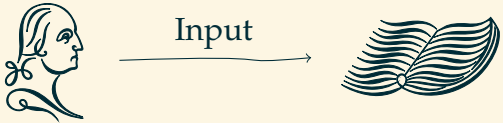
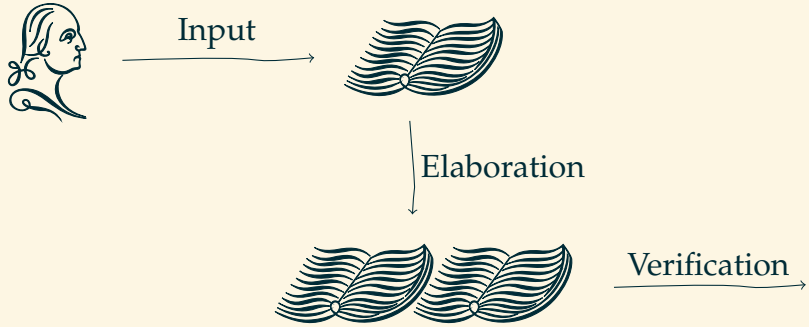
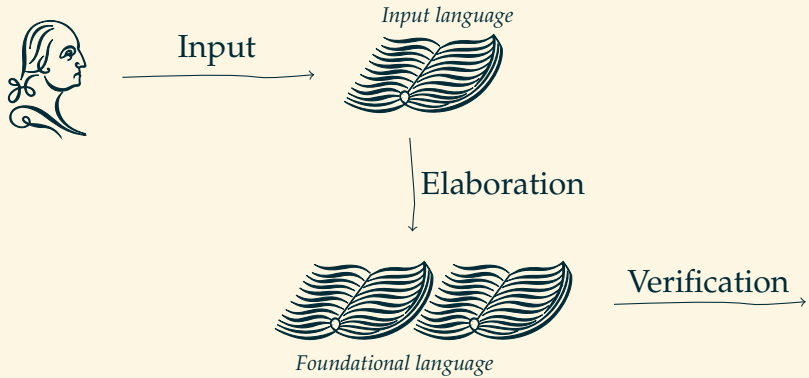
L∃∀N proof assistant


Input

Input

Verification


Input
Elaboration
Verification


Input
Input language
Elaboration
Verification
Foundational language

Foundations

Foundations

- I don’t care about foundations

Foundations

- I don't care about foundations
- Set theory $\iff$ Type theory

Foundations

- I don't care about foundations
- Set theory $\iff$ Type theory

The best of both worlds(?):

Input language:	type-theoretic
Foundation:	set-theoretic

Foundations

- I don't care about foundations
- Set theory $\iff$ Type theory

The best of both worlds(?):

Input language:	type-theoretic
Foundation:	set-theoretic

Example: TypeScript / JavaScript

Naive type theory

Naive type theory

Everything is a *term*, every term has a unique *type*

Naive type theory

Everything is a *term*, every term has a unique *type*

$$3 : \mathbb{N}, \quad \mathbb{N} : \mathrm{Type}_0, \quad \mathrm{Type}_0 : \mathrm{Type}_1, \quad \ldots$$

Naive type theory

Everything is a *term*, every term has a unique *type*

$$3 : \mathbb{N}, \quad \mathbb{N} : \mathrm{Type}_0, \quad \mathrm{Type}_0 : \mathrm{Type}_1, \quad \dots$$

If $f : X \to \mathrm{Type}$, then

$$\prod_x f(x), \qquad \sum_x f(x) = \coprod_x f(x)$$

are types.

Naive type theory

Everything is a *term*, every term has a unique *type*

$$3 : \mathbb{N}, \quad \mathbb{N} : \mathrm{Type}_0, \quad \mathrm{Type}_0 : \mathrm{Type}_1, \quad \dots$$

If $f : X \to \mathrm{Type}$, then

$$\prod_x f(x), \qquad \sum_x f(x) = \coprod_x f(x)$$

are types.

$$X \to Y = \prod_x Y$$

Inductive types

```
inductive Nat
| zero : Nat
| succ : Nat → Nat
```

Inductive types

```
inductive Nat
| zero : Nat
| succ : Nat → Nat
```

Automatically comes with some rules:

Inductive types

```
inductive Nat
| zero : Nat
| succ : Nat → Nat
```

Automatically comes with some rules:

- For all n, we have $0 \neq S(n)$

Inductive types

```
inductive Nat
| zero : Nat
| succ : Nat → Nat
```

Automatically comes with some rules:

- For all n, we have $0 \neq S(n)$
- Induction principle

Propositions as types

A proposition P is a type.

Propositions as types

A proposition P is a type.

A term $h : P$ is a proof.

Propositions as types

A proposition P is a type.

