Intuitionistic Logic And Type Theory

Albert Lai

March 2025

One More Talk

One More Talk

3 past talks on intuitionistic logic. Why one more?

Answer: They didn't cover quantifers: $\forall$ and $\exists$.

This talk fixes that. Just One More ~~Turn~~ Talk.

One More Talk

3 past talks on intuitionistic logic. Why one more?

Answer: They didn't cover quantifers: $\forall$ and $\exists$.

This talk fixes that. Just One More ~~Turn~~ Talk.

Bonus: From $\forall$ and user-definable types by induction, derive the rest, plus many data types, and equality. (What Lean and Agda do.)

CSC/MAT A68: Introduction to Proofs And Programs

Intuitionistic/Constructivist Logic

Philosophy: Proofs should be constructive:

- Proof of $A \vee B$ should tell you which one it goes for.
- Proof of $\exists$ should contain an algorithm to build an example.

Cons:

- Loses the duality of classical logic.
  E.g., $A \wedge \neg A$ is false, but $A \vee \neg A$ is not true.

Pros:

- Gains the duality of introduction and elimination.
  Every connective is characterized by introduction rules (how to prove it) and elimination rules (how to use it to prove other things). A.k.a. Natural Deduction.
- Equivalent to functional programming.

Overview

sentence $\cong$ type

proof of sentence $\cong$ program/element of type

sentence	type	set analog
true	Unit	singleton set
false	Empty	empty set
$A \wedge B$	$A \times B$	product
$A \vee B$	$A \uplus B$, $A + B$	disjoint union, sum
$A \to B$	B^A, $A \to B$	function space
$\neg A$, $A \to$ false	$A \to$ Empty	
$\forall x \colon A \cdot B(x)$	$\prod x \colon A \cdot B(x)$	indexed cartesian product
$\exists x \colon A \cdot B(x)$	$\sum x \colon A \cdot B(x)$	indexed sum

$A \to B$

	proof	program
intro	Given: locally assume A, get B. Get $A \to B$.	Given: locally assume $x\colon A$, get $e : B$. Get $(\lambda x\colon A \cdot e) : A \to B$.
elim	Given A, $A \to B$. Get B.	Given $a : A$, $f : A \to B$. Get $f(a) : B$.

(x variable, e may use x.)

$A \wedge B$, $A \times B$

	proof	program
intro	Given A, B. Get $A \wedge B$.	Given $a : A$, $b : B$. Get $\langle a, b \rangle : A \times B$.
elim	Given $A \wedge B$. Get A.	Given $e : A \times B$. Get $\mathsf{fst}(e) : A$.
	Given $A \wedge B$. Get B.	Given $e : A \times B$. Get $\mathsf{snd}(e) : B$.

$A \vee B, A + B$

	proof	program
intro	Given A. Get $A \vee B$.	Given $a : A$. Get $\mathsf{inl}(a) : A + B$.
	Given B. Get $A \vee B$.	Given $b : B$. Get $\mathsf{inr}(b) : A + B$.
elim	Given $A \vee B$, $A \to C$, $B \to C$. Get C.	Given $e : A + B$, $f_l : A \to C$, $f_r : B \to C$. Get $\mathsf{cases}(e, f_l, f_r) : C$.

true, Unit

	proof	program
intro	Get true.	Get $\star$: Unit.

false, Empty

	proof	program
elim	Given false. Get A.	Given e : Empty. Get miracle(e) : A.

$\forall$, $\prod$

	proof	program
intro	Given: locally assume $x\colon A$, get $B(x)$. Get $\forall x\colon A \cdot B(x)$.	Given: locally assume $x\colon A$, get $e : B(x)$. Get $(\lambda x\colon A \cdot e) : \prod x\colon A \cdot B(x)$.
elim	Given $a : A$, $\forall x\colon A \cdot B(x)$. Get $B(a)$.	Given $a : A$, $f : \prod x\colon A \cdot B(x)$. Get $f(a) : B(a)$.

(x variable, e may use x.)

Math Tip: Think indexed cartesian product: index set A, family B indexed by A.

CS/SE Tip: If you prefer FP, $\prod$ is for you!

$\exists$, $\sum$

	proof	program
intro	Given $a : A$, $B(a)$. Get $\exists x\colon A \cdot B(x)$.	Given $a : A$, $b : B(a)$. Get $\langle a, b\rangle : \sum x\colon A \cdot B(x)$.
elim	Given $\exists x\colon A \cdot B(x)$, $\forall x\colon A \cdot B(x) \to C$. Get C.	Given $e : \sum x\colon A \cdot B(x)$, $f : \prod x\colon A \cdot B(x) \to C$. Get dcase$(e, f) : C$.

Math Tip: $\prod$ is categorical limit, $\sum$ is categorical colimit!

Philosophy Tip: Cocartes says: coproduct/colimit ergo sum.

CS/SE Tip: If you prefer OOP, $\sum$ is for you!

“Solution to Enrolment Inflation”

Corollary (“Solution to Enrolment Inflation”):

Math students fail 1/2 of the course.
SE students fail at least 3/4 of the course (4/4 if prefer state variables).

