion
- enhanced support for forward chaining in the tactic language
- rewrite the mathematics

Today: Mathematics in the making

Lean is an interactive prover based on the Calculus of Inductive Constructions.

- 2017: Start of mathlib, today Lean's de facto the standard library

[The Lean Mathematical Library, The mathlib Community, Proc of CPP'2020]

Today: Mathematics in the making

Lean is an interactive prover based on the Calculus of Inductive Constructions.

- 2017: Start of mathlib, today Lean's de facto the standard library

  [The Lean Mathematical Library, The mathlib Community, Proc of CPP'2020]

- 2018: Field medal awarded to P. Scholze

Today: Mathematics in the making

Lean is an interactive prover based on the Calculus of Inductive Constructions.

- 2017: Start of mathlib, today Lean's de facto the standard library

  [The Lean Mathematical Library, The mathlib Community, Proc of CPP'2020]

- 2018: Field medal awarded to P. Scholze
- 2019: Definition of perfectoid spaces in Lean

  [Formalizing perfectoid spaces - K. Buzzard, J. Commelin, P. Massot, Proc. of CPP'2020]

Today: Mathematics in the making

```
-- We fix a prime number p
parameter (p : primes)

/-- A perfectoid ring is a Huber ring that is complete, uniform,
that has a pseudo-uniformizer whose p-th power divides p in the power bounded subring,
and such that Frobenius is a surjection on the reduction modulo p.-/
structure perfectoid_ring (R : Type) [Huber_ring R] extends Tate_ring R : Prop :=
(complete  : is_complete_hausdorff R)
(uniform   : is_uniform R)
(ramified  : ∃ ϖ : pseudo_uniformizer R, ϖ^p ∣ p in Rᵒ)
(Frobenius : surjective (Frob Rᵒ/p))

/-
CLVRS ("complete locally valued ringed space") is a category
whose objects are topological spaces with a sheaf of complete topological rings
and an equivalence class of valuation on each stalk, whose support is the unique
maximal ideal of the stalk; in Wedhorn's notes this category is called 𝒱.
A perfectoid space is an object of CLVRS which is locally isomorphic to Spa(A) with
A a perfectoid ring. Note however that CLVRS is a full subcategory of the category
`PreValuedRingedSpace` of topological spaces equipped with a presheaf of topological
rings and a valuation on each stalk, so the isomorphism can be checked in
PreValuedRingedSpace instead, which is what we do.
-/

/-- Condition for an object of CLVRS to be perfectoid: every point should have an open
neighbourhood isomorphic to Spa(A) for some perfectoid ring A.-/
def is_perfectoid (X : CLVRS) : Prop :=
∀ x : X, ∃ (U : opens X) (A : Huber_pair) [perfectoid_ring A],
  (x ∈ U) ∧ (Spa A ≅ U)

/-- The category of perfectoid spaces.-/
def PerfectoidSpace := {X : CLVRS // is_perfectoid X}

end
```

Today: Mathematics in the making

- 2017: Start of mathlib, today Lean's de facto the standard library

  [The Lean Mathematical Library, The mathlib Community, Proc of CPP'2020]

- 2018: Field medal awarded to P. Scholze
- 2019: Definition of perfectoid spaces in Lean

  [Formalizing perfectoid spaces - K. Buzzard, J. Commelin, P. Massot, Proc. of CPP'2020]

Today: Mathematics in the making

- 2017: Start of mathlib, today Lean's de facto the standard library

  [The Lean Mathematical Library, The mathlib Community, Proc of CPP'2020]

- 2018: Field medal awarded to P. Scholze
- 2019: Definition of perfectoid spaces in Lean

  [Formalizing perfectoid spaces - K. Buzzard, J. Commelin, P. Massot, Proc. of CPP'2020]

- 2022: Liquid Tensor Experiment: J. Commelin et al.

  [Half a year of the Liquid Tensor Experiment: Amazing developments, P. Scholze on the Xena blog, 2021]

Quoting P. Schölze about the Liquid Tensor experiment:

"(...) This makes the rest of the proof of the Liquid Tensor Experiment considerably more explicit and more elementary, removing any use of stable homotopy theory. I expect that Commelin's complex may become a standard tool in the coming years."

"(...) this made me realize that actually the key thing happening is a reduction from a non-convex problem over the reals to a convex problem over the integers."

If more mathematicians start using proof assistants

- Publications

If more mathematicians start using proof assistants

- Publications
- Teaching

If more mathematicians start using proof assistants

- Publications
- Teaching
- Collaborations

If more mathematicians start using proof assistants

- Publications
- Teaching
- Collaborations
- Creativity

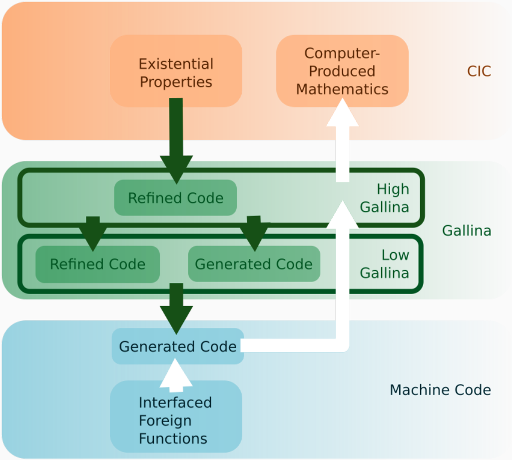
G. Huet: "Formal Mathematics in COC is fun"

- Novel community of users of CIC, with different motivations
- Many exciting projects (e.g., P. Massot's project about sphere eversion)
- Impact on the implementation of interactive provers
- But difficult non-technological questions remain

  modularity, hierarchies, isomorphisms,. . .

Can we make symbolic computation fast and correct?

Joint work in progress with G. Melquiond et al.



The FRESCO project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101001995)