5 Homotopy Type Theory

5.1 Proof relevant equality

If we are only use $\mathrm{E}^{=}$ aka J we cannot in general prove that there is only one proof of equality, but what can we prove? It turns out that we can indeed verify some equalities:

$$\begin{aligned}
\mathrm{trans}\, p\, \mathrm{refl} &= p \\
\mathrm{trans}\, \mathrm{refl}\, p &= p \\
\mathrm{trans}\, (\mathrm{trans}\, p\, q)\, r &= \mathrm{trans}\, p\, (\mathrm{trans}\, q\, r) \\
\mathrm{trans}\, p\, (\mathrm{sym}\, p) &= \mathrm{refl} \\
\mathrm{trans}\, (\mathrm{sym}\, p)\, p &= \mathrm{refl}
\end{aligned}$$

where $p : a =_A b$, $q : b =_A c$ and $r : c =_A d$.

It is easy to verify these equalities using only J because all the quantifications are over arbitrary $p : x = y$ and there is no repetition of variables.

Exercise 18 *Explicitly construct the proofs using J.*

A structure with these properties is called a groupoid. A groupoid is a category where every morphism is an isomorphism. Here the objects are the elements of the type A, given $a, b : A$ the homset (actually a type) is $a =_A b$, composition is trans, identity is refl and sym assigns to every morphism its inverse.

We can go further an observe that resp also satisfies some useful equalities:

$$\begin{aligned}
\mathrm{resp}\, f\, \mathrm{refl} &= \mathrm{refl} \\
\mathrm{resp}\, f\, (\mathrm{trans}\, p\, q) &= \mathrm{trans}\, (\mathrm{resp}\, f\, p)\, (\mathrm{resp}\, f\, q)
\end{aligned}$$

where $f : A \to B, p : a =_A b, q : b =_A c$.

In categorical terms this means that f is a functor: its effect on objects $a : A$ is $f\, a : B$ and its effect on morphism $p : a =_A b$ is $\mathrm{resp}\, f\, p : f\, a =_B f\, b$.

Exercise 19 *Why don't we prove that* resp f *also preserves symmetries?*

$$\mathrm{resp}\, f\, (\mathrm{sym} p) = \mathrm{sym}\, (\mathrm{resp}\, f\, p)$$

Indeed, the idea of Streicher's and Hofmann's proof is to turn this around and to show that we can generally interpret types as groupoids where the equality type corresponds to the homset. Interestingly we can also interpret J in this setting but clearly we cannot interpret K because it forces the groupoid to be trivial, i.e. to be an equivalence relation. However, in the moment there is no construction which generates non-trivial groupoids, but this will change once we have the univalence principle or if we introduce Higher Inductive Types (HITs).

However, groupoids are not the whole story. There is no reason to assume that the next level of equalities, i.e. the equality of the equality of equality proofs

is trivial. We need to add some further laws, so called coherence laws, which are well known and we end up with a structure which is called a 2-Groupoid (in particular all the homsets are groupoids). But the story doesn't finish here we can go on forever. The structure we are looking for is called a weak ω-groupoid. Alas, it is not very easy to write down what this is precisely. Luckily, homotopy theoreticians have already looked at this problem and they have a definition: a weak ω-groupoid is a Kan complex, that is a simplicial set with all Kan fillers. This is what Voevodsky has used in his homotopical model of Homotopy Type Theory. However, it was noted that this construction uses classical principles, i.e. the axiom of choice. Thierry Coquand and his team have now formulated a constructive alternative which is based on cubical sets.

5.2 What is a proposition?

Previously, we have identified propositions as types but it is fair enough to observe that types have more structure, they can carry more information by having different inhabitants. This becomes obvious in the next exercise:

Exercise 20 *Given* $A, B : \mathbf{Type}$ *and a relation* $R : A \to B \to \mathbf{Type}$ *the axiom of choice can be stated as follows:*

$$(\forall x : A.\exists y : B.R\,x\,y) \to \exists f : A \to B.\forall x : A.R\,x\,(f\,x)$$

Apply the propositions as types translation and prove the axiom of choice.