“Solution to Enrolment Inflation”

Corollary (“Solution to Enrolment Inflation”):

Math students fail 1/2 of the course.
SE students fail at least 3/4 of the course (4/4 if prefer state variables).
CS theory-inclined students rage-quit CMS because prefer English.

CSC/MAT/PHL 2168: Type Theory

Multiple Levels of Types And Why

Motivation:

$\mathbb{N}$ induction is $\forall p\colon \mathbb{N} \to \text{Type} \cdot p(0) \wedge \ldots \to \forall n\colon \mathbb{N} \cdot p(n)$

$\mathbb{N} \to$ Type is the domain so should be a type.

"$\mathbb{N} \to$ Type is a type" yields Girard's paradox.

Multiple Levels of Types And Why

Motivation:
$\mathbb{N}$ induction is $\forall p\colon \mathbb{N} \to \text{Type} \cdot p(0) \wedge \ldots \to \forall n\colon \mathbb{N} \cdot p(n)$
$\mathbb{N} \to \text{Type}$ is the domain so should be a type.
"$\mathbb{N} \to \text{Type}$ is a type" yields Girard's paradox.

Solved preemptively by Russell: Tower of types with levels 0, 1, 2, ...

- Type 0 has $\mathbb{N}$, $\mathbb{N} \to \mathbb{B}$
- Type 1 has Type 0, $\mathbb{N} \to$ Type 0
- Type 2 has Type 1, $\mathbb{N} \to$ Type 1
- ...

Lean and Agda have that (and more for practicality or richness).

Dependent Product

Familiar notation: "$\forall x\colon A \cdot B(x)$", "$\prod x\colon A \cdot B(x)$", "$A \to B$" if B doesn't vary by x.

Unified notation: $(x\colon A) \to B(x)$
with $A \to B$ as special case.

Intro and elim rules as before.

$(x\colon A) \to (y\colon B) \to C(x,y)$ means $(x\colon A) \to ((y\colon B) \to C(x,y))$

Level:
if A : Type u and $B(x)$: Type v
then $(x\colon A) \to B(x)$: Type $\max(u,v)$

Refresher: Defining Sets by Induction

$\mathbb{N}$ is the smallest set such that

- $0 \in \mathbb{N}$
- for all $n \in \mathbb{N}$, $s(n) \in \mathbb{N}$

Recall: “smallest” means no other members, has induction, can use recursion.

Refresher: Defining Sets by Induction

$\mathbb{N}$ is the smallest set such that

- $0 \in \mathbb{N}$
- for all $n \in \mathbb{N}$, $s(n) \in \mathbb{N}$

Recall: “smallest” means no other members, has induction, can use recursion.

$\mathbb{B}$ is the smallest set such that: $b_0 \in \mathbb{B}$ and $b_1 \in \mathbb{B}$.

Refresher: Defining Sets by Induction

$\mathbb{N}$ is the smallest set such that

- $0 \in \mathbb{N}$
- for all $n \in \mathbb{N}$, $s(n) \in \mathbb{N}$

Recall: “smallest” means no other members, has induction, can use recursion.

$\mathbb{B}$ is the smallest set such that: $b_0 \in \mathbb{B}$ and $b_1 \in \mathbb{B}$.

Preview:

- Membership rules above become intro rules.
- Terminology: 0, s, b_0, b_1 are called constructors.
- Induction gives elim rules.

Defining Inductive Types: Overview

Declare type name, level, constructor names and types.

```
inductive N : Type 0 where
  | z : N
  | s : N -> N
```

Constructor types become intro rules.

Auto-generated:

- induction principle (good for both proofs and recursion)
  elim rules are easy special cases
- computation rules: how to execute recursion

Marvelous details too long to fit here, but try osmosis from examples.

Unit

```
inductive Unit : Type 0 where
  | * : B
```

Induction:
Unit.rec : $(p : \mathsf{Unit} \to \mathsf{Type}\, u) \to p(\star) \to (e : \mathsf{Unit}) \to p(e)$

Computation:
$\mathsf{Unit.rec}(p, a, \star) = a$

Empty

```
inductive Empty : Type 0 where
```

(i.e., there are 0 constructors / intro rules)

Induction:
Empty.rec: $(p : \text{Empty} \to \text{Type}\, u) \to (e : \text{Empty}) \to p(e)$

Elimination just sets $p := \lambda x \cdot A$:
miracle: $(A : \text{Type}\, u) \to \text{Empty} \to A$
miracle$(A, e) :=$ Empty.rec$((\lambda x \cdot A), e)$

(No computation rule provided or needed.)

