Last time: • Structure of dependent type theory
• Π - types
• $\to$ - types
• $\wedge / \times$ - types

$\}$ §1-2 of Rijke

This time: • Inductive types $\}$ §3-4 of Rijke

Next time • The identity type $\to$ homotopy $\}$ §5 of Rijke

General idea

- Type formers: The rules that define Π, $\to$, $\wedge/\times$, inductive types are all type formers.

  When we study type theory, we can choose which type formers to include.

  When people talk about HoTT, they often mean a type theory with particular type formers (the ones introduced in this course) + an axiom.

- Comparison with ZF-based mathematics:
  - Products, functions, etc have to be encoded in ZF.
  - Thus, everyday mathematics is far away from foundational rules.
  - In type theory, we postulate that products, functions, etc exist.
  - In type theory, everyday mathematics is closer to foundations.

- Inductive types are <u>freely generated</u> by <u>canonical terms</u>.

- In Agda:

```
data Bool : Type where
  true false : Bool
```

Bool is generated by the two canonical terms true, false.

To define a (dependent) function out of Bool, it suffices to define it on its canonical elements true and false.

The

data — where

syntax tells Agda that we are defining an inductive Type.

In pen-and-paper HoTT, we specify the behavior of inductive types by hand.

<u>The booleans</u>: bool

bool - form:
$$\frac{}{\vdash \text{bool type}}$$

bool-intro: $$\frac{}{\vdash \mathrm{true} : \mathrm{bool}} \qquad \frac{}{\vdash \mathrm{false} : \mathrm{bool}}$$

bool-elim: $$\frac{\Gamma, x : \mathrm{bool} \vdash D \ \mathrm{type} \qquad \Gamma \vdash a : D[\mathrm{true}/x] \qquad \Gamma \vdash b : D[\mathrm{false}/x]}{\Gamma, x : \mathrm{bool} \vdash \mathrm{ind\text{-}bool}_{a,b} : D}$$

bool-comp: $$\frac{\Gamma, x : \mathrm{bool} \vdash D \ \mathrm{type} \qquad \Gamma \vdash a : D[\mathrm{true}/x] \qquad \Gamma \vdash b : D[\mathrm{false}/x]}{\begin{array}{c}\Gamma \vdash \mathrm{ind\text{-}bool}_{a,b}[a/x] \doteq a : D[\mathrm{true}/x] \\ \Gamma \vdash \mathrm{ind\text{-}bool}_{a,b}[b/x] \doteq b : D[\mathrm{false}/x]\end{array}}$$

Ex. not : bool $\to$ bool

(Remember $\to$-intro: $$\frac{x : P \vdash q : Q}{\lambda x . q : P \to Q}$$)

weakened bool-form

$$\frac{\dfrac{\dfrac{}{x : \mathrm{bool} \vdash \mathrm{bool} \ \mathrm{type}} \qquad \dfrac{}{\vdash \mathrm{false} : \mathrm{bool} \doteq \mathrm{bool}[\mathrm{true}/x]} \qquad \dfrac{}{\vdash \mathrm{true} : \mathrm{bool} \doteq \mathrm{bool}[\mathrm{false}/x]}}{x : \mathrm{bool} \vdash \mathrm{ind\text{-}bool}_{\mathrm{false},\mathrm{true}} : \mathrm{bool}}}{\vdash \lambda x . \mathrm{ind\text{-}bool}_{\mathrm{false},\mathrm{true}} : \mathrm{bool} \to \mathrm{bool}}$$

Coproducts $+$

$+$-form:
$$\frac{\Gamma \vdash P \text{ type} \qquad \Gamma \vdash Q \text{ type}}{\Gamma \vdash P + Q \text{ type}}$$

$+$-intro:
$$\frac{\Gamma \vdash p : P}{\Gamma \vdash \mathrm{inl}(p) : P + Q} \qquad \frac{\Gamma \vdash q : Q}{\Gamma \vdash \mathrm{inr}(q) : P + Q}$$

$+$-elim:
$$\frac{\begin{array}{c}\Gamma, x : P + Q \vdash D \text{ type} \\ \Gamma, p : P \vdash a : D[\mathrm{inl}(p)/x] \\ \Gamma, q : Q \vdash b : D[\mathrm{inr}(q)/x]\end{array}}{\Gamma, x : P + Q \vdash \text{ind-}+_{a,b} : D}$$

