Preuves formelles mutatis mutandis

Assia Mahboubi
En collaboration avec Enzo Crance et Cyril Cohen

Formalisation des mathématiques et types dépendants – 2 Juin 2025

Inria, LS2N, Nantes Université, Vrije Universiteit Amsterdam

Formalization of Mathematics and Dependent Types

Interational Congress of Mathematicians 2022

Plenary
Special Plenary Lecture
Room 1

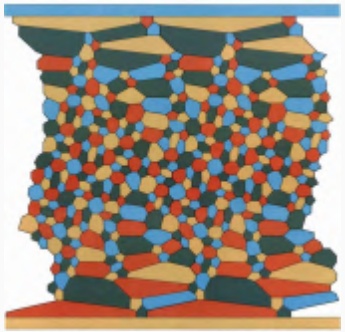
Saturday, July 9, 10:15 - 11:15

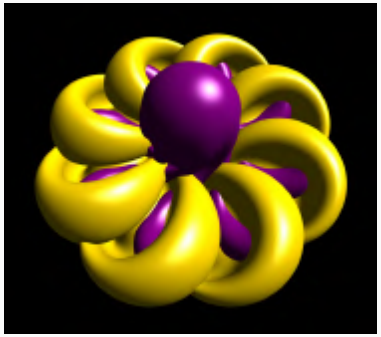
The rise of formalism in mathematics

Lecture on proof formalisation for ordinary mathematicians
Abstract:
Formalism is the art of writing down what you actually mean. Mathematics has a rich history of formalisation: Euclid, Russell–Whitehead and Bourbaki all tried it. This century Avigad, Hales and Gonthier have shown us that there is another way. Now a new generation of young people are formalising algebra, analysis, category theory, combinatorics, geometry, number theory, topology and more at Masters level and beyond, this time using a computer. Lean's mathematics library **mathlib** contains nearly a million lines of free and open source code corresponding to proofs of over 80,000 theorems such as the fundamental theorem of Galois theory, and it is growing fast. AIs trained on the library have solved IMO problems by themselves. What is happening? This is not about making sure the papers are right. This is not about making a computer program which will print out a one billion line proof of the Birch and Swinnerton-Dyer conjecture using only the axioms of mathematics. This is not about extracting the beauty from a proof and leaving only the directed acyclic graph. This is about developing computer tools which have the potential to help researchers and PhD students in new ways. Much remains to be done. I will give an overview of the area.

Chair: Martin Hairer (United Kingdom)
Plenary Speaker: Kevin Buzzard (United Kingdom)

A fast growing corpus of formalized mathematics





[A computer-checked proof of the four color theorem, G. Gonthier 2003]

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

[A formal proof of the Kepler conjecture Hales et al., Cambridge University Press, 2017]

[Formalising the h-Principle and Sphere Eversion, F. van Doorn, P. Massot, O. Nash, Procs. of CPP 2023]

An instrument for mathematics in the making

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."

Ready for monumental endeavors

FLT

Ongoing Lean formalisation of the proof of Fermat's Last Theorem

Readme | Versions (15) | Dependencies (10)

Fermat's Last Theorem

Website ready | Documentation passing | Blueprint WIP | Paper WIP | Gitpod ready-to-code

An ongoing multi-author open source project to formalise a proof of Fermat's Last Theorem in the Lean theorem prover.

Information about the project

The project is currently being led by Kevin Buzzard. Until September 2029 it is being funded by grant EP/Y022904/1, awarded by the EPSRC. The project is hosted at Imperial College London. Kevin would like to extend many many thanks to both of these institutions for their ongoing support of this nonstandard research.

General information ("What is Fermat's Last Theorem/Lean?" / "Why are you doing this?" etc) is here.

The route we will be taking was planned out essentially entirely by Richard Taylor in discussions with Buzzard. It is a modern variant of the original Wiles/Taylor-Wiles proof. For more details about the mathematics behind the proof, a good place to start is the blueprint.

Information on how to contribute is available here.

Apache 2.0
10 hours ago
460 stars

Lean

- **v4.20.0-rc5**
- v4.19.0-rc3
- v4.18.0
- v4.18.0-rc1
- v4.17.0-rc1
- v4.16.0-rc2
- v4.16.0-rc1
- v4.15.0-rc1
- v4.14.0
- v4.14.0-rc3

30 more

Homepage

imperialcollegelondon.gith...

Repository

Backstage

A page in analytic number theory

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),

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

where we compute the last integral numerically.[4]

[Major arcs for Goldbach's problem. H. Helfgott. arxiv 2013]

Iwan Quemerais' internship (2022): Verify a Legendre quadrature algorithm

- Fast and rigorous arbitrary-precision computation of Gauss-Legendre quadrature nodes and weights

  F. Johansson and M. Mezzarobba. SIAM J. Sci. Comput. (2018)

- Accurate multiple-precision Gauss-Legendre quadrature

  L. Fousse. IEEE Symp. on Computer Arithmetic (2007)

- Ball arithmetic (cf F. Johansson's Arb)

Ball arithmetic, formalized

Objective: complement Coq/Rocq's Interval library with a center-radius variant:

$$\mathbb{B}(m, r) \triangleq \{\ x \in \mathbb{R}^n \mid ||m - x|| \leq r\ \},\ \text{for}\ m \in \mathbb{R}^n, r \in \overline{\mathbb{R}^+}$$

Ball arithmetic, formalized

Objective: complement Coq/Rocq's Interval library with a center-radius variant:

$$\mathbb{B}(m, r) \triangleq \{\ x \in \mathbb{R}^n \mid ||m - x|| \leq r\ \},\ \text{for}\ m \in \mathbb{R}^n, r \in \overline{\mathbb{R}^+}$$

Ingredients:

- Different types of numbers

  $$\mathbb{F} \subset \mathbb{R}_\perp \subset \overline{\mathbb{R}_\perp} \text{ and } \mathbb{F} \subset \overline{\mathbb{R}_\perp}^+ \subset \overline{\mathbb{R}_\perp}$$

- Different types of balls

  for $m, r \in \mathbb{F}$, for $m \in \mathbb{R}$ and $r \in \overline{\mathbb{R}^+}$, NaBs,...

