Intuitionistic Logic, Type Theory, and Computer Science

Sewon Park
School of Computing, KAIST

The first Korea Logic Day 2021

January 14, 2021

Overview

1. Briefly introduce Intuitionistic Logic, Type Theory, and Computable Analysis
2. Construct an interpretation of an Intuitionistic Type Theory
3. Observe the logic of Computable Analysis (= Intuitionistic)

Intuitionistic Logic

1. The classical logic without non-constructive reasoning principles:
   - The law of excluded middle $P \vee \neg P$
   - Double negation elimination $\neg\neg P \to P$
2. Often introduce principles that contradict the classical logic: continuity principle
3. In consequence, proofs have to be done constructively:
   - $P \vee Q$: have to specify if it is P or Q
   - $\exists x.\ P(x)$: have to construct x that satisfies $P(x)$

The belief in the universal validity of the principle of the excluded third in mathematics is considered by the intuitionists as a phenomenon of the history of civilization of the same kind as the former belief in the rationality of π, or in the rotation of the firmament about the earth - L. E. J. Brouwer

Program Extraction

for any prime numbers $p_1 < \cdots < p_d$ *there exists a prime number* $p > p_d$

1. Suppose any prime $p_1 < \cdots < p_d$
2. Construct a natural number p
3. Prove (i) p is a prime number and (ii) $p > p_d$ holds

The proof contains information on how to construct p given $p_1, \cdots, p_d$

= (extracting computational content) ⇒

A computer program that computes p from $p_1, \cdots, p_d$

- We need a framework where "proof"s become objects

Type Theory

Dependent Type Theory

- A Foundation of Mathematics...
- A language where we can do all: Construct and Prove (and compute)

Construction	Sets $A, B, C, \cdots$	Elements $a, b, c, \cdots$
Logic	Propositions $a \in A$	Proofs
Type Theory	Types	Terms

Judgements

Type theory consists of judgements and rules for deriving judgements

- Types

$$\vdash A \ \mathsf{type}$$

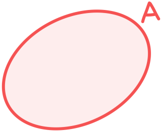
$\vdash \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathsf{N} \ \mathsf{type}$

- Terms

$$\vdash a : A$$

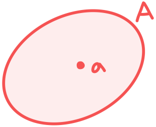
$\vdash * : \mathbf{1}, \vdash tt, \mathit{ff} : \mathbf{2}, \vdash 0 : \mathsf{N}$,
$n : \mathsf{N} \vdash S\ n : \mathsf{N}$

- Spaces



$\emptyset, \{*\}, \{tt, \mathit{ff}\}, \mathbb{N}$

- Points



$* \in \{*\}, tt, \mathit{ff} \in \{tt, \mathit{ff}\}$ $0 \in \mathbb{N}$, if $n \in \mathbb{N}$ then $S(n) \in \mathbb{N}$

Type judgement $\vdash a : A$ is not $a \in A$!

Identity Types

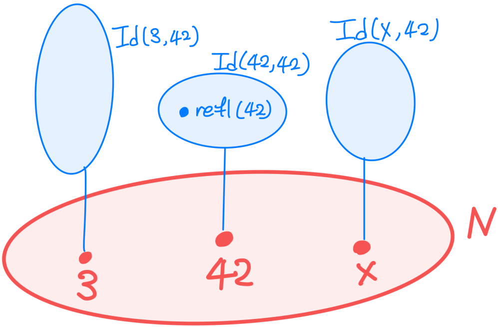
Type theory consists of judgements and rules for deriving judgements

- Type Formation:

$$\frac{\vdash a, b : A}{\vdash \mathsf{Id}(a, b)\ \mathsf{type}}$$

- Term Introduction:

$$\frac{\vdash a : A}{\vdash \mathsf{refl}(a) : \mathsf{Id}(a, a)}$$



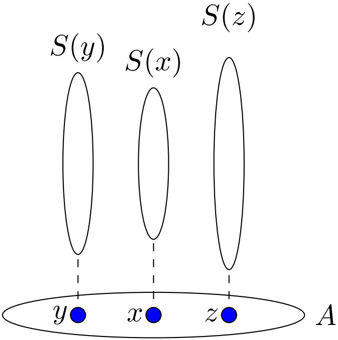
1. Propositions as types: $\vdash t : P$, t is a proof of P
2. Dependent types:

$$n : \mathsf{N} \vdash \mathsf{Id}(n, 42)\ \mathsf{type}$$

Π-Types

- Assuming a is a term of type A, $S(a)$ is a type:

$$a : A \vdash S(a) \ \mathsf{type}$$



- Type Formation:

$$\vdash \Pi(a : A)S(a) \ \mathsf{type}$$

- Term Intro.:

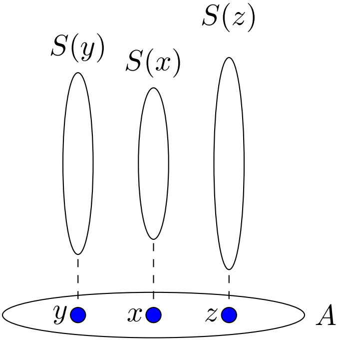
$$\vdash f : \Pi(a : A)S(a)$$

if $\vdash f(a) : S(a)$ for all $a : A$.