A term $h : P$ is a proof.

$$P \to Q \quad \text{“}=\text{”} \quad P \implies Q$$

Some components of elaboration

Some components of elaboration

- unification

Some components of elaboration

- unification
- type classes

Some components of elaboration

- unification
- type classes
- tactics

Unification

Example: $f : X \to Y$ is *surjective* if
for all y there exists an x such that $f(x) = y$

Unification

Example: $f : X \to Y$ is *surjective* if
for all y there exists an x such that $f(x) = y$

"Where does x live?"

Type class system

Every field is a ring is a group.

Type class system

Every field is a ring is a group.

Mathlib contains a graph ($\approx$ 550 nodes, $>$ 6000 edges)

Type class system

Every field is a ring is a group.

Mathlib contains a graph (≈ 550 nodes, > 6000 edges)

A lemma about topological groups can be applied to $\mathbb{R}$

Type class system

Every field is a ring is a group.

Mathlib contains a graph (≈ 550 nodes, > 6000 edges)

A lemma about topological groups can be applied to $\mathbb{R}$

because $\mathbb{R}$ is a normed field.

Example tactic: `simp`

Simp-lemmas have the form $a = b$ or $p \iff q$.

Example tactic: `simp`

Simp-lemmas have the form $a = b$ or $p \iff q$.

random example formula

Example tactic: `simp`

Simp-lemmas have the form $a = b$ or $p \iff q$.

random example formula

Example tactic: `simp`

Simp-lemmas have the form $a = b$ or $p \iff q$.

random example formula

↓ simp

rbndom exbmqle formulb

Example tactic: `simp`

Good simp-lemmas satisfy:

$$\text{complex LHS} = \text{simple RHS}$$

Example tactic: `simp`

Good simp-lemmas satisfy:

$$\text{complex LHS} = \text{simple RHS}$$

Examples:

$$1 \cdot x = x, \qquad a - a = 0, \qquad f(a+b) = f(a) + f(b)$$

$$x \in \{y\} \iff x = y, \qquad \text{proj}_1(x, y) = x$$

Example tactic: `simp`

Good simp-lemmas satisfy:

$$\text{complex LHS} = \text{simple RHS}$$

Examples:

$$1 \cdot x = x, \qquad a - a = 0, \qquad f(a+b) = f(a) + f(b)$$

$$x \in \{y\} \iff x = y, \qquad \text{proj}_1(x, y) = x$$

Bad examples:

$$p \wedge q \iff q \wedge p, \qquad f(x) = f(y) \iff x - y \in \ker(f)$$

Example tactic: `simp`

`simp` can prove

$$f(a+b) - f(b) \in \{\mathrm{proj}_1(a,b)\}$$

Example tactic: `simp`

`simp` can prove

$$f(a+b) - f(b) \in \{\mathrm{proj}_1(a,b)\}$$

$$\int_0^1 x^n \,\mathrm{d}x = \frac{1}{n+1}$$

Other automation

Other automation

- Decision procedures (Gröbner, Omega, etc)

Other automation

- Decision procedures (Gröbner, Omega, etc)
- Ganesalingam–Gowers

Other automation

- Decision procedures (Gröbner, Omega, etc)
- Ganesalingam–Gowers
- Machine learning, `gpt-f`

Other automation

- Decision procedures (Gröbner, Omega, etc)
- Ganesalingam–Gowers
- Machine learning, `gpt-f`
- Ad-hoc tactics

Other theorem provers

Automath, Mizar, Metamath, HOL, Isabelle, Coq, Agda

Other theorem provers

Automath, Mizar, Metamath, HOL, Isabelle, Coq, Agda

I'm not good at comparing them

Other theorem provers

Automath, Mizar, Metamath, HOL, Isabelle, Coq, Agda

I'm not good at comparing them

See articles and website of Freek Wiedijk

Mathlib overview

Started in 2017, now 20 maintainers

Mathlib overview

Started in 2017, now 20 maintainers

$\geqslant$ 170 contributors (profs, students, CS, maths)
$\geqslant$ 1800 files, $\approx 600,000$ lines of code

Mathlib overview

Started in 2017, now 20 maintainers

$\geqslant$ 170 contributors (profs, students, CS, maths)
$\geqslant$ 1800 files, $\approx 600,000$ lines of code

`leanprover-community.github.io/undergrad.html`

Mathlib overview

Started in 2017, now 20 maintainers

$\geqslant$ 170 contributors (profs, students, CS, maths)
$\geqslant$ 1800 files, $\approx 600,000$ lines of code

`leanprover-community.github.io/undergrad.html`

`leanprover-community.github.io/mathlib-overview.html`

Proof assistants and education

Several mathematicians in the various communities do this

Proof assistants and education

Several mathematicians in the various communities do this

I have no personal experience

Proof assistants and education

Several mathematicians in the various communities do this

I have no personal experience

Are you interested? Join Zulip and chat with us!

Pointers

Community website
`leanprover-community.github.io`

Chatroom
`leanprover.zulipchat.com`

Natural number game
`wwwf.imperial.ac.uk/~buzzard/xena/natural_number_game/`