More types, A o C

"Rem 1.5.1"

[0) Formation rules : data needed]

1) Constructors

2) Eliminators

"recursor" := 3) Computation (elim o constr)

{ dependent version

"inductor" 4) Uniqueness [optional] :

judgmental / propositional

for maps into/out of

Functions cont'd $A \to B$

<u>Constructors</u> λ-abstraction $\longleftrightarrow$ writing '$f(x) :\equiv \phi$'

$(\lambda x, \phi)$

s.t. $\phi : B$ if $x : A$.

<u>Elim</u> application(s) : for each $a : A$, have

$(A \to B) \xrightarrow{\text{"ev}_a\text{"}} B$ (not [yet] specified !)

Comp $\quad ev_a(\lambda x, \phi) :\equiv (\lambda x, \phi)(a)$

$$:\equiv \phi[x/a]$$

Uniqueness (judgmental) for "out of":

$$\text{if } f(x) \equiv \phi \quad \rightsquigarrow$$

$$f \equiv \lambda x, f(x)$$

("functions are det'd by values")

Π - types $\quad$ generalise fct types.

intermezzo: universes $\quad$ we will need families of types $\rightsquigarrow$ need universes ("collections of types")

$$A \text{ type} \overset{(\text{meta})}{\Rightarrow} A : \mathcal{U},$$

$\mathcal{U}$ must itself be a type

$\rightarrow \quad \mathcal{U} : V \quad$, etc.

Postulate hierarchy:

if you want $\mathcal{U}_{-1}$: $\mathcal{U}_0 : \mathcal{U}_1 : \cdots$

($\exists$ in Lean) (see below)

cumulative: $\quad A : \mathcal{U}_i : \mathcal{U}_{i+1}$

$\Rightarrow \quad A : \mathcal{U}_{i+1}$ $\quad$ (membership is transitive).

An <u>A-family of types</u>:

$j : A \to \mathcal{U}$, $\mathcal{U}$ some $\mathcal{U}_i$,

where $A \to \mathcal{U}$ is just a function type!

In particular, $j \equiv \lambda x, j(x)$.

Given $j : A \to \mathcal{U}$ [formation rule],

have a type

$$\prod_{a:A} j(a) \equiv \Pi\,(x:A),\, j(a)$$

Lean's notation

<u>Constr</u> want functions f out of A,

while allowing $f(a) : j(a)$,

i.e. the codomain type to vary.

$\rightsquigarrow$ again just λ-abs :

$\underline{\lambda a, \phi}$ s.t. $\phi : j(a)$ if $a : A$

<u>Elim</u> again application : given $x : A$,

have $\prod_{(a:A)} j(a) \xrightarrow{ev_x} j(x)$

<u>Comp</u> again $ev_x \circ \lambda a, \phi \equiv \phi[a/x]$,

also written $(\lambda a, \phi)(x)$

<u>Uniq'ss</u> same.

polymorphic version

this is in particular

... universe $\mathcal{U}$:

def'd for A a universe ...

given $j : \mathcal{U} \to \mathcal{U}'$ $\qquad \prod_{B:\mathcal{U}} j(B) \qquad (j(B) : \mathcal{U}')$

Silly example

$$j : \underset{\underset{\mathcal{U}_{-1}}{\parallel\!|}}{Prop} \to \mathcal{U}_2 ,$$

$$j(\bot) :\equiv \mathcal{U}_0$$
$$j(\top) :\equiv \mathcal{U}_1$$

$\rightsquigarrow$ can have $f : \prod_{(x:Prop)} j(x)$

with $f(\bot) :\equiv \mathbb{N} \quad (: \mathcal{U}_0)$
$f(\top) :\equiv \mathbb{R} \quad (: \mathcal{U}_0 : \mathcal{U}_1)$

even sillier example

$$\Big\{ j(\bot) :\equiv \mathbb{N}, \quad j(\top) :\equiv \mathbb{R}$$
$$\mathbb{N} : \mathcal{U}_0 : \mathcal{U}_2 \qquad \mathbb{R} : \mathcal{U}_0 : \mathcal{U}_2$$

$$f(\bot) :\equiv 15 , \quad f(\top) :\equiv \pi .$$

cont'd define a fct

$$\alpha : \left(\prod_{x:Prop} j(x) \right) \to \mathbb{R} \to \mathbb{R} ,$$

$$\alpha :\equiv \lambda f , \lambda y , f(\bot) \cdot y + f(\top) .$$

Above f gives

$$\alpha(f) : y \mapsto 15y + \pi$$

Silliest example $\quad j(\bot) :\equiv \mathbb{N} \to \mathbb{R} \to \mathbb{R}$

$$j(\top) :\equiv \mathbb{R} \to \mathbb{R} \to \mathbb{R}$$

& $\quad \alpha : \prod_{x:\mathrm{Prop}} j(x) \quad ,$

$$\alpha(\bot) :\equiv \lambda k, \lambda y, k \cdot y \qquad (\mathbb{N} \times \mathbb{R} \xrightarrow{\cdot} \mathbb{R})$$

$$\alpha(\top) :\equiv \lambda x, \lambda y, x \cdot y \qquad (\mathbb{R} \times \mathbb{R} \xrightarrow{+} \mathbb{R})$$

can now look at fcts out of this, etc.

