The HoTT-Book

Chapters 2.12 through 2.13

HoTT-Seminar, SS2017

Today's Topics

1. Coproducts
2. Natural Numbers
3. Exercises

Coproducts

Coproducts

Introduction

- For each type former we have considered so far, there is a way to destruct its elements:
- The projection functions destruct a pair:

$$\mathrm{pr}_1 : A \times B \to A$$
$$\mathrm{pr}_2 : A \times B \to B$$

- Function application destructs a function:

$$\lambda x.\, \lambda f.\, f(x) : \prod_{x:A} (A \to B) \to B$$

- Dependant pairs and functions can be destructed in the same way.

Coproducts

Introduction (cont.)

- As we have seen, the identity type of such types can be characterized straightforwardly, along with all of their higher structure:

$$(x =_{A\times B} y) \simeq (\mathrm{pr}_1(x) =_A \mathrm{pr}_1(y)) \times (\mathrm{pr}_2(x) =_B \mathrm{pr}_2(y))$$

$$(f =_{A\to B} g) \simeq \left(\prod_{x:A} (f(x) =_B g(x))\right)$$

- The idea is to destruct the elements and compare the resulting pieces.
- Again, the same idea works for dependant pairs and functions as well.

Coproducts

Introduction (cont.)

- Now for coproducts, this doesn't work because it is not possible to destruct an element of a coproduct.
- There is no sensible way of defining functions $f : A + B \to A$ or $g : A + B \to B$ that destruct an element of a coproduct.
- That's why we have to think of something else.
- We will now introduce a general method yielding a characterization at least for some particular cases.

- As a coproduct can be seen as a disjoint union, we expect that $A + B$ contains exact copies of A and B disjointly, so that we should have

$$\prod_{a_1,a_2:A} (\mathsf{inl}(a_1) = \mathsf{inl}(a_2)) \simeq (a_1 = a_2)$$
$$\prod_{b_1,b_2:B} (\mathsf{inr}(b_1) = \mathsf{inr}(b_2)) \simeq (b_1 = b_2)$$
$$\prod_{(a:A)} \prod_{(b:B)} (\mathsf{inl}(a) = \mathsf{inr}(b)) \simeq \mathbf{0}.$$

- We prove this as follows:

Coproducts

Proof

- We will define *leq* while *req* is completely analogous.

$$leq : \prod_{(a_0:A)} \prod_{(x:A+B)} ((\mathrm{inl}(a_0) = x) \simeq \mathrm{rec}_{A+B}(\mathcal{U}, \lambda a.\, (a_0 = a), \lambda b.\, \mathbf{0})(x))$$

$$req : \prod_{(b_0:B)} \prod_{(x:A+B)} ((x = \mathrm{inr}(b_0)) \simeq \mathrm{rec}_{A+B}(\mathcal{U}, \lambda a.\, \mathbf{0}, \lambda b.\, (b = b_0))(x))$$

- Then we can use these to get

$$\lambda a_1.\, \lambda a_2.\, leq(a_1)(\mathrm{inl}(a_2)) : \prod_{a_1, a_2:A} (\mathrm{inl}(a_1) = \mathrm{inl}(a_2)) \simeq (a_1 = a_2)$$

$$\lambda b_1.\, \lambda b_2.\, req(b_2)(\mathrm{inr}(b_1)) : \prod_{b_1, b_2:B} (\mathrm{inr}(b_1) = \mathrm{inr}(b_2)) \simeq (b_1 = b_2)$$

$$\lambda a.\, \lambda b.\, leq(a)(\mathrm{inr}(b)) : \prod_{(a:A)} \prod_{(b:B)} (\mathrm{inl}(a) = \mathrm{inr}(b)) \simeq \mathbf{0}$$

$$\lambda a.\, \lambda b.\, req(b)(\mathrm{inl}(a)) : \prod_{(a:A)} \prod_{(b:B)} (\mathrm{inl}(a) = \mathrm{inr}(b)) \simeq \mathbf{0}$$

Coproducts

Proof (cont.)