What brave interns end up spending their implementation efforts on

```
Lemma add_Fball_Rball : forall p b1 b2, subset (add_Rball (convert b1) (convert b2)) (convert (Badd p b1 b2)).
Proof.
  move=> p.
  case=> [| m1 r1] //.
  case=> [| m2 r2] //.
  - by case (add_Rball (convert (Bmr m1 r1))).
  - unfold subset => /=.
    move: (add_NR_correct p m1 m2).
    case: (F.toX m1) => [H1| m1'] /=.
    + by rewrite H1.
    + case: (F.toX m2) => [H2 | m2' H] /=.
      * by rewrite H2.
      * destruct H.
        move: H. rewrite F.real_correct.
        move: H0.
        case: ( F.toX (add_NR p m1 m2)) => //.
        move=> r H _.
        move: (add_UP_correct p r1 r2).
        case: (F.toX r1) .
        ++ move=> H2 /=.
           move: (add_UP_correct p
                    (add_UP p r1 r2)
                    (Delta p (add_NR p m1 m2))).
           rewrite H2 => H3.
           by rewrite H3.
        ++ move=> r1'.
           case: (F.toX r2).
           ** move=> H2 /=.
              move: (add_UP_correct p
                       (add_UP p r1 r2)
                       (Delta p (add_NR p m1 m2))).
              rewrite H2 => H3.
              by rewrite H3.
           ** move=> r2'/=.
              move: H => /= H1 H2.
              move: (add_UP_correct p
                       (add_UP p r1 r2)
                       (Delta p (add_NR p m1 m2))).
              move: H2.
              case: (F.toX (add_UP p r1 r2)).
              +++ move=> H2 H3.
                  rewrite H3 => //.
              +++ move=> a H2.
                  move: H1.
                  case: (F.toX (Delta p (add_NR p m1 m2))).
                  *** move=> H1 H3.
                      rewrite H3 => //.
                  *** move=> b H1.
                      case: (F.toX (add_UP p
                                      (add_UP p r1 r2)
                                      (Delta p (add_NR p m1 m2)))) => //.
                      move => r_add.
                      move: H1 H2.
                      unfold le_upper => /=.
                      move=> H1 H2 H3.
                      have H4 : -b <= r- (m1' + m2') <= b. exact: Stdlib.Rabs_def2_le.
                      apply Rle_minus_r.
                      apply Rabs_le.
```

Proving this specification of the ball addition takes 58l.

A pervasive issue in libraries of formalized mathematics

A few examples among many:

- A Formal Disproof of Hirsch Conjecture
  X. Allamigeon, Q. Canu and P.-Y. Strub. Proc. of CPP. (2023)
- Extended reals in Lean's mathlib library
- Sandrine Blazy's interview for ETAPS 2024

Natural numbers, in unary and binary representation

Natural numbers in unary representation

```
(* Unary representation for natural numbers *)
Inductive nat : Type := O : nat | S : nat -> nat.

Check nat_ind : ∀P : nat -> Type,
   P 0 -> (forall n : nat, P n -> P (S n)) -> ∀n : nat, P n

Definition (_ + _) : nat -> nat -> nat := fun x, y : nat =>
 match x, y with
 |O, m => m (* 0 + m = m *)
 |S n, m => S (n + m) (* (S n) + m = S (n + m) *)
 end.

Fact test_add : 17 + 3 = 20. Proof. by compute. Qed.

Definition prime x : nat -> bool := ...

Fact test_prime : prime 17 = true. Proof. by compute. Qed.

(* too long *)
Fact test_prime_larger : prime 29986577 = true. Proof. by compute. Qed.
```

Natural numbers, in binary representation

```
Inductive positive : Type :=
  | xI : positive → positive (* p1 *)
  | xO : positive → positive (* p0 *)
  | xH : positive.           (* 1  *)
```

```
Inductive N : Type :=
| O_N : N
| Npos : positive → N.
```

Natural numbers, in binary representation

```
Inductive positive : Type :=
  | xI : positive → positive (* p1 *)
  | xO : positive → positive (* p0 *)
  | xH : positive.           (* 1  *)
```

```
Inductive N : Type :=
| O_N : N
| Npos : positive → N.
```

```
Check N_ind : ∀P : N -> Prop, P O_N -> (∀ p : positive, P (Npos p)) -> ∀ n : N, P n
```

Natural numbers, in binary representation

```
Inductive positive : Type :=
  | xI : positive → positive  (* p1 *)
  | xO : positive → positive  (* p0 *)
  | xH : positive.            (* 1  *)
```

```
Inductive N : Type :=
| O_N : N
| Npos : positive → N.
```

```
Check N_ind : ∀P : N -> Prop, P O_N -> (∀ p : positive, P (Npos p)) -> ∀ n : N, P n
```

```
Definition S_pos (p : positive) : positive :=
   match p with | xH ⇒ xO xH | xO p ⇒ xI p | xI p ⇒ xO (S_pos p) end.
```

```
Definition S_N (n : N) := match n with Npos p ⇒ Npos (S_pos p) | _ ⇒ Npos xH end.
```

Natural numbers, in binary representation

```
Inductive positive : Type :=
  | xI : positive → positive  (* p1 *)
  | xO : positive → positive  (* p0 *)
  | xH : positive.            (* 1  *)
```

```
Inductive N : Type :=
| O_N : N
| Npos : positive → N.
```

```
Check N_ind : ∀P : N -> Prop, P O_N -> (∀ p : positive, P (Npos p)) -> ∀ n : N, P n
```

```
Definition S_pos (p : positive) : positive :=
   match p with | xH ⇒ xO xH | xO p ⇒ xI p | xI p ⇒ xO (S_pos p) end.
```

```
Definition S_N (n : N) := match n with Npos p ⇒ Npos (S_pos p) | _ ⇒ Npos xH end.
```

```
Definition (_ +_N _) : N -> N -> N := fun x, y : N => ...
```

Relating representations

Casts $\uparrow_{\mathbb{N}} : \mathtt{N} \to \mathbb{N}$ and $\downarrow_{\mathbb{N}} : \mathbb{N} \to \mathtt{N}$:

- Are inverse of each other:

$$\forall x : \mathbb{N}, \quad \uparrow_{\mathbb{N}} (\downarrow_{\mathbb{N}} x) = x \quad \text{and} \quad \forall x : \mathtt{N}, \quad \downarrow_{\mathbb{N}} (\uparrow_{\mathbb{N}} x) = x$$