This is strange because usually the axiom of choice is an indicator of using some non-constructive principle in Mathematics. But now we can actually prove it in Type Theory? Indeed, the formulation above doesn't really convey the content of the axiom of choice because the existential quantification is translated as a Σ-type and hence makes the choice of the witness explicit. In conventional Mathematics propositions do not carry any information hence the axiom of choice has to build the choice function without any access to choices made when showing the premise.

To remedy this mismatch we are more specific about propositions: we say that a type $P : \mathbf{Type}$ is a proposition if it has at most one inhabitant, that is we define

$$\begin{aligned}&\mathrm{isProp} : \mathbf{Type} \to \mathbf{Type}\\ &\mathrm{isProp}\,A :\equiv \Pi x, y : A.x =_A y\end{aligned}$$

I write $P : \mathbf{Prop}$ for a type that is propositional, i.e. we can prove $\mathrm{isProp}\,P$. That is we are interpreting $\mathbf{Prop}$ as $\Sigma P : \mathbf{Type}.\mathrm{isProp}\,P$ but I am abusing notation in that I omit the first projection (that is an instance of subtyping as an implicit coercion).

Exercise 21 *Show that equality for natural number is a proposition, that is establish:*

$$\forall x, y : \mathbb{N}.\mathrm{isProp}\,(x =_{\mathbb{N}} y)$$

Looking back at the proposition as types translation we would like that for any proposition in predicate logic P we have that $[\![P]\!] : \mathbf{Prop}$. That can be shown to be correct for the so called negative fragment, that is the subset of predicate logic without $\vee$ and $\exists$.

Exercise 22 *Show that if* $P, Q : \mathbf{Prop}$ *and* $R : A \to \mathbf{Prop}$ *where* $A : \mathbf{Type}$ *then*

1. $[\![P \implies Q]\!] : \mathbf{Prop}$
2. $[\![P \wedge Q]\!] : \mathbf{Prop}$
3. $[\![\mathrm{True}]\!] : \mathbf{Prop}$
4. $[\![\mathrm{False}]\!] : \mathbf{Prop}$
5. $[\![\forall x : A.P]\!] : \mathbf{Prop}$

However this fails for disjunction and existential quantification: $[\![\mathrm{True} \vee \mathrm{True}]\!]$ is equivalent to Bool which is certainly not propositional since true $\neq$ false. Also $\exists x : \mathrm{Bool}.\mathrm{True}$ is equivalent to Bool and hence also not propositional.

To fix this we introduce a new operation which assigns to any type a proposition which expresses the fact that the type is inhabited. That is given $A : \mathbf{Type}$ we construct $\|A\| : \mathbf{Prop}$, this is called the *propositional truncation* of A. We can construct elements of $\|A\|$ from elements of A, that is we have a function $\eta : A \to \|A\|$. However, we hide the identity of a, that is we postulate that $\eta\, a = \eta\, b$ for all $a, b : A$. How can we construct a function out of $\|A\|$? It would be unsound if we would allow this function to recover the identity of an element. This can be avoided if the codomain of the function is itself propositional. That is given $P : \mathbf{Prop}$ and $f : A \to P$ we can lift this function to $\hat{f} : \|P\| \to A$ with $\hat{f}(\eta\, a) \equiv f\, a$.

Using $\|-\|$ we can redefine $[\![-]\!]$ such that $[\![P]\!] : \mathbf{Prop}$:

$$\begin{aligned} [\![P \vee Q]\!] &\equiv \|[\![P]\!] + [\![Q]\!]\| \\ [\![\exists x : A.P]\!] &\equiv \|\Sigma x : A.[\![P]\!]\| \end{aligned}$$

Now the translation of the axiom of choice

$$(\forall x : A.\exists y : B.R\, x\, y) \to \exists f : A \to B.\forall x : A.R\, x\, (f\, x)$$

which is

$$\Pi x : A.\|\Sigma y : B.R\, x\, y\| \to \|\Sigma f : A \to B.\forall x : A.R\, x\, (f\, x)\|$$

is more suspicious. It basically say if for every $x : A$ there is $y : B$ with a certain property, but I don't tell you which one, then there is a function $f : A \to B$ which assigns to every $x : A$ a $f\, x : B$ with a certain property but I don't tell you which function. This sounds like a lie to me!