Σ-Types

- Assuming a is a term of type A, $S(a)$ is a type:

$$a : A \vdash S(a) \ \mathsf{type}$$



- Type Formation:

$$\vdash \Sigma(a : A)S(a) \ \mathsf{type}$$

- Term Intro:

$$\vdash (a, s) : \Sigma(a : A)S(a)$$

if $\vdash a : A$ and $\vdash s : S(a)$

Π, Σ as Quantifiers?

$$\mathsf{isEven}(n) :\equiv \Sigma(k : \mathsf{N}).\ \mathsf{Id}(2 \times k, n)$$

$$\mathsf{isOdd}(n) :\equiv \Sigma(k : \mathsf{N}).\ 2 \times \mathsf{Id}(k + 1, n)$$

1. $k : \mathsf{N}, m : \mathsf{N} \vdash p(k, m) : 2 \times k = n$ is an identification "$2 \times k = n$"
2. A pair (k, p) is of type $\Sigma(k : \mathsf{N}).\ 2 \times k = n$ if $k : \mathsf{N}$ and $p : 2 \times k = n$
3. A proof of $\mathsf{isEven}(n)$ is a pair of a natural number k and the reason why $2 \times k = n$

$$\vdash f : \Pi(n : \mathsf{N}).\ \mathsf{isOdd}(n) + \mathsf{isEven}(n)$$

1. A function f where $f(n) = (b, k, p)$ such that b indicates (i) if n is even or not, (ii) k is a natural number, (iii) and p says why $n = 2 \times k$ or $2 \times k + 1$

Axioms

- Thus far, we haven't mentioned anything about principles
- Axioms are given as a term constant:
- Law of Excluded middle

$$\frac{\vdash P\ \mathsf{type}}{\vdash \mathsf{LEM}(P) : \mathsf{isProp(P)} \to P + \neg P}$$

- Double Negation Elim.:

$$\frac{\vdash P\ \mathsf{type}}{\vdash \mathsf{LEM}(P) : \mathsf{isProp(P)} \to \neg\neg P \to P}$$

- Continuity Principle, $\neg\neg$-stability, Function Extensionality, Propositional Extensionality, Markov Principle, and so on

Computable Mathematics

Computable Mathematics

In computer science, we

1. Define what it means to compute mathematical objects.
2. Classify functions that are "computable" and that are not
3. Classify functions w.r.t. their computational complexity

A mathematical universe where we only have computable functions

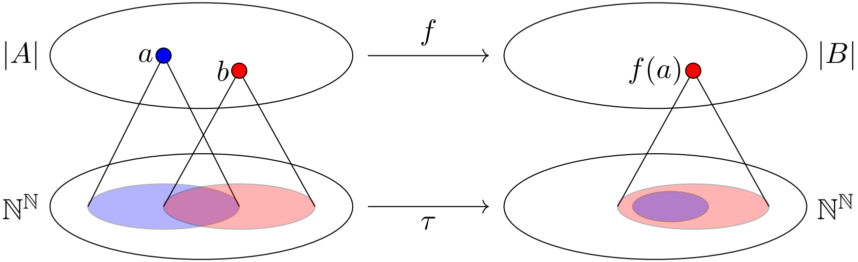
Assemblies

- An assembly $A = (|A|, \Vdash_A)$ is a pair of a set $|A|$ and a surjective relation $\Vdash_A \subseteq |A| \times \mathbb{N}^{\mathbb{N}}$:

$$\forall (a \in |A|).\ \exists (\phi \in \mathbb{N}^{\mathbb{N}}).\ \phi \Vdash_A a$$

- A function $f : A \to B$ is a computed by $\tau : \mathbb{N}^{\mathbb{N}} \rightharpoonup \mathbb{N}^{\mathbb{N}}$ if

$$\forall (a \in |A|).\ \forall (\phi \in \mathbb{N}^{\mathbb{N}}).\ \phi \Vdash_A a \Rightarrow \tau(\phi) \Vdash_B f(a)$$