Relating representations

Casts $\uparrow_{\mathbb{N}} : \mathtt{N} \to \mathbb{N}$ and $\downarrow_{\mathbb{N}} : \mathbb{N} \to \mathtt{N}$:

- Are inverse of each other:

$$\forall x : \mathbb{N}, \quad \uparrow_{\mathbb{N}} (\downarrow_{\mathbb{N}} x) = x \quad \text{and} \quad \forall x : \mathtt{N}, \quad \downarrow_{\mathbb{N}} (\uparrow_{\mathbb{N}} x) = x$$

- Relate zeros and successors:

$$\forall n : \mathbb{N}, \quad \downarrow_{\mathbb{N}} (\mathtt{S}_{\mathbb{N}}\ n) = \mathtt{S}_{\mathtt{N}}\ (\downarrow_{\mathbb{N}}\ n) \quad \text{and} \downarrow_{\mathbb{N}} 0_{\mathbb{N}} = 0_{\mathtt{N}}$$

- Relate additions:

$$\forall n, m : \mathbb{N}, \quad \downarrow_{\mathbb{N}} (n +_{\mathbb{N}} m) = (\downarrow_{\mathbb{N}} n) +_{\mathtt{N}} (\downarrow_{\mathbb{N}} m)$$

- ...

Relating representations

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} =$$

Relating representations

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

Relating representations

This should be enough for:

- Refinements:

$$
\begin{aligned}
2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = & \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}}) \\
= & \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (S_{\mathbb{N}}\ (S_{\mathbb{N}}\ 0_{\mathbb{N}}) +_{\mathbb{N}} (S_{\mathbb{N}}\ 0_{\mathbb{N}}))
\end{aligned}
$$

Relating representations

This should be enough for:

- Refinements:

$$
\begin{aligned}
2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = {} & \uparrow_{\mathbb{N}} \downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}}) \\
= {} & \uparrow_{\mathbb{N}} (\downarrow_{\mathbb{N}} (S_{\mathbb{N}}\ (S_{\mathbb{N}}\ 0_{\mathbb{N}})) +_{\mathbb{N}} \downarrow_{\mathbb{N}} (S_{\mathbb{N}}\ 0_{\mathbb{N}}))
\end{aligned}
$$

Relating representations

This should be enough for:

- Refinements:

$$
\begin{aligned}
2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = {} & \uparrow_{\mathbb{N}} \downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}}) \\
= {} & \uparrow_{\mathbb{N}} ((S_{\mathbb{N}} \downarrow_{\mathbb{N}} (S_{\mathbb{N}}\ 0_{\mathbb{N}})) +_{\mathbb{N}} (S_{\mathbb{N}} \downarrow_{\mathbb{N}} 0_{\mathbb{N}}))
\end{aligned}
$$

Relating representations

This should be enough for:

- Refinements:

$$
\begin{aligned}
2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = & \uparrow_{\mathbb{N}} \downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}}) \\
& = \uparrow_{\mathbb{N}} ((S_{\mathbb{N}} \ (S_{\mathbb{N}} \downarrow_{\mathbb{N}} 0_{\mathbb{N}})) +_{\mathbb{N}} (S_{\mathbb{N}} \ 0_{\mathbb{N}}))
\end{aligned}
$$

Relating representations

This should be enough for:

- Refinements:

$$
\begin{aligned}
2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = & \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}}) \\
& = \uparrow_{\mathbb{N}} ((S_{\mathbb{N}}\ (S_{\mathbb{N}}\ 0_{\mathbb{N}})) +_{\mathbb{N}} (S_{\mathbb{N}}\ 0_{\mathbb{N}}))
\end{aligned}
$$

Relating representations

This should be enough for:

- Refinements:

$$\begin{aligned} 2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = & \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}}) \\ = & \uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}}) \end{aligned}$$

Relating representations

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements:

$$\sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

Relating representations

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements:

$$\sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities:

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

Relating representations

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements:

$$\sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities:

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind :∀ P : N → Prop,
  P O_N → (∀ n : N, P n → P (S_N n)) →∀ n : N, P n.
```

Relating representations

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements:

$$\sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities:

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind :∀ P : N → Prop,
  P O_N → (∀ n : N, P n → P (S_N n)) →∀ n : N, P n.
```

- Generalization, e.g. using $\mathbb{N} \subset \mathbb{Z}$

Relating representations

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements:

$$\sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities:

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind :∀ P : N → Prop,
  P O_N → (∀ n : N, P n → P (S_N n)) →∀ n : N, P n.
```

- Generalization, e.g. using $\mathbb{N} \subset \mathbb{Z}$

Relating representations

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}}\downarrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements:

$$\sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities:

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind :∀ P : N → Prop,
  P O_N → (∀ n : N, P n → P (S_N n)) →∀ n : N, P n.
```

- Generalization, e.g. using $\mathbb{N} \subset \mathbb{Z}$

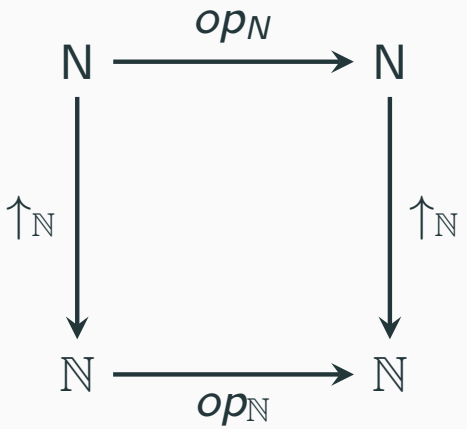
$\Rightarrow$ Can we automate these proofs entirely?