$+$-elim:
$$\frac{\begin{array}{c}\Gamma, x : P + Q \vdash D \text{ type} \\ \Gamma, p : P \vdash a : D[\mathrm{inl}(p)/x] \\ \Gamma, q : Q \vdash b : D[\mathrm{inr}(q)/x]\end{array}}{\begin{array}{c}\Gamma, p : P \vdash \text{ind-}+_{a,b}[\mathrm{inl}(p)/x] \doteq a : D[\mathrm{inl}(p)/x] \\ \Gamma, q : Q \vdash \text{ind-}+_{a,b}[\mathrm{inl}(q)/x] \doteq b : D[\mathrm{inl}(q)/x]\end{array}}$$

Logical interpretation:

- We can prove (produce a term of) $P+Q$ if we can prove P or we can prove Q.
- To prove something from $P+Q$ we do a proof by cases.
- So $+$ behaves like disjunction (or).

Ex. For any types A, B, C, there is a function

$$A\times B + A\times C \to A\times(B+C).$$

$$\dfrac{\dfrac{\overline{x_1 : A\times B \vdash (\mathrm{pr}_1 x_1, \mathrm{inl}\,\mathrm{pr}_2 x_1) : A\times(B+C)} \qquad \overline{x_2 : A\times C \vdash (\mathrm{pr}_1 x_2, \mathrm{inr}\,\mathrm{pr}_2 x_2) : A\times(B+C)}}{x : A\times B + A\times C \vdash ? : A\times(B+C)}}{\vdash ? : A\times B + A\times C \to A\times(B+C).}$$

Dependent pair types (aka dependent sum types, Sigma types) Σ

Σ- form:
$$\frac{\Gamma,\ x:P \vdash Q}{\Gamma \vdash \sum_{x:P} Q}$$

Σ-intro: $$\frac{\Gamma \vdash p : P \qquad \Gamma \vdash q : Q[p/x]}{\Gamma \vdash \mathrm{pair}(p,q) : \sum_{x:P} Q}$$

Σ-elim: $$\frac{\Gamma, y : \sum_{x:P} Q \vdash D \text{ type} \qquad \Gamma, x:P, z:Q \vdash a : D[\mathrm{pair}(x,z)/y]}{\Gamma, y : \sum_{x:P} Q \vdash \mathrm{ind}_\Sigma(a,y) : D}$$

Σ-comp: $$\frac{\Gamma, y : \sum_{x:P} Q \vdash D \text{ type} \qquad \Gamma, x:P, z:Q \vdash a : D[\mathrm{pair}(x,z)/y]}{\Gamma, x:P, z:Q \vdash \mathrm{ind}_\Sigma(a, \mathrm{pair}(x,z)) \doteq a : D[\mathrm{pair}(x,z)/y]}$$

Logical interpretation: To prove $\sum_{x:P} Q$ (thinking of P as a set and Q as a predicate on P), we need to produce a term $p:P$ and prove $Q[p/x]$. Thus, it behaves like $\exists_{x:P} Q(x)$.

Set interpretation: The canonical terms are $\mathrm{pair}(p,q)$. It behaves like $\bigsqcup_{x:P} Q(x)$.

Ex. Let $\mathrm{Vect}(n)$ denote the type of vectors of length $n:\mathbb{N}$. Then $\sum_{n:\mathbb{N}} \mathrm{Vect}(n)$ is the type of all vectors.

Ex. For any $x:P \vdash Q$, there is a projection function $\pi : \sum_{x:P} Q \to P$.

$$\dfrac{\dfrac{x:P,\ z:Q \vdash x : P}{y : \sum_{x:P} Q \vdash \mathrm{ind}_\Sigma(x,y) : P}}{\lambda x.\mathrm{ind}_\Sigma(x,y) : \sum_{x:P} Q \longrightarrow P}$$

Ex. There is a projection function

$$\begin{array}{ccccc} \sum_{n:\mathbb{N}} \mathrm{Vect}(n) & \qquad \text{cf.} \qquad & \coprod_{n:\mathbb{N}} \mathrm{Vect}(n) & \quad \text{in sets} \\ \downarrow & & \downarrow & \\ \mathbb{N} & & \mathbb{N} & \end{array}$$