Indeed lies can make our system inconsistent or they can lead to classical principle, which can be viewed is a form of lying that is not known to be inconsistent. Indeed, assuming one additional principle, *propositional extensionality* we can derive the principle of excluded middle. This proof is due to Diaconescu.

By propositional extensionality we mean that two propositions which are logically equivalent then they are equal:

$$\text{propExt} : \Pi P, Q : \mathbf{Prop}.(P \Leftrightarrow Q) \to P = Q$$

This is reasonable because all what we matters about a proposition is wether it is inhabited, and hence two proposition which are logically equivalent are actually indistinguishable and hence extensionally equal. Indeed, we will see that propExt is a consequence of the univalence principle.

Theorem 1 (Diaconescu) *Assuming the amended translation of the axiom of choice:*

$$\text{ac} : \Pi x : A.\|\Sigma y : B.R\,x\,y\| \to \|\Sigma f : A \to B.\forall x : A.R\,x\,(f\,x)\|$$

and propExt *we can derive the excluded middle for all propositions:*

$$\forall P : \mathbf{Prop}.P \vee \neg P$$

We are going to instantiate A with the type of inhabited predicates over Bool, that is

$$A :\equiv \Sigma Q : \text{Bool} \to \mathbf{Prop}.\exists b : \text{Bool}.Qb$$

$B \equiv \text{Bool}$ and the relation $R : A \to B \to \mathbf{Prop}$ is defined as follows:

$$R\,(Q, q)\,b :\equiv Q\,b$$

that is an inhabited predicate is related to the boolean it inhabits. Can we now prove the premise of the axiom?

$$\Pi x : A.\|\Sigma y : B.R\,x\,y\|$$

that is after plugging in the definition of A, B, R it becomes

$$\Pi(Q, q) : A.\|\Sigma b : B.Q\,b\|$$

and after some currying

$$\Pi Q : \text{Bool} \to \mathbf{Prop}.\|\Sigma b : \text{Bool}.Qb\| \to \|\Sigma b : B.Q\,b\|$$

it is quite obvious that we can.

Now let's look at the conclusion.

$$\|\Sigma f : A \to B.\Pi x : A.R\,x\,(f\,x)\|$$

Let's for the moment ignore the outermost $\|-\|$ and expand the types inside:

$$\Sigma f : A \to \text{Bool}.$$
$$\Pi(Q, q) : A.Q\,(f\,Q)$$

We consider two special predicates $T, F : \text{Bool} \to \mathbf{Prop}$:

$$T\,b :\equiv b = \text{true} \vee P$$
$$F\,b :\equiv b = \text{false} \vee P$$

Where does the P come from? From the beginning of this section: it is the $P : \mathbf{Prop}$ for which we want to show $P \vee \neg P$.

Now applying the conclusion of the axiom to these predicates we obtain two booleans $f\,T, f\,F : \text{Bool}$ and from the second part we know

$$T\,(f\,T) \equiv (f\,T = \text{true}) \vee P$$
$$F\,(f\,F) \equiv (f\,F = \text{false}) \vee P$$

Now let's analyse all the possibilities: there are four combinations:

1. $f\,T = \text{true} \wedge f\,F = \text{false}$
2. $f\,T = \text{true} \wedge P$
3. $P \wedge f\,F = \text{false}$
4. $P \wedge P$

In 2-4 we know that P holds, the only other alternative in which P is not proven is 1. In this case we can show $\neg P$ that is $P \to 0$. For this purpose assume P, in this case both $T\,b$ and $F\,b$ are provable for any b because P is and this means $T\,b \Leftrightarrow F\,b$ which now using propositional extensionality implies $T\,b = F\,b$. But this means that $T = F$ using functional extensionality. This cannot be since we assumed that $f\,T = \text{true}$ and $f\,F = false$, now $T = F$ would imply true = false which is false that is it implies 0. Hence we have shown $\neg P$ by deriving a contradiction from assuming P.