Parametricity

Related output from related input

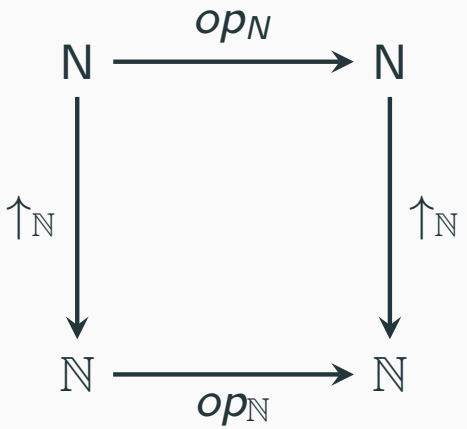
$$\forall x, \uparrow_{\mathbb{N}} (-x) = - \uparrow_{\mathbb{N}} (x)$$

Related output from related input

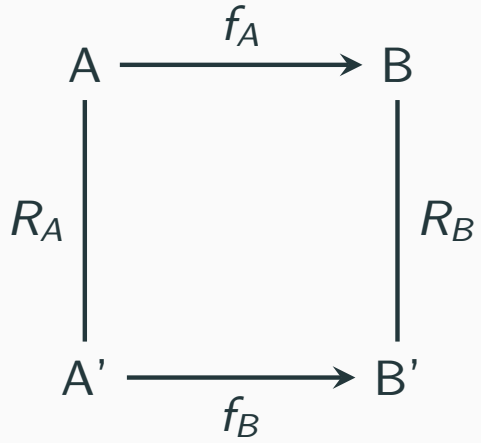


$$\forall x, \uparrow_{\mathbb{N}} (op_N\ x) = op_{\mathbb{N}}\ (\uparrow_{\mathbb{N}} (x))$$

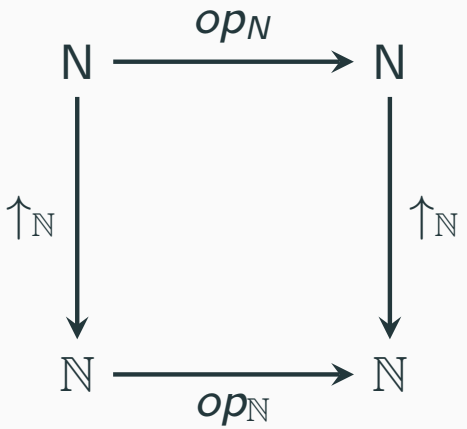
Related output from related input



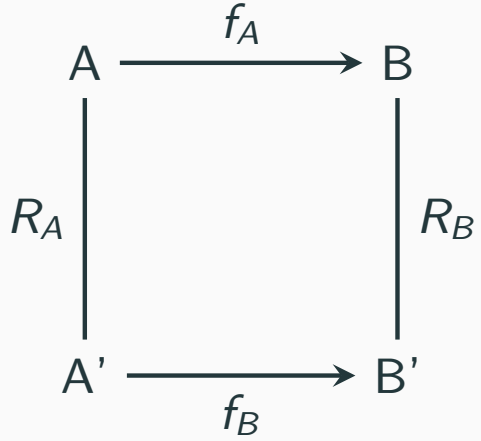
$$\forall x, \uparrow_{\mathbb{N}} (op_N\ x) = op_{\mathbb{N}}\ (\uparrow_{\mathbb{N}} (x))$$



Related output from related input



$$\forall x, \uparrow_{\mathbb{N}} (op_N\ x) = op_{\mathbb{N}}\ (\uparrow_{\mathbb{N}} (x))$$



$$\forall x, y, R\ x\ y \Rightarrow R\ (op_N\ x)\ (op_{\mathbb{N}}\ y) \quad \text{for} \quad R\ u\ v \triangleq u = \uparrow_{\mathbb{N}} v$$

Relational interpretation of (polymorphic) types

Any type T is associated a (here binary) relation $[T]$ on terms defined by:

- $[A \to B] \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in [A] \Rightarrow (f_1\ a, f_2\ a') \in [B]\}$
- $[\forall X, F\ X] \triangleq \{(f_1, f_2) \mid$

Relational interpretation of (polymorphic) types

Any type T is associated a (here binary) relation $[T]$ on terms defined by:

- $[A \to B] \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in [A] \Rightarrow (f_1\ a, f_2\ a') \in [B]\}$
- $[\forall X, F\ X] \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R},$

Relational interpretation of (polymorphic) types

Any type T is associated a (here binary) relation $[T]$ on terms defined by:

- $[A \to B] \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in [A] \Rightarrow (f_1\ a, f_2\ a') \in [B]\}$
- $[\forall X, F\ X] \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2\ A') \in F\ \mathcal{R}\}$

Relational interpretation of (polymorphic) types

Any type T is associated a (here binary) relation $[T]$ on terms defined by:

- $[A \to B] \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in [A] \Rightarrow (f_1\ a, f_2\ a') \in [B]\}$
- $[\forall X, F\ X] \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2\ A') \in F\ \mathcal{R}\}$

with $F\mathcal{R}$ defined by:

- $\mathcal{R}_1 \to \mathcal{R}_2 \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in \mathcal{R}_1 \Rightarrow (f_1\ a, f_2\ a') \in \mathcal{R}_2\}$
- $\forall \mathcal{R}, F\ \mathcal{R} \triangleq \{(f_1, f_2) \mid$

Relational interpretation of (polymorphic) types

Any type T is associated a (here binary) relation $[T]$ on terms defined by:

- $[A \to B] \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in [A] \Rightarrow (f_1\ a, f_2\ a') \in [B]\}$
- $[\forall X, F\ X] \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2\ A') \in F\ \mathcal{R}\}$

with $F\mathcal{R}$ defined by:

- $\mathcal{R}_1 \to \mathcal{R}_2 \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in \mathcal{R}_1 \Rightarrow (f_1\ a, f_2\ a') \in \mathcal{R}_2\}$
- $\forall \mathcal{R}, F\ \mathcal{R} \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R},$

Relational interpretation of (polymorphic) types

Any type T is associated a (here binary) relation $[T]$ on terms defined by:

- $[A \to B] \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in [A] \Rightarrow (f_1\ a, f_2\ a') \in [B]\}$
- $[\forall X, F\ X] \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2\ A') \in F\ \mathcal{R}\}$

with $F\mathcal{R}$ defined by:

- $\mathcal{R}_1 \to \mathcal{R}_2 \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in \mathcal{R}_1 \Rightarrow (f_1\ a, f_2\ a') \in \mathcal{R}_2\}$
- $\forall \mathcal{R}, F\ \mathcal{R} \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2 A') \in F\ \mathcal{R}\}$

Relational interpretation of (polymorphic) types

Any type T is associated a (here binary) relation $[T]$ on terms defined by:

- $[A \to B] \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in [A] \Rightarrow (f_1\ a, f_2\ a') \in [B]\}$
- $[\forall X, F\ X] \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2\ A') \in F\ \mathcal{R}\}$

with $F\mathcal{R}$ defined by:

- $\mathcal{R}_1 \to \mathcal{R}_2 \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in \mathcal{R}_1 \Rightarrow (f_1\ a, f_2\ a') \in \mathcal{R}_2\}$
- $\forall \mathcal{R}, F\ \mathcal{R} \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2 A') \in F\ \mathcal{R}\}$

Extensions:

- $\mathcal{R}_1 \times \mathcal{R}_2$ relates pairs of related elements
- *list* $\mathcal{R}$ relates lists of same length, with pairwise related elements
- ...

