Automated Theorem Proving

A humble computer's Math PhD dissertation

Anurudh Peduri

IIITH

26-09-2020

Talk Overview

Mathematics

Humans have been doing mathematics for centuries.

Mathematics

Humans have been doing mathematics for centuries.

We have made so many breakthroughs, invented so much mathematical machinery, and have used it in various fields.

Mathematics

Humans have been doing mathematics for centuries.

We have made so many breakthroughs, invented so much mathematical machinery, and have used it in various fields.

Now we have reached a stage with too much mathematical theory!

Verifying Mathematics

Researchers submit papers, and reviewers read and verify them.

Verifying Mathematics

Researchers submit papers, and reviewers read and verify them.

Humans are great at coming up with ideas, but are prone to errors.

Verifying Mathematics

Researchers submit papers, and reviewers read and verify them.

Humans are great at coming up with ideas, but are prone to errors.

Accurately verifying research level publications is a very difficult task.

Solution?

How do we overcome this?

Solution?

How do we overcome this? ...drumroll...

Solution?

How do we overcome this? ...drumroll... use computers!

Solution?

How do we overcome this? ...drumroll... use computers!

Computers are bad at coming up with ideas. But great at following instructions.

Solution?

How do we overcome this? ...drumroll... use computers!

Computers are bad at coming up with ideas. But great at following instructions.

If only we could instruct them on how to check math proofs...

Solution?

How do we overcome this? ...drumroll... use computers!

Computers are bad at coming up with ideas. But great at following instructions.

If only we could instruct them on how to check math proofs...

We can! Automated Theorem Provers (Proof Checkers to be accurate)

What are Theorem Provers?

Programs that can take in "math proofs" and verify them.

What are Theorem Provers?

Programs that can take in "math proofs" and verify them.

We also have Proof Assistants/Interactive Theorem Provers:
Programs that help us write proofs.

What are Theorem Provers?

Programs that can take in "math proofs" and verify them.

We also have Proof Assistants/Interactive Theorem Provers: Programs that help us write proofs.

A few popular Proof Assistants are `Coq`, `Lean`, `Isabelle`, `Agda`.

A few examples

Here are some (pseudo-)random theorems picked from the `Lean` library - `mathlib`

A few examples

```
theorem zmod.euler_criterion (p : ℕ) [fact (nat.prime p)] {a : zmod p} :
    a ≠ 0 → ((∃ (y : zmod p), y ^ 2 = a) ↔ a ^ (p / 2) = 1)
```

Euler's Criterion: a nonzero `a : zmod p` is a square if and only if `x ^ (p / 2) = 1`.

A few examples

```
theorem nat.exists_infinite_primes (n : ℕ) :
    ∃ (p : ℕ), n ≤ p ∧ nat.prime p
```

source

Euclid's theorem. There exist infinitely many prime numbers. Here given in the form: for every `n`, there exists a prime number `p ≥ n`.

A few examples

```
theorem nat.sum_four_squares (n : ℕ) :
  ∃ (a b c d : ℕ), a ^ 2 + b ^ 2 + c ^ 2 + d ^ 2 = n
```

A few examples

```
theorem abs_inner_le_norm {α : Type u} [inner_product_space α] (x y : α) :
    abs (has_inner.inner x y) ≤ ‖x‖ * ‖y‖
```

Cauchy–Schwarz inequality with norm

Teaching a computer math

Encoding math notation directly is difficult.

Teaching a computer math

Encoding math notation directly is difficult.

So, we invoke a very powerful result -

Curry-Howard Correspondence

Teaching a computer math

Encoding math notation directly is difficult.