- The idea is to fix one endpoint on the left side of the equivalence while we encode the right side by a type family.
- This motivates a naming of the righthand side type family as:

$$\mathrm{code}_{a_0} :\equiv \mathrm{rec}_{A+B}(\mathcal{U}, \lambda a.\, (a_0 = a), \lambda b.\, \mathbf{0})$$

- Now our goal looks as follows:

$$\mathit{leq} : \prod_{(a_0:A)} \prod_{(x:A+B)} ((\mathrm{inl}(a_0) = x) \simeq \mathrm{code}_{a_0}(x))$$

- We now define the two functions of the equivalence. Following our naming scheme we call them:

$$\mathrm{encode} : \prod_{(a_0:A)} \prod_{(x:A+B)} (\mathrm{inl}(a_0) = x) \to \mathrm{code}_{a_0}(x)$$

$$\mathrm{decode} : \prod_{(a_0:A)} \prod_{(x:A+B)} \mathrm{code}_{a_0}(x) \to (\mathrm{inl}(a_0) = x)$$

Coproducts

Proof (cont.)

- We can define encode by transporting reflexivity along p:

$$\begin{aligned}\mathsf{encode} &:\equiv \lambda a_0.\, \lambda x.\, \lambda p.\, \mathsf{transport}^{\mathsf{code}_{a_0}}(p, \mathsf{refl}_{a_0}) \\ &: \prod_{(a_0:A)} \prod_{(x:A+B)} (\mathsf{inl}(a_0) = x) \to \mathsf{code}_{a_0}(x)\end{aligned}$$

- Note that $\mathsf{refl}_{a_0} : \mathsf{code}_{a_0}(\mathsf{inl}(a_0)) \equiv (a_0 = a_0)$
- We can define decode by induction on x:
- If $x \equiv \mathsf{inl}(a)$ then $\mathsf{code}_{a_0}(x) \equiv (a_0 = a)$ and we've got

$$\lambda c.\, \mathsf{ap}_{\mathsf{inl}}(c) : \mathsf{code}_{a_0}(x) \to (\mathsf{inl}(a_0) = \mathsf{inl}(a)).$$

- If $x \equiv \mathsf{inr}(b)$ then $\mathsf{code}_{a_0}(x) \equiv \mathbf{0}$ and we've got

$$\lambda c.\, \mathsf{rec}_{\mathbf{0}}(\mathsf{inl}(a_0) = \mathsf{inl}(a), c) : \mathsf{code}_{a_0}(x) \to (\mathsf{inl}(a_0) = \mathsf{inl}(a)).$$

Coproducts

Proof (cont.)

- Next, we show that $\mathsf{encode}_{a_0}(x)$ and $\mathsf{decode}_{a_0}(x)$ are quasi-inverses for all x.
- We start with proving

$$\prod_{(a_0:A)} \prod_{(x:A+B)} \prod_{(p:\mathsf{inl}(a_0)=x)} \mathsf{decode}_{a_0}(x, \mathsf{encode}_{a_0}(x,p)) = p$$

by path induction on p assuming $x \equiv \mathsf{inl}(a_0)$ and $p \equiv \mathsf{refl}_{\mathsf{inl}(a_0)}$:

$$\begin{aligned}
&\mathsf{decode}_{a_0}(x, \mathsf{encode}_{a_0}(x,p)) \\
&\equiv \mathsf{decode}_{a_0}(\mathsf{inl}(a_0), \mathsf{encode}_{a_0}(\mathsf{inl}(a_0), \mathsf{refl}_{\mathsf{inl}(a_0)})) \\
&\equiv \mathsf{ap}_{\mathsf{inl}}(\mathsf{transport}^{\mathsf{code}_{a_0}}(\mathsf{refl}_{\mathsf{inl}(a_0)}, \mathsf{refl}_{a_0})) \\
&\equiv \mathsf{ap}_{\mathsf{inl}}(\mathsf{refl}_{a_0}) \\
&\equiv \mathsf{refl}_{\mathsf{inl}(a_0)} \\
&\equiv p.
\end{aligned}$$

Coproducts

Proof (cont.)