Parametricity and theorems for free

Theorem (Abstraction theorem for system F)

If $\vdash t \ : T$ *in System F, then* $[T]\ t\ t$

Parametricity and theorems for free

Theorem (Abstraction theorem for system F)

If $\vdash t : T$ *in System F, then* $[T]\ t\ t$

Also related to realizability, logical relations, full abstraction ...

C. Strachey, G. Plotkin, R. Statman, J. Reynolds, J.-Y. Girard,...

Parametricity and theorems for free

Theorem (Abstraction theorem for system F)

If $\vdash t : T$ *in System F, then* $[T]\ t\ t$

Also related to realizability, logical relations, full abstraction . . .

C. Strachey, G. Plotkin, R. Statman, J. Reynolds, J.-Y. Girard,. . .

Used to:

- Study definability in type systems
- Prove strong normalization results
- Obtain theorems for free

Parametricity and theorems for free

Theorem (Abstraction theorem for system F)

If $\vdash t : T$ *in System F, then* $[T]\ t\ t$

Also related to realizability, logical relations, full abstraction ...

C. Strachey, G. Plotkin, R. Statman, J. Reynolds, J.-Y. Girard,...

Used to:

- Study definability in type systems
- Prove strong normalization results
- Obtain theorems for free

(after the title of a paper by P. Wadler)

Parametricity and theorems for free

Theorem (Abstraction theorem for system F)

If $\vdash t : T$ *in System F, then* $[T]\ t\ t$

Also related to realizability, logical relations, full abstraction ...

C. Strachey, G. Plotkin, R. Statman, J. Reynolds, J.-Y. Girard,...

Used to:

- Study definability in type systems
- Prove strong normalization results
- Obtain theorems for free

(after the title of a paper by P. Wadler)

Example of free theorem: $\sum_{i=0_{\mathbb{N}}}^{17} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$

Example of free theorem

Prove than $\sum_{i=\mathbf{0}_{\mathbb{N}}}^{\mathbf{17}} i \;= \uparrow_{\mathbb{N}} \sum_{i=\mathbf{0}_{\mathbb{N}}}^{\mathbf{17}_{\mathbb{N}}} i$

Example of free theorem

Prove than $\sum_{i=0_{\mathbb{N}}}^{17} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathtt{N}}}^{17_{\mathtt{N}}} i$

By relating:

- fold $\mathbb{N}\ +_{\mathbb{N}}\ 0_{\mathbb{N}}\ [0_{\mathbb{N}}, \ldots 17_{\mathbb{N}}]$
- fold $\mathtt{N}\ +_{\mathtt{N}}\ 0_{\mathtt{N}}\ [0_{\mathtt{N}}, \ldots 17_{\mathtt{N}}]$

Example of free theorem

Prove than $\sum_{i=0_{\mathbb{N}}}^{17} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathtt{N}}}^{17_{\mathtt{N}}} i$

By relating:

- fold $\mathbb{N}\ +_{\mathbb{N}}\ 0_{\mathbb{N}}\ [0_{\mathbb{N}}, \ldots 17_{\mathbb{N}}]$
- fold $\mathtt{N}\ +_{\mathtt{N}}\ 0_{\mathtt{N}}\ [0_{\mathtt{N}}, \ldots 17_{\mathtt{N}}]$

Via:

- Relation $\mathcal{R} \triangleq \{(x, y) \mid y = \uparrow_{\mathbb{N}} x\}$

Example of free theorem

Prove than $\sum_{i=0_{\mathbb{N}}}^{17} i \;= \uparrow_{\mathbb{N}} \sum_{i=0_{\mathtt{N}}}^{17_{\mathtt{N}}} i$

By relating:

- fold $\mathbb{N}\ +_{\mathbb{N}}\ 0_{\mathbb{N}}\ [0_{\mathbb{N}}, \ldots 17_{\mathbb{N}}]$
- fold $\mathtt{N}\ +_{\mathtt{N}}\ 0_{\mathtt{N}}\ [0_{\mathtt{N}}, \ldots 17_{\mathtt{N}}]$

Via:

- Relation $\mathcal{R} \triangleq \{(x, y) \mid y = \uparrow_{\mathbb{N}} x\}$

Using:

- $[A \to B] \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in [A] \Rightarrow (f_1\ a, f_2\ a') \in [B]\}$
- $[\forall X, F\ X] \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2\ A') \in F\ \mathcal{R}\}$

with:

- $\mathcal{R}_1 \to \mathcal{R}_2 \triangleq \{(f_1, f_2) \mid \forall a\, a', (a, a') \in \mathcal{R}_1 \Rightarrow (f_1\ a, f_2\ a') \in \mathcal{R}_2\}$
- $\forall \mathcal{R}, F\ \mathcal{R} \triangleq \{(f_1, f_2) \mid \forall A\, A', \forall \mathcal{R}, (f_1\ A, f_2\ A') \in F\ \mathcal{R}\}$
- *list* $\mathcal{R}$ relates lists of same length, with pairwise related elements

A binary relation on type A and B is a term of type $A \to B \to \mathcal{U}$.

Pametricity translations for dependent type theory

A binary relation on type A and B is a term of type $A \to B \to \mathcal{U}$.