To summarise we have shown $P \vee \neg P$ because in the case 2-4 we have p and in 1 we have $\neg P$. But hang on what about the $||-||$ we have been ignoring? It doesn't matter since $P \vee \neg P$ is already a proposition and putting $||-||$ around it doesn't change anything.

What has really happened is that from the lie that we can recover information we have just hidden we can extract information as long as we hide the function doing the extraction. And this has nothing to do with Σ-types and existentials but with the behaviour of the hiding operation, or inhabitance $||-||$. Hence we can formulate a simpler version of the axiom of choice in Type Theory:

$$(\Pi x : A.\|B\,x\|) \to \|\Pi x : A.B\,x\|$$

This implies the revised translation of the axiom.

5.3 Dimensions of types

We have classified types as propositions if they have at most one inhabitant. uep says that equality is a proposition. Even of we are not accepting uep, we can use this to classify types: we say that a type is a set if all its equalities are propositions.

$$\begin{aligned}&\mathrm{isSet} : \mathbf{Type} \to \mathbf{Type}\\&\mathrm{isSet}\, A :\equiv \Pi_{x,y:A}\mathrm{isProp}\,(x =_A y)\end{aligned}$$

As for propositions we abuse notation and write $A : \mathbf{Set}$ if $A : \mathbf{Type}$ and we can show $\mathrm{isSet}\, A$. uep basically says that all types are set. The idea here is that sets are types which are quite ordinary, they reflect our intuition that equality is propositional.

Actually we can show that certain types are sets, e.g. exercise 21 basically asks to prove that $\mathbb{N} : \mathbf{Set}$. However, we can do much better, we can show in general that any type with a decidable equality is a set. This is a theorem due to Michael Hedberg.

Theorem 2 (Hedberg) *Given* $A : \mathbf{Type}$ *such that the equality is decidable*

$$d : \forall x, y : A.x = y \vee x \neq y$$

then we can show

$$\mathrm{isSet}\, A$$

To show Hedberg's theorem we establish a lemma saying that if there is a constant function on equality types then the equality is propositional. That is we assume

$$\begin{aligned}&f : \Pi_{x,y:A} x =_A y \to x =_A y\\&c : \Pi_{x,y:A}\Pi p, q : x =_A y \to f\, p = f\, q\end{aligned}$$

Now for arbitrary $x, y : A, p : x = y$ we can show that $p = \mathrm{trans}\,(f\, p)\,(\mathrm{sym}\,(f\,\mathrm{refl}))$ because using J this reduces ti showing that $\mathrm{refl} = \mathrm{trans}\,(f\,\mathrm{refl})\,(\mathrm{sym}\,(f\,\mathrm{refl}))$ which is one of the groupoid properties. Now given any $p, q : x = y$ we can show

$$\begin{aligned}p &= \mathrm{trans}\,(f\, p)\,(\mathrm{sym}\,(f\,\mathrm{refl}))\\&= \mathrm{trans}\,(f\, q)\,(\mathrm{sym}\,(f\,\mathrm{refl})) \qquad \text{using } c\\&= q\end{aligned}$$

And hence $=_A$ is propositional.

To construct the function f from decidability we assume $p : x = y$ and apply $d\, x\, y$. Either we have another proof $q : x = y$ and we return this one, or we have that $x \neq y$ but this contradicts that we already have a proof $p : x = y$ and we can use R^0. However, in either case the output doesn't depend on the actual value of the input and hence we can show that the function is constant.

Indeed, if we additionally assume the principle of functional extensionality

$$\text{funExt} : \forall f, g : A \to B.(\forall x : A.f\,x = g\,x) \to f = g$$

we can strengthen Hedberg's theorem:

Theorem 3 *Given* A : **Type** *such that the equality is stable*

$$s : \forall x, y : A.\neg\neg(x = y) \to x = y$$

then we can show

$$\text{isSet}\,A$$

This strengthening shows that function types are sets, e.g. $\mathbb{N} \to \mathbb{N}$: **Set** even though its equality is not decidable but it is stable.

To prove the stronger version we observe that for if equality is stable, we can construct the function $g : \Pi x, y : A.x = y \to x = y$ by composing s with the obvious embedding $x = y \to \neg\neg(x = y)$. Since $\neg\neg(x = y)$ is propositional, f must be constant.