Ex. Consider a dependent type $x:\mathrm{bool} \vdash D(x)$ type.
There is a function $\left[\sum_{x:\mathrm{bool}} D(x)\right] \to D(\mathrm{true}) + D(\mathrm{false})$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \lambda z.\mathrm{inl}\,z : D(\mathrm{true}) \to D(\mathrm{true})+D(\mathrm{false})} \qquad \dfrac{}{\vdash \lambda z.\mathrm{inr}\,z : D(\mathrm{false}) \to D(\mathrm{true})+D(\mathrm{false})}}{x:\mathrm{bool} \vdash \mathrm{ind}_{\mathrm{bool}}(\lambda z.\mathrm{inl}\,z, \lambda z.\mathrm{inr}\,z, x) : D(x) \to D(\mathrm{true})+D(\mathrm{false})}}{x:\mathrm{bool},\ y:D(x) \vdash \mathrm{ind}_{\mathrm{bool}}(\lambda z.\mathrm{inl}\,z, \lambda z.\mathrm{inr}\,z, x)\ y : D(\mathrm{true})+D(\mathrm{false})}}{z : \sum_{x:\mathrm{bool}} D(x) \vdash \mathrm{ind}_{\Sigma}(\mathrm{ind}_{\mathrm{bool}}(\lambda z.\mathrm{inl}\,z, \lambda z.\mathrm{inr}\,z, x)\ y, z) : D(\mathrm{true})+D(\mathrm{false})}}{\vdash \lambda z.\ \mathrm{ind}_{\Sigma}(\mathrm{ind}_{\mathrm{bool}}(\lambda z.\mathrm{inl}\,z, \lambda z.\mathrm{inr}\,z, x)\ y, z) : \left[\sum_{x:\mathrm{bool}} D(x)\right] \to D(\mathrm{true})+D(\mathrm{false})}$$

The natural numbers $\mathbb{N}$

$\mathbb{N}$-form:

$$\dfrac{}{\vdash \mathbb{N}\ \text{type}}$$

$\mathbb{N}$-intro:

$$\dfrac{}{\vdash 0 : \mathbb{N}} \qquad\qquad \dfrac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash sn : \mathbb{N}}$$

$\mathbb{N}$-elim:

$$\frac{\begin{array}{c}\Gamma, x:\mathbb{N} \vdash D \text{ type} \\ \Gamma \vdash a : D[0/x] \\ \Gamma, x:\mathbb{N}, y:D \vdash b : D[sx/x]\end{array}}{\Gamma, x:\mathbb{N} \vdash \mathrm{ind}_{\mathbb{N}}(a,b,x) : D}$$

$\mathbb{N}$-comp:

$$\frac{\begin{array}{c}\Gamma, x:\mathbb{N} \vdash D \text{ type} \\ \Gamma \vdash a : D[0/x] \\ \Gamma, x:\mathbb{N}, y:D \vdash b : D[sx/x]\end{array}}{\begin{array}{c}\Gamma \vdash \mathrm{ind}_{\mathbb{N}}(a,b,0) \doteq a : D[0/x] \\ \Gamma, x:\mathbb{N}, y:D \vdash \mathrm{ind}_{\mathbb{N}}(a,b,sx) \doteq b : D[sx/x]\end{array}}$$

Ex. $sss0$ is (in usual parlance) 3.

Ex. $\iota : \mathrm{bool} \to \mathbb{N}$

$$\frac{\dfrac{\begin{array}{c}\vdash s0 : \mathbb{N} \\ \vdash 0 : \mathbb{N}\end{array}}{x:\mathrm{bool} \vdash \mathrm{ind}_{\mathrm{bool}}(s0,0,x) : \mathbb{N}}}{\vdash \lambda x.\mathrm{ind}_{\mathrm{bool}}(s0,0,x) : \mathrm{bool} \to \mathbb{N}}$$

Ex. $\mathrm{add} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$

$$x:\mathbb{N} \vdash 0 : \mathbb{N}$$

$$x:\mathbb{N}, u:\mathbb{N}, z:\mathbb{N} \vdash sz : \mathbb{N}$$

$$\dfrac{\dfrac{x:\mathbb{N}, y:\mathbb{N} \vdash \mathrm{ind}_{\mathbb{N}}(0, sz, y) : \mathbb{N}}{x:\mathbb{N} \vdash \lambda y.\, \mathrm{ind}_{\mathbb{N}}(0, sz, y) : \mathbb{N} \to \mathbb{N}}}{\lambda x. \lambda y.\, \mathrm{ind}_{\mathbb{N}}(0, sz, y) : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})}$$