So, we invoke a very powerful result -

Curry-Howard Correspondence

Equivalence between “Mathematical Proofs” and “Computer Programs”

Curry-Howard Correspondence

Propositions
P

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
P		P

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof		

p is a proof of P

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element

p is a proof of P $\qquad$ $p \in P$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element
Theorem/True		

p is a proof of P $\quad$ $p \in P$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element
Theorem/True	$\leftrightarrow$	Non-empty Set

---

p is a proof of P $\qquad$ $p \in P$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element
Theorem/True	$\leftrightarrow$	Non-empty Set
And		

$P \wedge Q$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element
Theorem/True	$\leftrightarrow$	Non-empty Set
And	$\leftrightarrow$	Cartesian Product

$P \wedge Q$ $\qquad$ $P \times Q$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element
Theorem/True	$\leftrightarrow$	Non-empty Set
And	$\leftrightarrow$	Cartesian Product
Or		

$P \vee Q$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element
Theorem/True	$\leftrightarrow$	Non-empty Set
And	$\leftrightarrow$	Cartesian Product
Or	$\leftrightarrow$	Disjoint Union

$P \vee Q$ $\qquad$ $P \sqcup Q$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element
Theorem/True	$\leftrightarrow$	Non-empty Set
And	$\leftrightarrow$	Cartesian Product
Or	$\leftrightarrow$	Disjoint Union
Implies		

$$P \implies Q$$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets
Proof	$\leftrightarrow$	Element
Theorem/True	$\leftrightarrow$	Non-empty Set
And	$\leftrightarrow$	Cartesian Product
Or	$\leftrightarrow$	Disjoint Union
Implies	$\leftrightarrow$	Function

$P \implies Q$ $\qquad$ $P \to Q$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets	$\leftrightarrow$	Types
Proof	$\leftrightarrow$	Element		
Theorem/True	$\leftrightarrow$	Non-empty Set		
And	$\leftrightarrow$	Cartesian Product		
Or	$\leftrightarrow$	Disjoint Union		
Implies	$\leftrightarrow$	Function		

P P $\mathtt{P}$

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets	$\leftrightarrow$	Types
Proof	$\leftrightarrow$	Element	$\leftrightarrow$	Value
Theorem/True	$\leftrightarrow$	Non-empty Set		
And	$\leftrightarrow$	Cartesian Product		
Or	$\leftrightarrow$	Disjoint Union		
Implies	$\leftrightarrow$	Function		

p is a proof of P $\quad$ $p \in P$ $\quad$ `P p;`

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets	$\leftrightarrow$	Types
Proof	$\leftrightarrow$	Element	$\leftrightarrow$	Value
Theorem/True	$\leftrightarrow$	Non-empty Set	$\leftrightarrow$	Inhabited Type
And	$\leftrightarrow$	Cartesian Product		
Or	$\leftrightarrow$	Disjoint Union		
Implies	$\leftrightarrow$	Function		

p is a proof of P $p \in P$ `P p;`

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets	$\leftrightarrow$	Types
Proof	$\leftrightarrow$	Element	$\leftrightarrow$	Value
Theorem/True	$\leftrightarrow$	Non-empty Set	$\leftrightarrow$	Inhabited Type
And	$\leftrightarrow$	Cartesian Product	$\leftrightarrow$	Pairs/Product Type
Or	$\leftrightarrow$	Disjoint Union		
Implies	$\leftrightarrow$	Function		

$P \wedge Q$

$P \times Q$

```
pair<P, Q>
// or
struct P_and_Q {
  P p;
  Q q;
};
```

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets	$\leftrightarrow$	Types
Proof	$\leftrightarrow$	Element	$\leftrightarrow$	Value
Theorem/True	$\leftrightarrow$	Non-empty Set	$\leftrightarrow$	Inhabited Type
And	$\leftrightarrow$	Cartesian Product	$\leftrightarrow$	Pairs/Product Type
Or	$\leftrightarrow$	Disjoint Union	$\leftrightarrow$	union/Sum Type
Implies	$\leftrightarrow$	Function		