Exercise 23 *Show that equality of* $\mathbb{N} \to \mathbb{N}$ *is stable.*

We can extend the hierarchy we have started to construct with **Prop** and **Set**. For example a type whose equalities are sets is called a groupoid (indeed it corresponds to the notion of groupoid which we have introduced previously).

$$\begin{aligned}&\text{isGroupoid} : \mathbf{Type} \to \mathbf{Type}\\&\text{isGroupoid}\,A :\equiv \forall x, y : A.\text{isSet}\,(x =_A y)\end{aligned}$$

We can also extend this hierarchy downwards, we can redefine a proposition as a type such that its equalities are types with eactly one element - these types are called *contractible*:

$$\begin{aligned}&\text{isContractible} : \mathbf{Type} \to \mathbf{Type}\\&\text{isContractible}\,A :\equiv \Sigma a : A.\Pi x : A.x = a\end{aligned}$$

$$\begin{aligned}&\text{isProp} : \mathbf{Type} \to \mathbf{Type}\\&\text{isProp}\,A :\equiv \Pi x, y : A.\text{isContractible}\,(x = y)\end{aligned}$$

We can now define the hierarchy starting with contractible types. For historic reasons (i.e. to be compatible with notions from homotopy theory) we start counting with -2 and not with 0. I am calling the levels dimensions, they are also called truncation levels.

$$\begin{aligned}&\text{hasDimension} : \mathbb{N}_{-2} \to \mathbf{Type} \to \mathbf{Type}\\&\text{hasDimension}\,(-2)\,A :\equiv \text{isContractible}\,A\\&\text{hasDimension}\,(n+1)\,A :\equiv \forall x, y : A.\text{hasDimension}\,(x =_A y)\end{aligned}$$

I am writing $\mathbb{N}_{-2}$ for a version of the natural numbers where I start counting with -2. To summarize the definitions so far:

Dimension	**Name**
-2	Contractible types
-1	Propositions
0	Sets
1	Groupoids

We also introduce the notation $n-\mathbf{Type}$ for $A : \mathbf{Type}$ such that we can show $\mathrm{hasDimension}\, n\, A$.

To convince ourselves that this really is a hierarchy, that is that every $n-\mathbf{Type}$ is also a $n+1-\mathbf{Type}$ we need to show that the hierarchy actually stops at -2 that is that the equality of a contractible type is again contractible.

To show this assume is given a contractible type $A : \mathbf{Type}$ that is we have $a : A$ and $c : \Pi x : A.a = x$. Now we want to show that for all $x, y : A$ the equality $x =_A y$ is contractible, that is we have an element and all other elements are equal. We define $d : \Pi x, y : A.x =_A y$ as $d\, x\, y :\equiv \mathrm{trans}\,(\mathrm{sym}\,(c\, x))\,(c\, y)$. Now it remains to show that $e : \Pi x, y : A.\Pi p : x =_A y.d\, x\, y = p$. Using J we can reduce this to $d\, x\, x = \mathrm{refl}\, x$ unfolding $d\, x\, x \equiv \mathrm{trans}\,(\mathrm{sym}\,(c\, x))\,(c\, x)$ we see that this is an instance of one of our groupoid laws.

As a corollary we obtain that $\mathrm{hasDimension}\, n\, A$ implies $\mathrm{hasDimension}\,(n+1)\, A$.

5.4 Extensionality and univalence

As I have already mentioned in the introduction: extensionality means that we identify mathematical objects which behave the same even of they are defined differently. An example are the following two functions:

$$f, g : \mathbb{N} \to \mathbb{N}$$
$$f\, x :\equiv x + 1$$
$$g\, x :\equiv 1 + x$$

We can show that $\forall x : \mathbb{N}.f\, x = g\, x$ but can we show that $f = g$? The answer is **no**, because if there were a proof without any assumption it would have to be refl and this would only be possible if $f \equiv g$ but it is clear that they are not definitionally equal. However, we cannot exhibit any property not involving this equality which would differentiate them. Another example are the following two propositions:

$$P, Q : \mathbf{Prop}$$
$$P :\equiv \mathrm{True}$$
$$Q :\equiv \not\mathrm{False}$$

Again we cannot show that $P \equiv Q$ even though there clearly is no way to differentiate between them.