Booleans

```
inductive B : Type 0 where
  | b0 : B
  | b1 : B
```

Induction:

$\mathbb{B}.\mathsf{rec}\colon (p\colon \mathbb{B} \to \mathsf{Type}\, u) \to p(b_0) \to p(b_1) \to (b\colon B) \to p(b)$

Computation:

$\mathbb{B}.\mathsf{rec}(p, e_0, e_1, b_0) = e_0$

$\mathbb{B}.\mathsf{rec}(p, e_0, e_1, b_1) = e_1$

Naturals

```
inductive N : Type 0 where
  | z : N
  | s : N -> N
```

Induction:

$$\mathbb{N}.\mathrm{rec}\colon (p\colon \mathbb{N} \to \mathsf{Type}\, u) \to p(z) \to ((n\colon \mathbb{N}) \to p(n) \to p(s(n))) \to (n\colon \mathbb{N}) \to p(n)$$

Computation:

$$\mathbb{N}.\mathrm{rec}(p, base, step, z) = base$$
$$\mathbb{N}.\mathrm{rec}(p, base, step, s(n)) = step(n, \mathbb{N}.\mathrm{rec}(p, base, step, n))$$

Example:

$$m + n := \mathbb{N}.\mathrm{rec}((\lambda v \cdot \mathbb{N}), m, (\lambda n, r \cdot s(r)), n)$$

Product

Need two domain parameters.

```
inductive Prod (A: Type u) (B: Type v) : Type max(u,v) where
  | <_,_> : A -> B -> Prod A B
```

Induction:

$\mathsf{Prod.rec}\colon (A\colon \mathsf{Type}\, u) \to (B\colon \mathsf{Type}\, v) \to (p\colon A \times B \to \mathsf{Type}\, w)$

$\to ((a\colon A) \to (b\colon B) \to p(\langle a, b\rangle))$

$\to (e : A \times B) \to p(e)$

Elimination derivable, e.g.,

$\mathsf{fst}(A, B, e) := \mathsf{Prod.rec}(A, B, (\lambda x \cdot A), (\lambda a, b \cdot a), e)$

Computation: $\mathsf{Prod.rec}(A, B, p, f, \langle a, b\rangle) = f(a, b)$

Sum

```
inductive Sum (A: Type u) (B: Type v) : Type max(u,v) where
  | inl : A -> Sum A B
  | inr : B -> Sum A B
```

Induction:

$$\begin{aligned}\text{Sum.rec}\colon (A\colon \text{Type}\, u) &\to (B\colon \text{Type}\, v) \to (p\colon A + B \to \text{Type}\, w)\\ &\to ((a\colon A) \to p(\text{inl}(a)))\\ &\to ((b\colon B) \to p(\text{inr}(b)))\\ &\to (e\colon A + B) \to p(e)\end{aligned}$$

Elimination just sets $p := \lambda e \cdot C$ and re-order arguments.

Computation:

$\text{Sum.rec}(A, B, p, f_l, f_r, \text{inl}(a)) = f_l(a)$

$\text{Sum.rec}(A, B, p, f_l, f_r, \text{inr}(b)) = f_r(b)$

Dependent Sum

```
inductive Sigma (A: Type u) (B: A -> Type v) : Type max(u,v) where
  | <_,_> : (a:A) -> B a -> Sigma A B
```

Induction:

Sigma.rec: $(A : \mathsf{Type}\, u) \to (B : A \to \mathsf{Type}\, v) \to (p : (\sum x : A \cdot B(x)) \to \mathsf{Type}\, w)$

$$\to ((a : A) \to (b : B(a)) \to p(\langle a, b \rangle))$$

$$\to (e : \sum x : A \cdot B(x)) \to p(e)$$

Elimination just sets $p := \lambda e \cdot C$ and re-order arguments.

Computation:

$\mathsf{Sum.rec}(A, B, p, f, \langle a, b \rangle) = f(a, b)$

Equality

```
inductive Eq (A: Type u) : A -> A -> Type 0 where
  | refl : (x : A) -> Eq A x x
```

Induction:

$\mathsf{Eq.rec} : (A : \mathsf{Type}\, u) \to (x : A) \to (p : (y : A) \to \mathsf{Eq}(A, x, y) \to \mathsf{Type}\, v)$

$\quad \to p(x, \mathsf{refl}(x))$

$\quad \to (y : A) \to (e : \mathsf{Eq}(A, x, y)) \to p(y, e)$

Intro rule is reflexivity $x = x$.

Elimination: $q(x) \to x = y \to q(y)$. (Set $p := \lambda w, e \cdot q(w)$ in induction.)

Symmetry, transitivity, Leibniz ($x = y \to f(x) = f(y)$) provable (next slide).

Equality is mostly user-definable, only missing computation rules for inductive types (require built-in auto-gen).

Proving Symmetry, Transitivity, Leibniz

Use reflexivity and elimination ($q(x) \to x = y \to q(y)$) to prove:

Symmetry:
Set $q := \lambda w \cdot w = x$
$x = x \to x = y \to y = x$

Transitivity:
Set $q := \lambda w \cdot z = w$
$z = x \to x = y \to z = y$

Leibniz:
Set $q := \lambda w \cdot f(x) = f(w)$
$f(x) = f(x) \to x = y \to f(x) = f(y)$

CSC/ECE/MAT/PHL/PHY/CHM/BIO/SOC/POL/LAW/LOL 2170:
Category Theory for Everyone!

Just kidding!

But see:

David Spivak (website),
Category Theory for the Sciences
Seven Sketches in Compositionality: An Invitation to Applied Category Theory

Eugenia Cheng (website),
The Joy of Abstraction: An Exploration of Math, Category Theory, and Life