Ex. mult : $\mathbb{N} \to (\mathbb{N} \to \mathbb{N})$

$$x:\mathbb{N} \vdash 0:\mathbb{N}$$

$$x:\mathbb{N}, y:\mathbb{N}, z:\mathbb{N} \vdash \mathrm{add}(z, y)$$

$$\dfrac{\dfrac{x:\mathbb{N}, y:\mathbb{N} \vdash \mathrm{ind}_{\mathbb{N}}(0, \mathrm{add}(z,y)) : \mathbb{N}}{x:\mathbb{N} \vdash \lambda y.\, \mathrm{ind}_{\mathbb{N}}(0, \mathrm{add}(z,y)) : \mathbb{N} \to \mathbb{N}}}{\lambda x. \lambda y.\, \mathrm{ind}_{\mathbb{N}}(0, \mathrm{add}(z,y)) : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})}$$

Why do we need the identity type? (if we're not interested in homotopy)

We already have a notion of equality:

$\doteq$ judgmental equality

(The identity type is called propositional equality.)

Logical interpretation: types are propositions / terms are proofs
$\rightarrow$ Proving equality means constructing a term of an equality type

We can prove many judgmental equalities:

Ex. $\mathrm{add}(x, 0) \doteq x$
$\mathrm{add}(x, sy) \doteq s\, \mathrm{add}(x, y)$

... but not all the ones we want.

Ex. $\mathrm{add}(0, x) \doteq x$
$\mathrm{add}(sx, y) \doteq s\ \mathrm{add}(x, y)$

$\mathbb{N}$-elim:
(aka induction)

$$\frac{\begin{matrix} x:\mathbb{N} \vdash D(x)\ \text{type} \\ \vdots \end{matrix}}{x:\mathbb{N} \vdash \mathrm{ind} : D(x)}$$

Type constructors often internalize structure

- At a 'meta' level, we can talk about contexts:
  Ex. $x:A,\ y:B(x),\ z:C(x,y) \vdash$

  We can discuss this at the 'type-and-term level' using Σ-types:

  Ex. $z : \sum_{x:A} \sum_{y:B(x)} C(x,y)$

- Similarly, we can talk about dependent terms as 'meta' function:

  Ex. $x:A,\ y:B(x) \vdash c(x,y) : C(x,y)$

  We can internalize such 'meta'-functions as terms of a Π-type

  Ex. $c : \prod_{x:A} \prod_{y:B(x)} C(x,y)$

- bool
- $\mathbb{N}$
- $\emptyset$
- $\mathbb{1}$

  can be seen as internalizing external notions

- The universe type internalizes the judgment A type.

- We'll see that the identity type internalizes judgmental equality.

<u>Identity type</u> =

NB: Compare these with the rules in Rijke.
"based path induction"

$=$-form $$\frac{\ulcorner\vdash A \text{ type} \quad \ulcorner\vdash a : A \quad \ulcorner\vdash b : A}{a =_A b \text{ type}}$$

$=$-intro $$\frac{\ulcorner\vdash a : A}{\ulcorner\vdash r_a : a =_A a}$$
(<u>r</u>eflexivity)

$=$-elim $$\frac{x : A,\ y : A,\ z : x =_A y \vdash D(x,y,z) \text{ type} \qquad x : A \vdash d : D(x,x,r_x)}{x : A,\ y : A,\ z : x =_A y \vdash \mathrm{ind}_=(d,x,y,z) : D(x,y,z)}$$

$=$-comp $$\frac{x : A,\ y : A,\ z : x =_A y \vdash D(x,y,z) \text{ type} \qquad x : A \vdash d : D(x,x,r_x)}{x : A \vdash \mathrm{ind}_=(d,x,x,r_x) \doteq d : D(x,x,r_x)}$$

Type constructors internalize structure

- We can talk about judgmental equality at a 'meta' level.

<u>Ex.</u> $a \doteq b : A$

We can internalize this using identity types.

<u>Ex.</u> $r_a : a =_A b$

$$\left\{\begin{array}{l} \text{If } a \doteq b : A, \text{ then } (a =_A b) \doteq (a =_A a), \text{ and} \\ \text{if } r_a : a =_A a, \text{ then } r_a : a =_A b. \end{array}\right.$$

$\rightarrow$ Reflexity (r_-) turns judgmental equalities into propositional equalities.

Inverse of equalities : $\prod_{a,b:A} a =_A b \to b =_A a$

$$\dfrac{\dfrac{a:A \vdash r_a : a =_A a}{a,b:A,\ p : a =_A b \vdash \mathrm{ind}_=(r,a,b,p) : b =_A a}}{\lambda a,b,p.\ \mathrm{ind}_=(r,a,b,p) : \prod_{a,b:A} a =_A b \to b =_A a}$$