- We proceed with proving

$$\prod_{(a_0:A)} \prod_{(x:A+B)} \prod_{(c:\mathrm{code}_{a_0}(x))} \mathrm{encode}_{a_0}(x, \mathrm{decode}_{a_0}(x, c)) = c$$

  by induction over x:
- If $x \equiv \mathrm{inl}(a)$ then $c : \mathrm{code}_{a_0}(x) \equiv (a_0 = a)$ and we proceed with path induction on c assuming $a \equiv a_0$ and $c \equiv \mathrm{refl}_{a_0}$:

$$\begin{aligned}
&\mathrm{encode}_{a_0}(x, \mathrm{decode}_{a_0}(x, c)) \\
&\equiv \mathrm{encode}_{a_0}(\mathrm{inl}(a_0), \mathrm{decode}_{a_0}(\mathrm{inl}(a_0), \mathrm{refl}_{a_0})) \\
&\equiv \mathrm{transport}^{\mathrm{code}_{a_0}}(\mathrm{ap}_{\mathrm{inl}}(\mathrm{refl}_{a_0}), \mathrm{refl}_{a_0}) \\
&\equiv \mathrm{transport}^{\mathrm{code}_{a_0}}(\mathrm{refl}_{\mathrm{inl}(a_0)}, \mathrm{refl}_{a_0}) \\
&\equiv \mathrm{refl}_{a_0} \\
&\equiv c
\end{aligned}$$

Coproducts

Proof (cont.)

- If $x \equiv \mathsf{inr}(b)$ then $\mathsf{code}_{a_0}(x) \equiv \mathbf{0}$ and we've got

$$\begin{aligned} &\mathsf{ind}_{\mathbf{0}}(\lambda z.\, \mathsf{encode}_{a_0}(x, \mathsf{decode}_{a_0}(x, c)) = c) \\ &: \prod_{c:\mathsf{code}_{a_0}(x)} \mathsf{encode}_{a_0}(x, \mathsf{decode}_{a_0}(x, c)) = c \end{aligned}$$

□

- This method to prove the equivalence is called the encode-decode method.

Coproducts

Implications

- In particular, this theorem implies that we have the following functions:

$$\lambda a_1.\, \lambda a_2.\, \mathrm{encode}_{a_1}(\mathrm{inl}(a_2)) : \prod_{a_1, a_2 : A} (\mathrm{inl}(a_1) = \mathrm{inl}(a_2)) \to (a_1 = a_2)$$

$$\lambda a.\, \lambda b.\, \mathrm{encode}_{a}(\mathrm{inr}(b)) : \prod_{(a:A)} \prod_{(b:B)} (\mathrm{inl}(a) = \mathrm{inr}(b)) \to \mathbf{0}.$$

- The traditional reading of the first of these is just injectivity of inl.
- The full homotopical statement however, gives more information:
- The types $\mathrm{inl}(a_1) = \mathrm{inl}(a_2)$ and $a_1 = a_2$ are actually equivalent, as are $\mathrm{inr}(b_1) = \mathrm{inr}(b_2)$ and $b_1 = b_2$.
- The second one states that $\mathrm{inl}(a) \neq \mathrm{inr}(b)$ for any $a : A$ and $b : B$, which means the images of inl and inr are disjoint.
- In particular, since $\mathbf{2} \equiv \mathbf{1} + \mathbf{1}$, we have $0_{\mathbf{2}} \equiv \mathrm{inl}(\star) \neq \mathrm{inr}(\star) \equiv 1_{\mathbf{2}}$.

- As usual, we can also characterize the action of transport in coproduct types:
- Given a type X, a path $p : x_1 =_X x_2$, and type families $A, B : X \to \mathcal{U}$, we have

$$\mathsf{transport}^{A+B}(p, \mathsf{inl}(a)) = \mathsf{inl}(\mathsf{transport}^{A}(p, a)),$$
$$\mathsf{transport}^{A+B}(p, \mathsf{inr}(b)) = \mathsf{inr}(\mathsf{transport}^{B}(p, b)),$$

- As before, $A + B$ denotes abusively the type family $\lambda x.\, A(x) + B(x)$.
- The proof is an easy path induction.