- Term translation:

$$
\begin{aligned}
[\![\mathcal{U}]\!] &\triangleq \lambda A\, B.\, A \to B \to \mathcal{U} \\
[\![x]\!] &\triangleq x_R \\
[\![A\ B]\!] &\triangleq [\![A]\!]\ B\ B'\ [\![B]\!] \\
[\![\lambda x : A.\, t]\!] &\triangleq \lambda (x : A)(x' : A')(x_R : [\![A]\!]\ x\ x').\, [\![t]\!] \\
[\![\forall x : A.\, B]\!] &\triangleq \lambda f\ f'.\, \forall (x : A)(x' : A')(x_R : [\![A]\!]\ x\ x').\, [\![B]\!]\ (f\ x)\ (f'\ x')
\end{aligned}
$$

A binary relation on type A and B is a term of type $A \to B \to \mathcal{U}$.

- Term translation:

$$
\begin{aligned}
[\![\mathcal{U}]\!] &\triangleq \lambda A\, B.\, A \to B \to \mathcal{U} \\
[\![x]\!] &\triangleq x_R \\
[\![A\ B]\!] &\triangleq [\![A]\!]\ B\ B'\ [\![B]\!] \\
[\![\lambda x : A.\, t]\!] &\triangleq \lambda (x : A)(x' : A')(x_R : [\![A]\!]\ x\ x').\, [\![t]\!] \\
[\![\forall x : A.\, B]\!] &\triangleq \lambda f\ f'.\, \forall (x : A)(x' : A')(x_R : [\![A]\!]\ x\ x').\, [\![B]\!]\ (f\ x)\ (f'\ x')
\end{aligned}
$$

Theorem (Abstraction theorem)

If $\vdash t : T$ *then* $\vdash [\![t]\!] : [\![T]\!]\ t\ t$.

J.-P. Bernardy, M. Lasson, ...

coq-community/paramcoq

Pametricity translations for dependent type theory

A binary relation on type A and B is a term of type $A \to B \to \mathcal{U}$.

- Term translation:

$$\begin{aligned}
[\![\mathcal{U}]\!] &\triangleq \lambda A\, B.\, A \to B \to \mathcal{U} \\
[\![x]\!] &\triangleq x_R \\
[\![A\ B]\!] &\triangleq [\![A]\!]\ B\ B'\ [\![B]\!] \\
[\![\lambda x : A.\, t]\!] &\triangleq \lambda (x : A)(x' : A')(x_R : [\![A]\!]\ x\ x').\, [\![t]\!] \\
[\![\forall x : A.\, B]\!] &\triangleq \lambda f\ f'.\, \forall (x : A)(x' : A')(x_R : [\![A]\!]\ x\ x').\, [\![B]\!]\ (f\ x)\ (f'\ x')
\end{aligned}$$

- Context translation:

$$\begin{aligned}
[\![\langle\rangle]\!] &\triangleq \langle\rangle \\
[\![\Gamma, x : A]\!] &\triangleq [\![\Gamma]\!], x : A, x' : A', x_R : [\![A]\!]\ x\ x'
\end{aligned}$$

Theorem (Abstraction theorem)

If $\Gamma \vdash t : T$ *then* $[\![\Gamma]\!] \vdash t : T$, $[\![\Gamma]\!] \vdash t' : T'$ *and* $[\![\Gamma]\!] \vdash [\![t]\!] : [\![T]\!]\ t\ t'$.

J.-P. Bernardy, M. Lasson, ...

coq-community/paramcoq

Formal proofs for free?

This should be enough for:

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} =\uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements:

$$\sum_{i=0_{\mathbb{N}}}^{17} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identites:

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind :∀ P : N → Prop,
    P 0ₙ → (∀ n : N, P n → P (Sₙ n)) →∀ n : N, P n.
```

- Implications, e.g, using $\mathbb{N} \subset \mathbb{Z}$

Formal proofs for free?

This should be enough for:

- Refinements: Yes

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements: Yes

$$\sum_{i=0_{\mathbb{N}}}^{17} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identites:

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind :∀ P : N → Prop,
    P 0ₙ → (∀ n : N, P n → P (Sₙ n)) →∀ n : N, P n.
```

- Implications, e.g, using $\mathbb{N} \subset \mathbb{Z}$

Formal proofs for free?

This should be enough for:

- Refinements: Yes

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} =\uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements: Yes

$$\sum_{i=0_{\mathbb{N}}}^{17} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identites: No

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles: No

```
Fact N_ind :∀ P : N → Prop,
    P 0ₙ → (∀ n : N, P n → P (Sₙ n)) →∀ n : N, P n.
```

- Implications, e.g, using $\mathbb{N} \subset \mathbb{Z}$ No

Richer relational interpretations for types

The key ingredient is to turn:

$$[\![\mathcal{U}]\!]\ A\ B \ \triangleq\ A \to B \to \mathcal{U}$$

Into:

$$[\![\mathcal{U}]\!]\ A\ B \ \triangleq\ \Sigma(R : A \to B \to \mathcal{U}), \ldots$$

Free theorems by univalent parametricity

$$[\![\mathcal{U}]\!]\ A\ B \triangleq \Sigma(R : A \to B \to \mathcal{U}), R \text{ is a type equivalence}$$

[The Marriage of Univalence and Parametricity - N. Tabareau, E. Tanter, M. Sozeau. J. ACM (2021)]

Free theorems by univalent parametricity

$$[\![\mathcal{U}]\!]\ A\ B \triangleq \Sigma(R : A \to B \to \mathcal{U}), R \text{ is a type equivalence}$$

[The Marriage of Univalence and Parametricity - N. Tabareau, E. Tanter, M. Sozeau. J. ACM (2021)]

- Refinements: Yes

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements: Yes

$$\sum_{i=0_{\mathbb{N}}}^{17} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities:

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind : ∀ P : N → Prop,
   P 0ₙ → (∀ n : N, P n → P (Sₙ n)) → ∀ n : N, P n.
```

- Implications, e.g. using $\mathbb{N} \subset \mathbb{Z}$

Free theorems by univalent parametricity

$$[\![\mathcal{U}]\!]\ A\ B \triangleq \Sigma(R : A \to B \to \mathcal{U}), R \text{ is a type equivalence}$$

[The Marriage of Univalence and Parametricity - N. Tabareau, E. Tanter, M. Sozeau. J. ACM (2021)]

- Refinements: Yes

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements: Yes

$$\sum_{i=0_{\mathbb{N}}}^{17} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities: Yes

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind : ∀ P : N → Prop,
   P O_N → (∀ n : N, P n → P (S_N n)) → ∀ n : N, P n.
```

