1 Introduction

The point of homotopy type theory is to create a logical system in which isomorphism and equality are "the same". Of course, to say this we must furthermore ask what kind of "the same" is this? Should we say that isomorphism and equality are equal or are they isomorphic and what does that mean? Our theory will in fact satisfy the *univalence axiom*: the natural map

$$\text{idtoeqv} : (A =_{\mathcal{U}} B) \to (A \simeq B)$$

is an equivalence. We want to understand what this means.

The total loop space:

$$\Sigma_{x \in A}(x =_A x)$$

The total path space:

$$\Sigma_{x,y \in A}(x =_A y)$$

2 Chapter 1

2.1 Exercises

2.1.1 Exercise 1.9

Exercise 2.1.1. Define the type family $\text{Fin} : \mathbb{N} \to \mathcal{U}$ and dependent function $\text{fmax} : \prod_{\text{n}:\mathbb{N}} \text{Fin}(\text{succ}(\text{n}))$.

We use the recursive type constructor,

$$\text{Fin} = \text{rec}_{\mathbb{N}}(\mathcal{U}, \mathbf{0}, \lambda \text{n}.\lambda \text{A}.(\text{A} + \mathbf{1}))$$

This satisfies,

$$\text{Fin}(0) :\equiv \mathbf{0} \quad \text{Fin}(\text{succ}(\text{n})) :\equiv \text{Fin}(\text{n}) + \mathbf{1}$$

then the max function also has an inductive construction using the type family Fin,

$$\text{fmax} = \text{ind}_{\mathbb{N}}(\text{Fin} \circ \text{succ}, \star, \text{c}_\text{s})$$

where,

$$c_s : \prod_{n:\mathbb{N}} \text{Fin}(\text{succ}(\text{n})) \to \text{Fin}(\text{succ}(\text{succ}(\text{n})))$$

is the function,

$$c_s(n, q) :\equiv \text{inr}(\star)$$

Which satisfies the properties,

$$\text{fmax}(0) \equiv \star : \mathbf{1} \quad \text{fmax}(\text{succ}(\text{n})) \equiv \text{inr}(\star) : \text{Fin}(\text{succ}(\text{n})) + \mathbf{1}$$

2.1.2 Exercise 1.10

Exercise 2.1.2. Show that the Ackermann function $\mathrm{ack} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ is definable using only $\mathrm{rec}_{\mathbb{N}}$ satisfying the following equations:

$$\begin{aligned} \mathrm{ack}(0, n) &\equiv \mathrm{succ}(n) \\ \mathrm{ack}(\mathrm{succ}(m), 0) &\equiv \mathrm{ack}(m, 1) \\ \mathrm{ack}(\mathrm{succ}(m), \mathrm{succ}(n)) &\equiv \mathrm{ack}(m, \mathrm{ack}(\mathrm{succ}(m), n)) \end{aligned}$$

We define the following,

$$\mathrm{ack} :\equiv \mathrm{rec}_{\mathbb{N}}(\mathbb{N} \to \mathbb{N}, \mathrm{succ}, \lambda(m : \mathbb{N}).\lambda(g : \mathbb{N} \to \mathbb{N}).\lambda(n : \mathbb{N}).c_s(n, g, m))$$

where,

$$c_s :\equiv \mathrm{rec}_{\mathbb{N}}((\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N}), \lambda(g : \mathbb{N} \to \mathbb{N}).\lambda(m : \mathbb{N}).g(1), u)$$

where,

$$u :\equiv \lambda(n : \mathbb{N}).\lambda(c : (\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})).\lambda(g : \mathbb{N} \to \mathbb{N}).\lambda(m : \mathbb{N}).g(c(g, m))$$

Then we have the following,

$$c_s(0) \equiv \lambda(g : \mathbb{N} \to \mathbb{N}).\lambda(m : \mathbb{N}).g(1)$$

and

$$c_s(\mathrm{succ}(n)) \equiv u(n, c_s(n)) \equiv \lambda(g : \mathbb{N} \to \mathbb{N}).\lambda(m : \mathbb{N}).g(c_s(n, g, m))$$

Therefore,

$$\mathrm{ack}(0) \equiv \mathrm{succ} \quad \text{meaning } \mathrm{ack}(0, n) \equiv \mathrm{succ}(n)$$

and also,

$$\mathrm{ack}(\mathrm{succ}(m)) \equiv [\lambda(m : \mathbb{N}).\lambda(g : \mathbb{N} \to \mathbb{N}).\lambda(n : \mathbb{N}).c_s(n, g, m)](m, \mathrm{ack}(m)) \equiv \lambda(n : \mathbb{N}).c_s(n, \mathrm{ack}(m), m)$$

This means that,

$$\mathrm{ack}(\mathrm{succ}(m), n) \equiv c_s(n, \mathrm{ack}(m), m)$$

so in particular,

$$\mathrm{ack}(\mathrm{succ}(m), 0) \equiv c_s(0, \mathrm{ack}(m), m) \equiv \mathrm{ack}(m, 1)$$

and also,

$$\mathrm{ack}(\mathrm{succ}(m), \mathrm{succ}(n)) \equiv c_s(\mathrm{succ}(n), \mathrm{ack}(m), m) \equiv \mathrm{ack}(m, c_s(n, \mathrm{ack}(m), m)) \equiv \mathrm{ack}(m, \mathrm{ack}(\mathrm{succ}(m), n))$$

2.1.3 Exercise 1.11

Exercise 2.1.3. Show that for any type A we have $\neg\neg\neg A \to \neg A$.