Natural Numbers

- Just as we did for coproducts, we now use the encode-decode method to characterize the path space of the natural numbers:
- In this case, rather than fixing one endpoint, we characterize the two-sided path space all at once.
- Thus, our code is a type family

$$\mathrm{code} : \mathbb{N} \to \mathbb{N} \to \mathcal{U},$$

defined by double recursion over $\mathbb{N}$ as follows:

$$\begin{aligned} \mathrm{code}(0,0) &:\equiv \mathbf{1} \\ \mathrm{code}(\mathrm{succ}(m),0) &:\equiv \mathbf{0} \\ \mathrm{code}(0,\mathrm{succ}(n)) &:\equiv \mathbf{0} \\ \mathrm{code}(\mathrm{succ}(m),\mathrm{succ}(n)) &:\equiv \mathrm{code}(m,n). \end{aligned}$$

Natural Numbers

Proof

- Now our goal is:

$$\prod_{m,n:\mathbb{N}} (m = n) \simeq \mathsf{code}(m, n)$$

- We begin by defining a dependent function $r : \prod_{(n:\mathbb{N})} \mathsf{code}(n, n)$ by recursion:

$$r(0) :\equiv \star$$
$$r(\mathsf{succ}(n)) :\equiv r(n).$$

- Now, we can define

$$\mathsf{encode} :\equiv \lambda m.\, \lambda n.\, \lambda p.\, \mathsf{transport}^{\lambda n.\, \mathsf{code}(m,n)}(p, r(m))$$
$$: \prod_{m,n:\mathbb{N}} (m = n) \to \mathsf{code}(m, n)$$

Natural Numbers

Proof (cont.)

- Next, we define

$$\text{decode} : \prod_{m,n:\mathbb{N}} \text{code}(m,n) \to (m = n)$$

  by double induction on m, n:
- When m and n are both 0, we can define

$$\text{decode}(0,0) :\equiv \lambda c.\, \text{refl}_0 : \mathbf{1} \to (0 = 0) \equiv \text{code}(m,n) \to (m = n).$$

- When m is a successor and n is 0 or vice versa, we have $\text{code}(m,n) \equiv \mathbf{0}$, so the recursor for $\mathbf{0}$ suffices.
- Finally, when both m and n are successors, we can define

$$\begin{aligned}\text{decode}(\text{succ}(m),\text{succ}(n)) &:\equiv \text{ap}_{\text{succ}}(\text{decode}(m,n)) \\ &: \text{code}(\text{succ}(m),\text{succ}(n)) \to (\text{succ}(m) = \text{succ}(n))\end{aligned}$$

Natural Numbers

Proof (cont.)

- Next we show that $\mathrm{encode}(m, n)$ and $\mathrm{decode}(m, n)$ are quasi-inverses for all m, n:
- We start with proving

$$\prod_{(m,n:\mathbb{N})} \prod_{(p:m=n)} \mathrm{decode}(m, n, \mathrm{encode}(m, n, p)) = p$$

by induction on p: It suffices to show

$$\begin{aligned}
&\mathrm{decode}(n, n, \mathrm{encode}(n, n, \mathrm{refl}_n)) = \mathrm{refl}_n \\
&\equiv \mathrm{decode}(n, n, \mathrm{transport}^{\lambda n.\, \mathrm{code}(m,n)}(\mathrm{refl}_n, r(n))) = \mathrm{refl}_n \\
&\equiv \mathrm{decode}(n, n, r(n)) = \mathrm{refl}_n
\end{aligned}$$

- We can prove
$$\mathsf{decode}(n, n, r(n)) = \mathsf{refl}_n$$
by induction on n:
- If $n \equiv 0$, then $\mathsf{decode}(0, 0, r(0)) \equiv \mathsf{refl}_0$ by definition of decode.
- And in the case of a successor, by the inductive hypothesis we have:

$$\begin{aligned}
&\mathsf{decode}(\mathsf{succ}(n), \mathsf{succ}(n), r(\mathsf{succ}(n))) \\
&\equiv \mathsf{ap}_{\mathsf{succ}}(\mathsf{decode}(n, n, r(n))) \\
&= \mathsf{ap}_{\mathsf{succ}}(\mathsf{refl}_n) \\
&\equiv \mathsf{refl}_{\mathsf{succ}(n)}
\end{aligned}$$

Natural Numbers

Proof (cont.)