- Implications, e.g. using $\mathbb{N} \subset \mathbb{Z}$

Free theorems by univalent parametricity

$$[\![\mathcal{U}]\!]\ A\ B \triangleq \Sigma(R : A \to B \to \mathcal{U}), R \text{ is a type equivalence}$$

[The Marriage of Univalence and Parametricity - N. Tabareau, E. Tanter, M. Sozeau. J. ACM (2021)]

- Refinements: Yes

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements: Yes

$$\sum_{i=0_{\mathbb{N}}}^{17} i = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities: Yes

```
Fact addN_assoc :∀ x y : N, x + (y + z) = (x + y) + z.
```

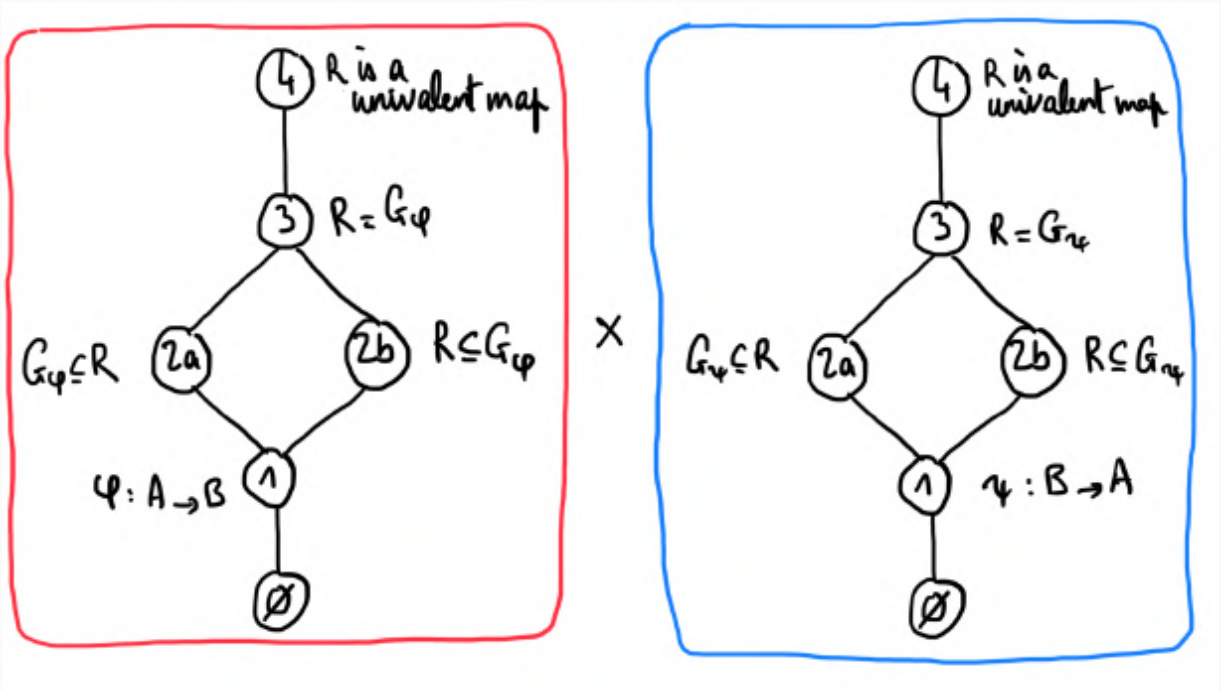
- Induction principles: Yes, but assuming the univalence axiom

```
Fact N_ind : ∀ P : N → Prop,
    P 0ₙ → (∀ n : N, P n → P (Sₙ n)) → ∀ n : N, P n.
```

- Implications, e.g. using $\mathbb{N} \subset \mathbb{Z}$ No

A hierarchy of structures on relations

Let $G_\phi \triangleq \{(x, y) \mid y = \phi\ x\}$ and define $[\![\mathcal{U}]\!]^{k,l}$ for $k, l \in \{0, 1, 2a, 2b, 3, 4\}$ as:



Examples:

- $[\![\mathcal{U}]\!]^{0,0}$: arbitrary relation
- $[\![\mathcal{U}]\!]^{4,4}$: type equivalence
- $[\![\mathcal{U}]\!]^{1,0}$: existence of a map
- $[\![\mathcal{U}]\!]^{3,3}$: type isomorphism
- $[\![\mathcal{U}]\!]^{3,2a}$: $\mathbb{N}$ and $\mathbb{Z}_n$
- $[\![\mathcal{U}]\!]^{3,2b}$: $\mathbb{R}_{\geq 0}$ and $\overline{\mathbb{R}}_{\geq 0}$

Generalized parametricity translations

Abstraction theorems:

- When restricting to $[\![\mathcal{U}]\!]^{0,0}$

  J.-P. Bernardy, M. Lasson et al. 2010 - 2013

- When restricting to $[\![\mathcal{U}]\!]^{4,4}$, under the univalence axiom

  [The Marriage of Univalence and Parametricity - N. Tabareau, E. Tanter, M. Sozeau. J. ACM (2021)]

Generalized parametricity translations

Abstraction theorems:

- When restricting to $[\![\mathcal{U}]\!]^{0,0}$

  J.-P. Bernardy, M. Lasson et al. 2010 - 2013

- When restricting to $[\![\mathcal{U}]\!]^{4,4}$, under the univalence axiom

  [The Marriage of Univalence and Parametricity - N. Tabareau, E. Tanter, M. Sozeau. J. ACM (2021)]

Trocq:

- More relations

  Abstraction theorems for $[\![\mathcal{U}]\!]^{k,l}$

- Stronger free theorems

  Non-systematic usage of the univalence axiom

- Implemented

  Translation and inference algorithm, programmed in Coq-Elpi

[Proof transfer for free, beyond univalence and equivalence, Cyril Cohen, Enzo Crance, A.M. - TOPLAS 2025]

Automating proof transfer

The `trocq` proof command is typically useful to prove a goal $G_1 \to G_2$, by:

- Trying to construct a relation at level $(1, 0)$ between the G_1 and G_2;
- Gradually computing sufficient conditions and synthesizing the proof;
- Finding base cases proofs of relations in a data-base populated by users.

Note: Lucie Lahaye recently proved the optimality of these conditions.