Product types $\quad$ given $A, B \quad$, have

$$A \times B$$

<u>Constr</u> $\quad a : A,\ b : B \rightsquigarrow (a,b) : A \times B$

<u>Elim</u> $\quad f : A \to B \to C \rightsquigarrow \hat{f} : A \times B \to C$

<u>Comp</u> $\quad \hat{f}(a,b) :\equiv f(a)(b)$

<u>Uniq'ss</u> $\quad$ propositional ; (requires identity types)

every element is a pair ,

namely $\quad x : A \times B \Rightarrow$

"refl" : $(pr_1(x), pr_2(x)) =_{A \times B} x$

where $pr_1 : A \times B \to A$ ...

$\pi_1 : A \to B \to A$, $\pi_1 :\equiv \lambda a, \lambda b, a$.

sim'ly for $pr_2 : A \times B \to B$.

more precisely we want a fct

$$\text{uniq} : \prod_{x : A \times B} (pr_1 x, pr_2 x) =_{A \times B} x \quad ,$$

which can be defined (by dependent elim!)

as $\widehat{\text{uni}}$ for

$$\text{uni} : \prod_{a : A} \prod_{b : B} (pr_1 (a,b), pr_2 (a,b)) =_{A \times B} (a,b)$$

given by $\text{uni} :\equiv \lambda a, \lambda b, \text{refl}_{(a,b)}$,

well-typed due to

$$(pr_1 (a,b), pr_2 (a,b)) \overset{\text{judgmental!}}{\equiv} (a,b) \quad ,$$

and $\text{refl}_\varphi : (\varphi = \varphi)$ is a constructor of the identity type.

Note that this doesn't give

$x \equiv (pr_1 x, pr_2 x)$ judgmentally!

Σ-types (dependent product) same formation : family $j : A \to \mathcal{U}$

$\rightsquigarrow$ $\sum_{x:A} j(x)$ or Lean: $\Sigma\, (x : A), j(x)$

... $(a, b) : \Sigma\, j(x)$.

Constr $\quad a : A \,,\; b : j(x) \rightsquigarrow (a,b) : \sum_{x:A}$

Lean: $\quad$ sigma.mk (a, b)

Elim $\quad$ since generalizes products, again by currying, now only **dependent**:

$$f : \prod_{x:A} \big(j(x) \to C \big) \quad \rightsquigarrow \quad \hat{f} : \Big(\sum_{x:A} j(x) \Big) \to C$$

scope understood

$$\left[\begin{array}{l} \text{in coordinates:} \\ (\sim \text{"pointwise constant } j\text{"}) \end{array} \quad \overbrace{\underset{A}{\overset{\cdot\cdot}{x}} \mapsto \big(\underset{j(x)}{\overset{\cdot\cdot}{x'}} \mapsto \underset{C}{\overset{\cdot\cdot}{y}} \big)}^{f} \rightsquigarrow \overbrace{\underset{A \times j(x)}{\overset{\cdot\cdot}{(x, x')}} \mapsto \underset{C}{\overset{\cdot\cdot}{y}}}^{\hat{f}} \right]$$

Comp $\quad$ "same" (!) :

$$\hat{f}\big(\underset{A}{\overset{\cdot\cdot}{a}}, \underset{j(x)}{\overset{\cdot\cdot}{b}} \big) :\equiv f(a)(b)$$

Uniq's $\quad$ analogous to product types.

"Logical" interp of Π, Σ

clear : $\quad$ write "P" instead of "j".

Given $P : A \to \mathcal{U}$, a "(type as) proposition $P(a)$ for each $a : A$, a "witness"

$$w : \prod_{x:A} P(x)$$

is $\quad w \equiv \lambda \underset{A}{\overset{\cdot\cdot}{a}}, \underset{P(x)}{\overset{\cdot\cdot}{w(x)}} \quad$, so a "proof of $P(a)$" for each a.

$\rightsquigarrow$ "$\forall a \in A, P(a)$". Interpreting "$a \in A$" as a proposition, this classical expression can be interpreted as a dependent implication:

not entailment

"$a \in A \Rightarrow P(a)$".

This corresponds to Π-types being dependent functions (dependent internal homs)

not arrows

see week 2

Similarly, $\Sigma \leftrightsquigarrow \exists$

Accordingly, the statement

$$\forall a \in A, \exists b \in B, R(x,y) \Rightarrow \exists f : A \to B, \forall a \in A, R(a, f(a))$$

which'll be our AoC, translates to (this is more transparent in Lean's notation)

$$\left(\prod_{x:A} \sum_{b:B} R(a,b) \right) \xrightarrow[\text{"AoC"}]{} \sum_{f:A\to B} \prod_{a:A} R(a, f(a))$$

a proof

— over to Lean