$P \vee Q$

$P \sqcup Q$

```
union P_or_Q {
  P p;
  Q q;
}
```

Curry-Howard Correspondence

Propositions	$\leftrightarrow$	Sets	$\leftrightarrow$	Types
Proof	$\leftrightarrow$	Element	$\leftrightarrow$	Value
Theorem/True	$\leftrightarrow$	Non-empty Set	$\leftrightarrow$	Inhabited Type
And	$\leftrightarrow$	Cartesian Product	$\leftrightarrow$	Pairs/Product Type
Or	$\leftrightarrow$	Disjoint Union	$\leftrightarrow$	union/Sum Type
Implies	$\leftrightarrow$	Function	$\leftrightarrow$	function

$P \implies Q$ $\qquad$ $P \to Q$ $\qquad$ `Q f(P p);`

A few more equivalences

Propositions	$\leftrightarrow$	Types

A few more equivalences

Propositions	$\leftrightarrow$	Types
Predicates		

$P(x)$ where $x \in S$

A few more equivalences

Propositions	$\leftrightarrow$	Types
Predicates	$\leftrightarrow$	Function

$P(x)$ where $x \in S$ $\qquad$ $P : S \to Prop$

A few more equivalences

Propositions	$\leftrightarrow$	Types
Predicates	$\leftrightarrow$	Function
Exists		

$$\exists x \in S, P(x)$$

A few more equivalences

Propositions	$\leftrightarrow$	Types
Predicates	$\leftrightarrow$	Function
Exists	$\leftrightarrow$	Sum Dependent Type

$\exists x \in S, P(x)$ $\qquad$ $(a, p_a) \in \Sigma_{x:S} P(x)$

A few more equivalences

Propositions	$\leftrightarrow$	Types
Predicates	$\leftrightarrow$	Function
Exists	$\leftrightarrow$	Sum Dependent Type
Forall		

$$\forall x \in S, P(x)$$

A few more equivalences

Propositions	$\leftrightarrow$	Types
Predicates	$\leftrightarrow$	Function
Exists	$\leftrightarrow$	Sum Dependent Type
Forall	$\leftrightarrow$	Product Dependent Type

$\forall x \in S, P(x)$

$$f \in \Pi_{x:S} P(x)$$
$$f : (x : S) \to P(x)$$

Finally

Proof Checker $\rightarrow$

Finally

Proof Checker $\rightarrow$ Type Checker! (or compiler)

Finally

Proof Checker $\rightarrow$ Type Checker! (or compiler)
Writing Proofs $\rightarrow$

Finally

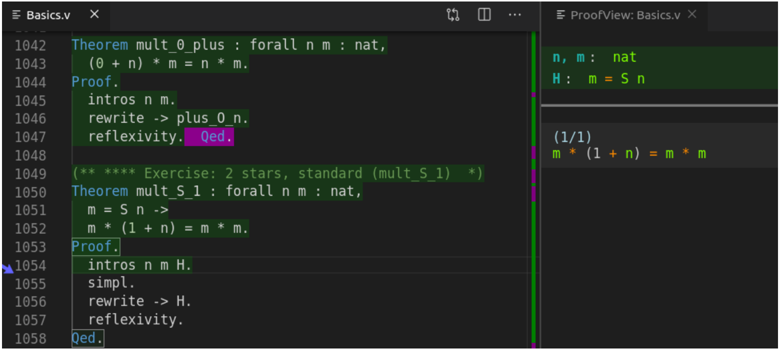
Proof Checker $\rightarrow$ Type Checker! (or compiler)
Writing Proofs $\rightarrow$ Writing programs to build values of a particular type

Finally

Proof Checker $\rightarrow$ Type Checker! (or compiler)
Writing Proofs $\rightarrow$ Writing programs to build values of a particular type
Interactive Proof Assistants: Assist you in these "constructions".

Interactive Theorem Provers

A sample proof in Coq



Getting Started

Numerous free resources available online!

- `Lean` Natural Number Game - by Kevin Buzzard: A gamified tutorial.
- Software Foundations: A collection of four textbooks on basics of theorem proving (using `Coq`).
- `Lean` Tutorial.

Thank You!