Free theorems by Trocq

- Refinements:

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} =\uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements:

$$\sum_{i=0_{\mathbb{N}}}^{17} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities:

```
Fact addN_assoc : ∀x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles:

```
Fact N_ind : ∀ P : N → Prop,
    P 0ₙ → (∀ n : N, P n → P (Sₙ n)) → ∀ n : N, P n.
```

- Implications, e.g using $\mathbb{N} \subset \mathbb{Z}$

Free theorems by Trocq

- Refinements: Yes

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} = \uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements: Yes

$$\sum_{i=0_{\mathbb{N}}}^{17} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities: Yes

```
Fact addN_assoc : ∀x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles: Yes, without assuming the univalence axiom

```
Fact N_ind : ∀ P : N → Prop,
    P O_N → (∀ n : N, P n → P (S_N n)) → ∀ n : N, P n.
```

- Implications, e.g using $\mathbb{N} \subset \mathbb{Z}$ Yes

Free theorems by Trocq

- Refinements: Yes

$$2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} =\uparrow_{\mathbb{N}} (2_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}})$$

- More refinements: Yes

$$\sum_{i=0_{\mathbb{N}}}^{17} i \; = \uparrow_{\mathbb{N}} \sum_{i=0_{\mathbb{N}}}^{17_{\mathbb{N}}} i$$

- Identities: Yes

```
Fact addN_assoc : ∀x y : N, x + (y + z) = (x + y) + z.
```

- Induction principles: Yes, without assuming the univalence axiom

```
Fact N_ind : ∀ P : N → Prop,
    P O_N → (∀ n : N, P n → P (S_N n)) → ∀ n : N, P n.
```

- Implications, e.g using $\mathbb{N} \subset \mathbb{Z}$ Yes

And more, e.g. setoid, generalized rewriting, coercions and cast insertion, etc.

Demo and conclusion

The Trocq plugin by examples: elementary number theory

- User input:
  - $(3, 2_a)$-relate $\mathbb{Z}_9$ and $\mathbb{Z}$, and their zero, multiplication, addition.

The Trocq plugin by examples: elementary number theory

- User input:
  - $(3, 2_a)$-relate $\mathbb{Z}_9$ and $\mathbb{Z}$, and their zero, multiplication, addition.
  - Also relate their 3-divisiblity :

$$3 \mid (x \bmod 9) \Rightarrow 3 \mid x.$$

- Transfer from $\mathbb{Z}_9$ to $\mathbb{Z}$:

```
Lemma flt3_step :
(forall (m n p : ℤ₉), m * n * p ≠ 0 [mod 3] -> (m^3 + n^3) ≠ p^3) ->
(forall (m n p : ℤ ), m * n * p ≠ 0 [mod 3] -> (m^3 + n^3) ≠ p^3).
Proof. by trocq=> /=. Qed.
```

"Simple things should be simple, complex things should be possible" A. Kay

- Trocq streamlines and extends several crucial automation devices.
- More fruits can surely be rept from the theory of programming languages.
- Formalizing mathematics should and will become easier and faster.

The Trocq plugin by example: natural numbers

- User input:

$$\forall x, \downarrow_{\mathbb{N}} (\uparrow_{\mathbb{N}} \ x) = x \qquad \downarrow_{\mathbb{N}} 0_{\mathbb{N}} = 0_{\mathtt{N}} \qquad \forall x, \downarrow_{\mathbb{N}} (S_{\mathbb{N}} \ x) = S_{\mathtt{N}} \ (\downarrow_{\mathbb{N}} x)$$

  $\Rightarrow$ (2a,3)-relate $\mathtt{N}$ and $\mathbb{N}$.

The Trocq plugin by example: natural numbers

- User input:

$$\forall x, \downarrow_{\mathbb{N}} (\uparrow_{\mathbb{N}} x) = x \qquad \downarrow_{\mathbb{N}} 0_{\mathbb{N}} = 0_{\mathbb{N}} \qquad \forall x, \downarrow_{\mathbb{N}} (S_{\mathbb{N}}\ x) = S_{\mathbb{N}}\ (\downarrow_{\mathbb{N}} x)$$

  $\Rightarrow$ (2a,3)-relate `N` and $\mathbb{N}$.

- Generation and axiom-free proof of the implication:

```
(∀ P : ℕ → Type, P 0ℕ → (∀ n : ℕ, P n → P (Sℕ n)) → ∀ n : ℕ, P n)
      →

∀ P : N → Type, P 0N → (∀ n : N, P n → P (SN n)) → ∀ n : N, P n
```

The Trocq plugin by examples: extended reals

- Consider types

```
Inductive R̄≥0 : Type := Fin : ℝ≥0 → R̄≥0 | Inf : R̄≥0.

Record summable := {to_seq : ℕ -> ℝ≥0; to_seqP : summable to_seq}
```

The Trocq plugin by examples: extended reals

- Consider types

```
Inductive R̄≥0 : Type := Fin : R≥0 → R̄≥0 | Inf : R̄≥0.

Record summable := {to_seq : ℕ -> R≥0; to_seqP : summable to_seq}
```

- User input:
  - $(4, 2_b)$-relate $\mathbb{R}_{\geq 0}$ and $\overline{\mathbb{R}}_{\geq 0}$,and their additions and sums
  - $(4, 2_b)$-relate `summable` and $\mathbb{N}$ -> $\overline{\mathbb{R}}_{\geq 0}$, and their additions and sums

The Trocq plugin by examples: extended reals

- Consider types

```
Inductive R̄≥0 : Type := Fin : R≥0 → R̄≥0 | Inf : R̄≥0.

Record summable := {to_seq : ℕ -> R≥0; to_seqP : summable to_seq}
```

- User input:
  - $(4, 2_b)$-relate $\mathbb{R}_{\geq 0}$ and $\overline{\mathbb{R}}_{\geq 0}$,and their additions and sums
  - $(4, 2_b)$-relate `summable` and $\mathbb{N}$ -> $\overline{\mathbb{R}}_{\geq 0}$, and their additions and sums
- Transfer from (summable sequences of positive) extended reals to reals:

```
Lemma Σ_R≥0_add : ∀u v : summable, Σ_R≥0 (u + v) = Σ_R≥0 u + Σ_R≥0 v.
Proof. trocq; exact Σ_R̄≥0_add. Qed.
```