OPLSS 22 Introduction to Type Theory (5)

Cosmo Viola, Jiawei Chen, Nabil Hassein, Daniel Zackon

June 2022

1 Equality

1.1 Equality Elimination

Last time, we defined equality. Now, we need an eliminator for equality.

```
E≡ : (M : (a b : A) → a ≡ b → Set)
              → ((a : A) → M a a (refl {a}))
              → (a b : A) (p : a ≡ b) → M a b p
E≡ M m–r a .a refl = m–r a
```

This eliminator is called J, and was named by Per Martin-Loef. We might try to prove the uniqueness of identity proofs using this axiom:

```
uip : (a b : A) (p q : a ≡ b) → p ≡ q
uip = E≡ (λ a b p → (q : a ≡ b) → p ≡ q) {}1
```

However, it turns out that we can't actually finish this proof; Axiom J does not allow UIP to be derived. If we really want UIP, we can include Thomas Streicher's Axiom K.

How do we know that Axiom J does not imply UIP? Martin Hofmann and Thomas Streicher produced a model where J holds but UIP does not. This model is the groupoid model, in which types are groupoids, i.e. a category where all morphisms are isomorphisms. If we do this, we find that there is a type on which the identity type is actually the group of integers $\mathbb{Z}$, and hence UIP does not hold in that model. Thus, J cannot prove UIP.

1.2 Equality as a Type

Through the 90s, it was accepted that Axiom K was necessary. However, people began to think about type theories where isomorphism of types is equality. In 2008, Vladimir Voevodsky seriously began thinking about this because of his work in algebraic topology. He eventually proposed a type theory where types are spaces in the sense of homotopy theory, and a type along with its identity types forms an ω-groupoid. This type theory satisfies the univalence principle.

In type theory, unlike in conventional mathematics, equality is just a type, so we can reason about equality in powerful ways. Specifically, we can consider equality between members of equality types. When mathematicians reason about objects, they often consider objects which behave the same as being equal. For instance, when doing group theory, mathematicians will often freely move between the group of integers, $\mathbb{Z}$, and the group of loops around a circle (i.e. $\pi_1(S^1)$, the fundamental group of the circle). This is because when looking at the structure of groups, these objects behave precisely the same, even though the objects aren't precisely equal. In such a case, we say that the objects are isomorphic. Homotopy type theory allows us to formalize this common reasoning technique.

We say that a function $f : A \to B$ is an isomorphism if $\forall b : B, \exists! a : A, f\ a = b$. A function $f : A \to B$ is an equivalence if for all $b : B$, there is a unique $a : A$ and $p : f\ a \equiv b$. A type is called contractible if it has exactly one element.

```
isContr : Set → Set
isContr A = Σ[ a ∈ A ] ((x : A) → a ≡ x)

isEquiv : (A → B) → Set
isEquiv : {A} {B} f =
  (b : B) → isContr (Σ[a ∈ A ] (f a ≡ b))

_ ≈ _ : Set → Set → Set
A ≈ B = Σ[ f ∈ (A → B)] (isEquiv f)

≡→≈ : (A ≡ B) → (A → B)
postulate
  ua : {A B : Set} → isEquiv (≡→≈ {A} {B})
```

This last postulate is the univalence axiom. Intuitively, it states that equivalence is equivalent to equality. If an equivalence exists between two types, then there is a function which accepts that equivalence as input and produces an equality as output. This is the formalization of moving between isomorphic structures mentioned earlier.

2 hsets

Next, we will discuss hsets. We call contractible types -2-types. For any type, we can ask if it is an n-type:

```
is-(-2)-type A = isContractible A
is-(n+1)-type A = (a b : A) → is-n-type (a ≡ b)
```

We call -1-types propositions:

```
isProp A = (a b : A) → a ≡ b
```

Before, we said that Prop = Set, but really we should say that

```
Prop = HProp = {A : Set | isProp A}
```

Note that $\vee$ and $\exists$ are not closed under HProp. Thus, we define a propositional truncation, which turns a type into a proposition by ignoring the details of the proof of that object.

```
||_|| : Set → Set
isProp ||A||
P ∨ Q = || P ⊎ Q ||
∃ A P = || Σ A P ||
```

Next, we consider HSet, the type of 0-types. For these types, equality is a proposition. These types are ordinary types in some sense.

```
HSet = { A : Type | isHSet A }

(Bool ≡ Bool) ≈ (Bool ≈ Bool) = Bool
```

There are two equivalences on Bool; id, and negation. Finally, we can see that HSet is a 1-type.

3 Cubical Type Theory

There's an important question left to answer though; can we give a computational interpretation to the univalence axiom? The answer, discovered recently, is yes, using cubical type theory. In homotopy type theory, we think of equalities as being paths between points, and paths are functions out of the interval $[0, 1]$. Cubical type theory introduces an abstraction of the interval as a type, called I, containing points $i0$ and $i1$.

```
a b : A, p : a ≡ b
p : I → A, p i0 = a, p i1 = b

refl' : (a : A) → a ≡ a
refl' a i = a
```

Univalence proves functional extensionality, so if we have a computational interpretation for univalence, we also have one for functional extensionality.

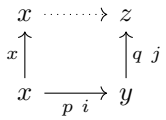
```
fun-ext : (f g : A → B) → ((a : A) → f a ≡ g a) → f ≡ g
fun-ext f g p i = p a i

lemma : (n : ℕ) → mapStream suc (from n) ≡ from (suc n)
hd lemma n i = suc n
tl lemma n i = lemma (suc n) i
```

There is a duality here. Proof by induction converts to pattern matching and structural recursion. Proof by copattern matching converts to copattern matching and guarded corecursion.

```
trans : {a b c : A} → a ≡ b → b ≡ c → a ≡ c
trans {a = a} {b = b} {c = c} p q i =
      hcomp (λ j → λ { (i = i0) → a
                     ; (i = i1) → q j }) (p i)
```

The hcomp function comes out of mathematics, and has a computational interpretation of filling in the missing side of a square, as seen in this diagram from the Cubical Agda documentation:



Also present in cubical type theory are glue types, which are important for giving the interpretation of univalence. Cubical type theory also introduces higher inductive types, which act as generalizations of quotients, and allow introducing new equalities into a type and its equality types.

Prof. Altenkirch is working on higher observational type theory with Kaposi and Shulmann at present.