- We proceed with proving

$$\prod_{(m,n:\mathbb{N})} \prod_{(c:code(m,n))} \text{encode}(m, n, \text{decode}(m, n, c)) = c$$

  by double induction on m and n:
- If both are 0, then we have

$$\prod_{c:code(0,0)} \text{encode}(0, 0, \text{decode}(0, 0, c)) = c$$
$$\equiv \prod_{c:\mathbf{1}} \text{transport}^{\lambda n.\, \text{code}(0,n)}(\text{refl}_0, r(0)) = c$$
$$\equiv \prod_{c:\mathbf{1}} \star = c$$

  which holds because every inhabitant of $\mathbf{1}$ is equal to $\star$.
- If m is 0 but n is a successor, or vice versa, then $c : \mathbf{0}$, so we're done.

- Finally, in the case of two successors, we have

$$\begin{aligned}
&\mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), \mathsf{decode}(\mathsf{succ}(m), \mathsf{succ}(n), c)) \\
&\equiv \mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), \mathsf{ap}_{\mathsf{succ}}(\mathsf{decode}(m, n, c))) \\
&\equiv \mathsf{transport}^{\mathsf{code}(\lambda n.\, \mathsf{succ}(m), n)}(\mathsf{ap}_{\mathsf{succ}}(\mathsf{decode}(m, n, c)), r(\mathsf{succ}(m))) \\
&= \mathsf{transport}^{\lambda n.\, \mathsf{code}(\mathsf{succ}(m), \mathsf{succ}(n))}(\mathsf{decode}(m, n, c), r(\mathsf{succ}(m))) \\
&\equiv \mathsf{transport}^{\lambda n.\, \mathsf{code}(m, n)}(\mathsf{decode}(m, n, c), r(m)) \\
&\equiv \mathsf{encode}(m, n, \mathsf{decode}(m, n, c)) \\
&= c
\end{aligned}$$

using Lemma 2.3.10: $\mathsf{transport}^{P \circ f}(p, u) = \mathsf{transport}^{P}(\mathsf{ap}_f(p), u)$ and the inductive hypothesis. □

- In particular, we have

$$\mathsf{encode}(\mathsf{succ}(m), 0) : (\mathsf{succ}(m) = 0) \to \mathbf{0}$$

which shows that “0 is not the successor of any natural number”.

- We also have

$$\lambda m.\, \lambda n.\, \lambda p.\, \mathsf{decode}(m, n, \mathsf{encode}(\mathsf{succ}(m), \mathsf{succ}(n), p)) : \prod_{m,n:\mathbb{N}} (\mathsf{succ}(m) = \mathsf{succ}(n)) \to (m = n)$$

which shows that the function succ is injective.

Exercises

Exercises

Exercise 2.9

- Prove that coproducts have the expected universal property

$$(A + B \to X) \simeq (A \to X) \times (B \to X).$$

- First, we define functions f and its inverse:

$$f :\equiv \lambda g.\, (g \circ \mathsf{inl}, g \circ \mathsf{inr}) : (A + B \to X) \to (A \to X) \times (B \to X)$$
$$\lambda p.\, \mathsf{rec}_{A+B}(X, \mathsf{pr}_1(p), \mathsf{pr}_2(p)) : (A \to X) \times (B \to X) \to (A + B \to X)$$

Exercises

Exercise 2.9 (cont.)

- Then, we show that they are quasi-inverses. We start with

$$\prod_{g:A+B\to X} f^{-1}(f(g)) = g.$$

- We have

$$\begin{aligned} f^{-1}(f(g)) &= g \\ &\equiv f^{-1}((g\circ \mathsf{inl}, g\circ \mathsf{inr})) = g \\ &\equiv \mathsf{rec}_{A+B}(X, g\circ \mathsf{inl}, g\circ \mathsf{inr}) = g \end{aligned}$$

- Now, by function extensionality it suffices to prove

$$\prod_{x:A+B} \mathsf{rec}_{A+B}(X, g\circ \mathsf{inl}, g\circ \mathsf{inr})(x) = g(x)$$

which is an easy induction over x as we know that:

$$g(\mathsf{inl}(a)) = g(\mathsf{inl}(a)) \qquad g(\mathsf{inr}(b)) = g(\mathsf{inr}(b))$$

Exercises

Exercise 2.9 (cont.)