We have already mentioned the two principles which are missing here':

$$\text{funExt} : \forall f, g : A \to B.(\forall x : A.f\,x = g\,x) \to f = g$$
$$\text{propExt} : \Pi P, Q : \mathbf{Prop}.(P \Leftrightarrow Q) \to P = Q$$

We note that the corresponding principles are true in set theory, which seems to contradict my statement that Type Theory is better for extensional reasoning than set theory. However, this shortcoming can be fixed since all constructions preserve these equalities. This can be made precise by interpreting the constructions in the setoid model, where every type is modelled by a set with an equivalence relation. In this case we can model function types by the set of functions and equality is extensional equality. The same works for **Prop**, a proposition is modelled by the set of propositions identified if they are logically equivalent.

However, the shortcoming of set theory becomes obvious if we ask the next question: when are two sets equal? For example

$$A, B : \mathbf{Set}$$
$$A :\equiv \overline{1} + \overline{2}$$
$$B :\equiv \overline{2} + \overline{1}$$

We have no way to distinguish two sets with 3 elements, hence following the same logic as above they should be equal. However, they are not equal in set theory under the usual encoding of finite sets and + and it would be hard to fix this in general.

When are two sets equal? Given $f : A \to B$ we say that f is an isomorphism if there it has an inverse, that is [3]

$$\begin{aligned}&\text{isIso} : (A \to B) \to \mathbf{Type}\\ &\text{isIso}\, f :\equiv \quad \Sigma g : B \to A\\ &\qquad\qquad\quad \eta : \Pi x : B.f\,(g\,x) = x\\ &\qquad\qquad\quad \epsilon : \Pi x : A.g\,(f\,x) = x\end{aligned}$$

And we define $A \simeq B :\equiv \Sigma f : A \to B.\text{isIso}\, f$. Using isIso we can formulate a new extensionality principle: we want to say that there is an isomorphism between isomorphism of sets and equality of sets. Indeed, we can observe that the is a function from equality of sets to isomorphism because every set is isomorphic to itself using J.

Exercise 24 *Define* $\text{eq2iso} : \Pi_{A,B:\mathbf{Set}} A = B \to A \simeq B$

Using this we can state extensionality for sets

$$\text{extSet} : \text{isIso}\,\text{eq2iso}$$

As a corollary we get that $A \simeq B \to A = B$. Since $\overline{1} + \overline{2} \simeq \overline{1} + \overline{2}$ we can show that $\overline{1} + \overline{2} = \overline{1} + \overline{2}$

[3] I am using here a record like syntax for iterated Σ-types.

Exercise 25 *Show that*

1. $(A + B) \to C = (A \to C) \times (B \to C)$
2. $(A \times B) \to C = A \to B \to C$
3. $1 + \mathbb{N} = \mathbb{N}$
4. $\mathbb{N} \times \mathbb{N} = \mathbb{N}$
5. $\mathrm{List}\,\mathbb{N} = \mathbb{N}$
6. $\mathbb{N} \to \mathbb{N} \neq \mathbb{N}$

assuming $A, B, C : \mathbf{Set}$. *For which of the results do we not need* extSet*?*

Exercise 26 *An alternative to isomorphism is bijection, one way to say that a function is bijective is to say there is a unique element in the domain which is mapped to an element of the codomain. We define* $\exists!$ *(exists unique) :*

$$\exists! x : A.P\,x :\equiv \exists x : A.P\,x \wedge \forall y : A, P\,y \implies x = y$$

and using this we define what is a bijection:

$$\begin{aligned}&\mathrm{isBij} : (A \to B) \to \mathbf{Prop}\\ &\mathrm{isBij}\,f :\equiv \forall y : B.\exists! x : A.f\,x = y\end{aligned}$$

Show that isomorphism and bijection are logically equivalent:

$$\forall\, f : A \to B.\mathrm{isBij}\,f \Leftrightarrow \mathrm{isIso}\,f$$

extSet is incompatible with uniqueness of equality proofs (uep) because there are two elements of $\overline{2} \simeq \overline{2}$, namely identity and negation. If we assume that there is only one proof of equality for $\overline{2} = \overline{2}$ then we also identify this two proofs and hence $0_2 = 1_2$ which is inconsistent.