We need to show that the type $(((A \to \mathbf{0}) \to \mathbf{0}) \to \mathbf{0}) \to (A \to \mathbf{0})$ is inhabited. Indeed consider,

$$\lambda(f : \neg\neg\neg A).\lambda(a : A).f(\lambda(g : \neg A).g(a))$$

2.1.4 Exercise 1.12

Exercise 2.1.4. Using the propositions as types interpretation, derive the following tautologies,

(a) If A then (if B then A).

(b) If A then not (not A).

(c) If (not A or not B) then not (A and B).

(a) $(\lambda(a : A).\lambda(b : B).a) : (A \to B \to A)$

(b) $(\lambda(a : A).\lambda(p : \neg A).p(a)) : (A \to (A \to \mathbf{0}) \to \mathbf{0})$

(c) $\mathrm{rec}_{\neg A + \neg B}(\neg(A \times B), [\lambda(p : \neg A).\lambda(f : A \times B).p \circ \mathrm{pr}_1(f)], [\lambda(p : \neg B).\lambda(f : A \times B).p \circ \mathrm{pr}_2(f)]) : (\neg A + \neg B) \to \neg(A \times B)$

2.1.5 Exercise 1.15

Exercise 2.1.5. Show that the indiscernibility of identicals follows from path induction.

We want to show show that for any family of types $C : A \to \mathcal{U}$ there is,

$$f : \prod_{x,y:A} \prod_{p:x=_A y} C(x) \to C(y)$$

such that,

$$f(x, x, \mathrm{refl}_x) :\equiv \mathrm{id}_{C(x)}$$

We apply path induction to the family of types,

$$D(x, y, p) :\equiv C(x) \to C(y)$$

and our base case is,

$$\prod_{x:A} D(x, x, \mathrm{refl}_x) \equiv \prod_{x:A} C(x) \to C(x)$$

which is inhabited by id_C. Then we set,

$$f :\equiv \mathrm{ind}_{=_A}(D, \mathrm{id}_C)$$

which satisfies,

$$f(x, x, \mathrm{refl}_x) :\equiv \mathrm{id}_{C(x)}$$

3 4 Equivalences

3.0.1 Exercise 4.6

For $A, B : \mathcal{U}$ define

$$\mathrm{idtoqinv}_{A,B} : (A =_{\mathcal{U}} B) \to \sum_{f:A\to B} \mathrm{quiv}(f)$$

by path induction. Explicitly, consider

$$C : \prod_{A,B:\mathcal{U}} A =_{\mathcal{U}} B \to \mathcal{U}$$

defined by

$$C(A, B, p) :\equiv \sum_{f:A\to B} \mathrm{quiv}(f)$$

and notice that we have

$$c : \prod_{A:\mathcal{U}} C(A, A, \mathrm{refl}_A)$$

defined by,

$$c(A) = (\mathrm{id}_A, \mathrm{id}_A)$$

therefore, there is an element,

$$\mathrm{idtoqinv} : \prod_{A,B:\mathcal{U}} \prod_{p:A=_{\mathcal{U}}B} \sum_{f:A\to B} \mathrm{quiv}(f)$$

which is the same as an a term

$$\mathrm{idtoqinv} : \prod_{A,B:\mathcal{U}} (A =_{\mathcal{U}} B) \to \sum_{f:A\to B} \mathrm{quiv}(f)$$

Let **qinv-univalence** denote the modified form of the univalence axiom that asserts that idtoqinv has a quasi-inverse.

(a) Show that **qinv-univalence** suffices in the proof of function extension in §4.9.

(b) Show that **qinv-univalence** suffices to prove Theorem 4.1.3.

(c) Show that **qinv-univalence** is inconsistent.

4 Examples of Using Univalence

Exercise 4.0.1. 17.1(a) of this book Show that the type

$$C :\equiv \sum_{A:\mathcal{U}} \mathrm{isContr}(A)$$

is contractible.

It suffices to show that C is inhabited and for all $A, B : C$ exhibit an element of type $A =_C B$. First note that $1 : C$ or more formally 1 along with the inductive proof of contractibility. Then for $(A, t_A), (B, t_B) : C$ we need to exhibt elements $p : A =_{\mathcal{U}} B$ and $\gamma : (p_*(t_A) =_{\mathrm{isContr}(B)} t_B)$. However, $\mathrm{isContr}(B)$ is a mere proposition so the second is free given the first. However, we are given $t_A : \mathrm{isContr}(A)$ and $t_B : \mathrm{isContr}(B)$ so there are elements of $A \simeq 1$ and $B \simeq 1$ which compose to give $A \simeq B$ and hence by the inverse univalence map $A =_{\mathcal{U}} B$.