- Finally, we prove

$$\prod_{p:(A\to X)\times(B\to X)} f(f^{-1}(p)) = p.$$

- We have $f(f^{-1}(p)) \equiv f(\mathrm{rec}_{A+B}(X, \mathrm{pr}_1(p), \mathrm{pr}_2(p)))$, and we can split up $f(\mathrm{rec}_{A+B}(X, \mathrm{pr}_1(p), \mathrm{pr}_2(p))) = p$ into

$$\mathrm{rec}_{A+B}(X, \mathrm{pr}_1(p), \mathrm{pr}_2(p)) \circ \mathrm{inl} = \mathrm{pr}_1(p)$$
$$\mathrm{rec}_{A+B}(X, \mathrm{pr}_1(p), \mathrm{pr}_2(p)) \circ \mathrm{inr} = \mathrm{pr}_2(p)$$

and then show both by function extensionality

$$\lambda a.\, \mathrm{refl}_{\mathrm{pr}_1(p)(a)} : \prod_{a:A} \mathrm{rec}_{A+B}(X, \mathrm{pr}_1(p), \mathrm{pr}_2(p))(\mathrm{inl}(a)) = \mathrm{pr}_1(p)(a)$$
$$\lambda b.\, \mathrm{refl}_{\mathrm{pr}_2(p)(b)} : \prod_{b:B} \mathrm{rec}_{A+B}(X, \mathrm{pr}_1(p), \mathrm{pr}_2(p))(\mathrm{inr}(b)) = \mathrm{pr}_2(p)(b)$$

since both equations hold judgementally.

- So we've got

$$\mathsf{funext}(\lambda a.\, \mathsf{refl}_{\mathsf{pr}_1(p)(a)}) : \mathsf{rec}_{A+B}(X, \mathsf{pr}_1(p), \mathsf{pr}_2(p)) \circ \mathsf{inl} = \mathsf{pr}_1(p)$$
$$\mathsf{funext}(\lambda b.\, \mathsf{refl}_{\mathsf{pr}_2(p)(b)}) : \mathsf{rec}_{A+B}(X, \mathsf{pr}_1(p), \mathsf{pr}_2(p)) \circ \mathsf{inr} = \mathsf{pr}_2(p)$$

and thus

$$\mathsf{pair}^{=}((\mathsf{funext}(\lambda a.\, \mathsf{refl}_{\mathsf{pr}_1(p)(a)}), \mathsf{funext}(\lambda b.\, \mathsf{refl}_{\mathsf{pr}_2(p)(b)}))) : f(\mathsf{rec}_{A+B}(X, \mathsf{pr}_1(p), \mathsf{pr}_2(p))) = p$$

□

Exercises

Exercise 2.17 (iii) for Coproducts

- Show that if $A \simeq A'$ and $B \simeq B'$, then $(A + B) \simeq (A' + B')$.
- So, given equivalences $g : A \simeq A'$ and $h : B \simeq B'$ we define

$$f :\equiv \mathrm{rec}_{A+B}(A' + B', \mathrm{inl} \circ g, \mathrm{inr} \circ h) : (A + B) \to (A' + B')$$
$$f^{-1} :\equiv \mathrm{rec}_{A'+B'}(A + B, \mathrm{inl} \circ g^{-1}, \mathrm{inr} \circ h^{-1}) : (A' + B') \to (A + B)$$

and show that they are quasi-inverses.

Exercises

Exercise 2.17 (iii) for Coproducts (cont.)

- We start with proving

$$\prod_{x:A+B} f^{-1}(f(x)) = x$$

by induction on x:

- If $x \equiv \mathsf{inl}(a)$, then we have

$$\begin{aligned} & f^{-1}(f(x)) \\ & \equiv f^{-1}(f(\mathsf{inl}(a))) \\ & \equiv f^{-1}(\mathsf{inl}(g(a))) \\ & \equiv \mathsf{inl}(g^{-1}(g(a))) \\ & = \mathsf{inl}(a) \\ & \equiv x \end{aligned}$$

- If $x \equiv \mathsf{inr}(b)$, then we have

$$\begin{aligned} & f^{-1}(f(x)) \\ & \equiv f^{-1}(f(\mathsf{inr}(b))) \\ & \equiv f^{-1}(\mathsf{inr}(h(b))) \\ & \equiv \mathsf{inr}(h^{-1}(h(b))) \\ & = \mathsf{inr}(b) \\ & \equiv x \end{aligned}$$

Exercises

Exercise 2.17 (iii) for Coproducts (cont.)