So far we have only considered sets what about types in general? There is a twist: if equality is not propositional then the η and ϵ components of isIso are not propositional in general. Indeed, assuming extSet for all types is unsound. Instead we need to refine the notion of isomorphism by introducing an extra condition which relates η and ϵ. We call this *equivalence* of types.

$$\begin{aligned}&\mathrm{isEquiv} : (A \to B) \to \mathbf{Type}\\ &\mathrm{isEquiv}\,f :\equiv \quad \Sigma g : B \to A\\ &\qquad\qquad\quad \eta : \Pi x : B.f\,(g\,x) = x\\ &\qquad\qquad\quad \epsilon : \Pi x : A.g\,(f\,x) = x\\ &\qquad\qquad\quad \delta : \Pi x : A.\eta\,(f\,x) = \mathrm{resp}\,f\,(\epsilon\,x)\end{aligned}$$

And we define $A \cong B :\equiv \Sigma f : A \to B.\mathrm{isEquiv}\,f$.

Exercise 27 *Define* eq2equiv : $\Pi_{A,B:\mathbf{Set}} A = B \to A \cong B$

We can now state extensionality for types, which is what is commonly called univalence.

$$\text{univalence} : \text{isEquiv}\,\text{eq2equiv}$$

As before for extSet a consequence is that equivalence of types implies equality $A \cong B \to A = B$ which is what most people remember about univalence. In the special case of sets equivalence and isomorphism agree because the type of δ is an equivalence, that is for sets we have $(A \simeq B) = (A \cong B)$. But more is true, even for types in general equivalence and isomorphism are logically equivalence, that is $(A \simeq B) \Leftrightarrow (A \cong B)$. While this may be surprising at the first glance, it just means that we can define functions in both directions but they are not inverse to each other. Indeed, in general there are more proofs of an isomorphism that of an equivalence, indeed isEquiv f is a proposition, while isIso f in general isn't (but it is if f is a function between sets). However, the logical equivalence of equivalence and isomorphism means that to establish an equivalence all we need to do is to construct an isomorphism. IN particular all the equalities of exercise 25 hold for types in general.

The definition of isEquiv looks strangely assymmetric. Indeed, we could have replaced δ with its symmetric twin:

$$\delta' : \Pi y : A.\text{resp}\, g\, \eta\, y = \epsilon\, (g\, y)$$

Indeed, we can either use δ or δ' it doesn't make any difference. Why don't we use both? Indeed, this would exactly be the definition of an adjunction between groupoids. However, assuming both messes everything up, now isEquiv f is no longer a proposition and we need to add higher level coherence equations to fix this. Indeed, there is such an infinitary (coinductive) defintion of equivalence. But we don't need to use this, the asymmetric definition (or its mirror image) does the job.

Exercise 28 *We can extend exercise 26 to the case of equivalences by taking the equality proofs into account. That is we say that not only there is a unique inverse but that the pair of inverses and the proof that there are an inverse are unique that is contractible. We define*

$$\begin{aligned}&\text{isEquiv}' : (A \to B) \to \mathbf{Type}\\ &\text{isEquiv}'\, f :\equiv \Pi y : B.\text{isContr}\,(\Sigma x : A.f\, x = y)\end{aligned}$$

We have observed that already extSet implies that there are non-propositional equality, e.g. $\overline{2} = \overline{2}$. In other words the first universe $\mathbf{Type}_0$ is not a set. Nicolai Kraus has generalize this and shown that using univalence we can construct types which are non $n-\mathbf{Type}$ for any $n : \mathbb{N}$. I leave is an exercise to do the forst step in this construction:

Exercise 29 *Show that the equalities of* $\Sigma X : \mathbf{Type}_0.X = X$ *is not always sets and hence* $\mathbf{Type}_1$ *is not a groupoid.*