Category of Assemblies Asm

1. $\mathbf{0} := (\emptyset, \emptyset)$
2. $\mathbf{1} := (\{*\}, \{*\} \times \mathbb{N}^{\mathbb{N}})$
3. $\mathbf{2} := (\{tt, ff\}, \{tt\} \times \{\phi \mid \phi(0) = 0\} \cup \{ff\} \times \{\phi \mid \phi(0) = 1\})$
4. $(\mathbb{N}, \{(n, \phi) \mid \phi(0) = n\})$
5. $(\mathbb{R}_+, \Vdash_{\mathbb{R}_+})$ where

$$\phi \Vdash_{\mathbb{R}_+} x :\Leftrightarrow |\phi(n)/2^{-n} - x| \leq 2^{-n}$$

6. When we collect assemblies and computable functions, it forms a category (Quasi-topos) Asm.

Realizabiltiy

Type Theory	Asm
$\vdash A$ type	$[\![A]\!] \in \mathrm{Ob}(\mathsf{Asm})$
$\vdash a : A$	$[\![a]\!] \in [\![A]\!]$

- If $[\![-]\!]$ is defined recursively to the construction of terms and types, we can get the program that computes $[\![a]\!] \in [\![A]\!]$ from $\vdash a : A$.

1. $[\![\mathbf{0}]\!] :\equiv \mathbf{0} \quad [\![\mathbf{1}]\!] :\equiv \mathbf{1} \quad [\![\mathbf{2}]\!] :\equiv \mathbf{2}$
2. $[\![A \to B]\!] :\equiv [\![B]\!]^{[\![A]\!]}$
3. $[\![\mathsf{Id}(a,b)]\!] :\equiv \begin{cases} \mathbf{1} & \text{if } [\![a]\!] = [\![b]\!], \\ \mathbf{0} & \text{otherwise.} \end{cases}$

Family of Assemblies

A family of assemblies indexed by a assembly A:

$$\mathcal{F} : A \to \mathsf{Asm}$$

Definition (Dependent Product)

$\Pi_{a\in|A|}\mathcal{F}(a)$ is an assembly

- $\Pi_{a\in|A|}\mathcal{F}(a) := \{f : |A| \to \bigcup_{a\in|A|} |\mathcal{F}(a)| \mid f \text{ is trackable}\}$
  f is trackable $:\Leftrightarrow$ there is τ s.t. $\forall\alpha \Vdash_A a.\ \tau(a) \Vdash_{\mathcal{F}(a)} f(a)$

Definition (Dependent Pair)

$\Sigma_{a\in|A|}\mathcal{F}(a)$ is an assembly

- $|\Sigma_{a\in|A|}\mathcal{F}(a)| := \{(a, b) \in |A| \times (\bigcup_{a\in|A|} |\mathcal{F}(a)|) \mid b \in \mathcal{F}(a)\}$
- $\langle\alpha, \beta\rangle \Vdash_{\Sigma_{a\in|A|}\mathcal{F}(a)} (a, b) :\Leftrightarrow \alpha \Vdash_A a \wedge \beta \Vdash_{\mathcal{F}(a)} b$

Dependent Type Theory as a Language of Asm

- When we prove

$$\vdash f : \Pi(x : \mathsf{N}).\ \Sigma(y : \mathsf{N}).\ \mathsf{isNice}(x, y)$$

  we get computable $[\![f]\!] : \mathbb{N} \to \mathbb{N}$ that computes <u>Nice</u> number from x.
- A proof in the type theory automatically becomes a proof in computable analysis + computability result
- the logic of Computable Analysis by checking the validity of axioms

the Logic of Computable Analysis

1. A principle $\vdash A$ type is valid in Asm if $[\![A]\!]$ has a point
2. law of excluded middle is not valid

$$[\![\Pi(A:U)A+\neg A]\!] = [\![\Pi(A:U)A \vee \neg A]\!] = \mathbf{0}$$

3. Functional Extensionality, Markov Principle, Continuity Principle are valid
4. Extensionality of Identity is valid:

$$\Pi(p,q:\mathsf{Id}(a,b)).\ \mathsf{Id}(p,q)$$

5. Propositional extensionality is not valid:

$$\mathsf{isProp}(P) \to \mathsf{isProp}(Q) \to (P \to Q) \to (Q \to P) \to \mathsf{Id}(P,Q)$$

6. Hence, Univalence Axiom is not valid.

Conclusion

Conclusion

1. Introduced intuitionistic logic and a minimal dependent type theory
2. Constructed an interpretation of the type theory in Asm
3. Checked which principles that Asmadmits