Formalising large corner-free sets

A journey in type theory, effective topos, constructive logic and more

Gareth Ma

University of Warwick

May 18, 2024

Imagine...

You are Andrew Wiles.

You worked on a marvelous proof on a conjecture for 6 years, and finally publish it.

Two months later, a critical flaw was discovered, and your work is voided. 🙁

If only that computers can verify Mathematical proofs...

Formalise? What?

We can **formalise** proofs. Informally, it means to

Definition

Rewrite Mathematical proofs in a machine-understandable language.

The language I used is the `Lean 4` language + its Mathematics library `Mathlib 4`.

Formalise? What? 👀

Transitivity

Let P, Q, R be *logical* statements. If $P \implies Q$ and $Q \implies R$, then $P \implies R$.

Proof.

Suppose P holds. Then by $P \implies Q$, we know that Q holds. And since $Q \implies R$, we know that R holds. Hence, P implies R. $\square$

```
example {P Q R : Prop} (hPQ : P → Q) (hQR : Q → R) :
    P → R := by
  intro p
  have := hPQ p
  exact hQR this
```

Formalise what? (My project)

For my 3rd year project, I formalised a extremal combinatorics result in 2021 by Ben Green, under the supervision of Damiano Testa.

The resulting project is original work building on top of the `Lean 4 + Mathlib 4` libraries.

To my knowledge, this is the **best** result of this type formalised in any theorem prover.

1. Type Theory

2. Corner-free Sets

3. Asymptotics

4. Correctness via decidability instances

5. Conclusion

Flaws of set theory

In (naive) set theory, everything is a set. Numbers are encoded as nested sets, operations are set functions, etc.

There are many problems:

- $3 \in 17$ is a valid question.
- Russell's Paradox: $A \in A$ holds for some A.

Flaws of set theory

Slightly absurdly, the problem fundamentally stems from that **everything is a set**.

Idea

What if we separate objects into “elements” and “parents”?

For example, the numbers 3 and 17 will have the type $\mathbb{N}$ of natural numbers.

1. $3 \in 17$ question: An element cannot be an element of another element.
2. $A \in A$ paradox: There cannot be a Set that is an element of another Set.

Curry-Howard Correspondence

Function composition

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

Proof.

Let $p \in P$ be an element. Then, we can compute $f(p) \in Q$, and hence get $g(f(p)) \in R$, giving us the function $g \circ f : P \to R$. □

Curry-Howard Correspondence

Notice the similarity between this and the example with logical statements before!

Transitivity

Let P, Q, R be *logical* statements. If $P \implies Q$ and $Q \implies R$, then $P \implies R$.

Function composition

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

These two examples can be unified via ~~category~~ type theory.

Types

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

Curry-Howard Correspondence

Notice the similarity between this and the example with logical statements before!

Transitivity

Let P, Q, R be *logical* statements. If $P \implies Q$ and $Q \implies R$, then $P \implies R$.

Function composition

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

These two examples can be unified via ~~category~~ type theory.

Types

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

Curry-Howard Correspondence

Notice the similarity between this and the example with logical statements before!

Transitivity

Let P, Q, R be *logical* statements. If $P \implies Q$ and $Q \implies R$, then $P \implies R$.

Function composition

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

These two examples can be unified via ~~category~~ type theory.

Types

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

Curry-Howard Correspondence

Notice the similarity between this and the example with logical statements before!

Transitivity

Let P, Q, R be *logical* statements. If $P \implies Q$ and $Q \implies R$, then $P \implies R$.

Function composition

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

These two examples can be unified via ~~category~~ type theory.

Types

Let $P, Q, R \subseteq \mathbb{N}$ be sets of numbers. If $f : P \to Q$ and $g : Q \to R$ are two functions, then we can form a function $P \to R$.

Corner-free sets

A set $S \subseteq \mathbb{Z}^2$ is a corner-free set if for all $x, y, d \in \mathbb{Z}$,

$$\{(x, y), (x, y + d), (x + d, y)\} \subseteq \S \implies d = 0$$

Apart from corner-free sets, 3-AP-free sets are also commonly studied in extremal combinatorics. In my essay, I unified the approaches taken to construct the state-of-the-art lower bounds for both structures. For corner-free sets, an outline is given as follows:

1. Constructing an appropriate corner-free “two-dimensional” additive semiring $X = X_{r,q,d} \subseteq \mathbb{Z}_q^d \times \mathbb{Z}_q^d$ with special properties, parametrised by certain parameters r, q, d;
2. Use the naive embedding $\zeta : \mathbb{Z}_q^d \to \mathbb{Z}$ by parsing vectors as base-q digits of integers;
3. Prove that for $(\zeta(x), \zeta(y)), (\zeta(x'), \zeta(y)), (\zeta(x), \zeta(y')) \in \widetilde{\zeta}(X)$, $\zeta(x') + \zeta(y) = \zeta(x) + \zeta(y') \implies x' + y = x + y'$ (using the special properties of construction);
4. Conclude that $\widetilde{\zeta}(X)$ is also cornerfree;
5. Optimise parameters

Good luck to me. Live demo time!

Acknowledgement

I would like to thank my supervisor, Dr. Damiano Testa, for his tremendous support, encouragement, and the insightful discussions about the project during our near-weekly meetings. I would also like to thank Julian Berman, who is the creator of the `lean.nvim` plugin. Without the full power of my NeoVim setup, this project would have been drastically slowed down. Finally, I would like to thank the Mathlib community as a whole, and especially to Bhavik, David, Eric, Jireh, Karl, Kendall, Kevin, Mario, Thomas, Yaël and others (in alphabetical order), who helped with simplifying the proofs for various results and answered many of my questions.