Dependent Type Theory

Why, how and what to do with it?

Ruxandra Icleanu
Mentor: Julian Gould

DRP Fall 2022

Foundations

A few possible candidates for approaching foundations of mathematics:

- set theory
- category theory
- type theory - everything is a type or a term of a given type

Simple Type Theory

- Simply typed lambda calculus
- "$q : Q$" $\Leftrightarrow$ q is a term of type Q $\Leftrightarrow$ q is a proof (or witness) of Q
- syntax: context $\vdash$ conclusion
- we have rules for formation, introduction, elimination, and computation
- we can prove statements like $P \wedge Q \rightarrow P$
- "$\rightarrow$": corresponds to implication (logic)/ function (set)

Simple Type Theory

$$\frac{P\ \text{type} \qquad Q\ \text{type}}{P \wedge Q\ \text{type}}\ \wedge\text{- form} \qquad\qquad \frac{P\ \text{type} \qquad Q\ \text{type}}{P \to Q\ \text{type}}\ \to\text{- form}$$

$$\frac{\Gamma \vdash p : P \qquad \Gamma \vdash q : Q}{\Gamma \vdash (p, q) : P \wedge Q}\ \wedge\text{- intro} \qquad\qquad \frac{\Gamma \vdash x : P \vdash q : Q}{\Gamma \vdash \lambda x.q : P \to Q}\ \to\text{- intro}$$

$$\frac{\Gamma \vdash s : P \wedge Q}{\Gamma \vdash \text{left-pr}\ s : P}\ \wedge\text{- elim - r} \qquad\qquad \frac{\Gamma \vdash p : P \qquad f : P \to Q}{\Gamma \vdash f\,p : Q}\ \to\text{- elim}$$

$$\frac{\Gamma \vdash s : P \wedge Q}{\Gamma \vdash \text{right-pr}\ s : Q}\ \wedge\text{- elim - l}$$

$\wedge$ - and $\to$ - computation rules (omitted)

Dependent Type Theory

$\approx$ simple type theory $+$ dependent types

- a type can depend on a term of another type
  e.g. type $Vect(n)$ of vectors of length n
  type $isPrime(n)$
- if types are propositions, then DT are predicates
- if types are sets, then DT are indexed families of sets
- if types are programs, then DT are programs with a given parameter

Dependent Type Theory

$$\frac{\Gamma \vdash P \text{ type} \quad \Gamma \vdash Q \text{ type}}{\Gamma \vdash P \wedge Q \text{ type}} \; \wedge\text{ - form}$$

$$\frac{\Gamma \vdash P \text{ type} \quad \Gamma \vdash Q \text{ type}}{\Gamma \vdash P \rightarrow Q \text{ type}} \; \rightarrow\text{ - form}$$

$$\frac{\Gamma \vdash p : P \quad \Gamma \vdash q : Q}{\Gamma \vdash (p, q) : P \wedge Q} \; \wedge\text{ - intro}$$

$$\frac{\Gamma \vdash x : P \vdash q : Q}{\Gamma \vdash \lambda x.q : P \rightarrow Q} \; \rightarrow\text{ - intro}$$

$$\frac{\Gamma \vdash s : P \wedge Q}{\Gamma \vdash \text{left-pr } s : P} \; \wedge\text{ - elim - r}$$

$$\frac{\Gamma \vdash s : P \wedge Q}{\Gamma \vdash \text{right-pr } s : Q} \; \wedge\text{ - elim - l}$$

$$\frac{\Gamma \vdash p : P \quad f : P \rightarrow Q}{\Gamma \vdash f\, p : Q} \; \rightarrow\text{ - elim}$$

$\wedge$ - and $\rightarrow$ - computation rules (omitted)

Dependent Type Theory: Function types

if $x : A \vdash B(x)$, then $\prod_{x:A} B(x)$ is a type

How to interpret them in

- set theory?
- logic?

e.g. $\prod_{n \in \mathbb{N}} Vect(n)$

Dependent Type Theory: Inductive Types

One example: Dependent pair Types (Σ types)

For a type family B over A, we can consider pairs (a, b) of terms with $a : A$ and $b : B(a)$

intuition: there is no term of type $\prod_{n \in \mathbb{N}} isOdd(n)$
but there are plenty of terms of type $\Sigma_{n \in \mathbb{N}} isOdd(n)$

How to interpret them in
- set theory?
- logic?

Proof assistants

Tool for formal proofs based on the Curry-Howard isomorphism
propositions $\Leftrightarrow$ types

Proof assistants: Example

We want to formalise a basic statement from group theory:

"In any group G, e is unqiue, i.e. if $x \in G, \forall y \in$ satisfying $xy = yx = x$, we have that $x = e$."

Proof assistants: Example

Type of groups? Can be defined as following

$A : Set$

$e : A$

$inv : A \to A$

$m : A \times A \to A$ (or equivalently, $m : A \to (A \to A)$)

Group $G := \sum_{A:Set} \sum_{e:A} \sum_{i:A \to A} \sum_{m:A \times A \to A}$ (axioms)

Axioms

- associativity: $a * (b * c) = (a * b) * c, \forall a, b, c \in G$

  $\Leftrightarrow$

  define is_associative$(m) := \prod_{a,b,c:A} (m(a, m(b, c)) = m(m(a, b), c))$

Similarly, we can define functions that enforce the axiom for identity and inverse:

- identity:
  $\text{left\_id}(e) :\equiv \prod_{x:G}(m(e, x) = x)$
  $\text{right\_id}(e) :\equiv \prod_{x:G}(m(x, e) = x)$
- inverse:
  $\text{left\_inv}(i) :\equiv \prod_{x:G}(m(i(x), x) = e)$
  $\text{right\_inv}(i) :\equiv \prod_{x:G}(x, m(i(x)) = e)$

Back to the goal: formalising the fact that identity is unique

$$\prod_{x:G}\prod_{y:G}((m(x, y) = x \land m(y, x) = x) \Rightarrow x = e)$$

References

- Introduction to Homotopy Type Theory - Egbert Rijke
- School on Univalent Mathematics, Cortona, 2022