- And finally, we've got

$$\prod_{x:A'+B'} f(f^{-1}(x)) = x$$

by induction on x:

- If $x \equiv \mathsf{inl}(a)$, then we have

$$\begin{aligned} & f(f^{-1}(x)) \\ & \equiv f(f^{-1}(\mathsf{inl}(a))) \\ & \equiv f(\mathsf{inl}(g^{-1}(a))) \\ & \equiv \mathsf{inl}(g(g^{-1}(a))) \\ & = \mathsf{inl}(a) \\ & \equiv x. \end{aligned}$$

- If $x \equiv \mathsf{inr}(b)$, then we have

$$\begin{aligned} & f(f^{-1}(x)) \\ & \equiv f(f^{-1}(\mathsf{inr}(b))) \\ & \equiv f(\mathsf{inr}(h^{-1}(b))) \\ & \equiv \mathsf{inr}(h(h^{-1}(b))) \\ & = \mathsf{inr}(b) \\ & \equiv x. \end{aligned}$$

□

- Let's prove that for any type $A : \mathcal{U}$ we've got

$$A + A = \mathbf{2} \times A.$$

- First, by univalence it suffices to prove

$$A + A \simeq \mathbf{2} \times A.$$

which we do by defining an equivalence f

$$f :\equiv \lambda x.\, \mathrm{rec}_{A+A}(\mathbf{2} \times A, \lambda a.\, (0_{\mathbf{2}}, a), \lambda a.\, (1_{\mathbf{2}}, a))(x) : A + A \to \mathbf{2} \times A$$
$$\lambda p.\, \mathrm{rec}_{\mathbf{2}}(A + A, \mathrm{inl}(\mathrm{pr}_2(p)), \mathrm{inr}(\mathrm{pr}_2(p)))(\mathrm{pr}_1(p)) : \mathbf{2} \times A \to A + A$$

Exercises

Some Funny Arithmetic (cont.)

- Now, we show that f is an equivalence by first proving

$$\prod_{x:A+A} f^{-1}(f(x)) = x$$

by induction on x:

- If $x \equiv \mathsf{inl}(a)$, then we have

$$\begin{aligned} & f^{-1}(f(x)) \\ & \equiv f^{-1}(f(\mathsf{inl}(a))) \\ & \equiv f^{-1}((0_{\mathbf{2}}, a)) \\ & \equiv \mathsf{inl}(a) \\ & \equiv x. \end{aligned}$$

- If $x \equiv \mathsf{inr}(a)$, then we have

$$\begin{aligned} & f^{-1}(f(x)) \\ & \equiv f^{-1}(f(\mathsf{inr}(a))) \\ & \equiv f^{-1}((1_{\mathbf{2}}, a)) \\ & \equiv \mathsf{inr}(a) \\ & \equiv x. \end{aligned}$$

- Finally, we show that

$$\prod_{p:\mathbf{2}\times A} f(f^{-1}(p)) = p$$

by induction on p; so it suffices to prove

$$\prod_{(b:\mathbf{2})} \prod_{(a:A)} f(f^{-1}((b,a))) = (b,a)$$

by induction on b:

- If $b \equiv 0_{\mathbf{2}}$, then we have

$$\begin{aligned} & f(f^{-1}((0_{\mathbf{2}}, a))) \\ & \equiv f(\mathsf{inl}(a)) \\ & \equiv (0_{\mathbf{2}}, a) \end{aligned}$$

- If $b \equiv 1_{\mathbf{2}}$, then we have

$$\begin{aligned} & f(f^{-1}((1_{\mathbf{2}}, a))) \\ & \equiv f(\mathsf{inr}(a)) \\ & \equiv (1_{\mathbf{2}}, a) \end{aligned}$$

□

The End

Questions?