A Type Theory with Native Homotopy Universes

Robin Adams[1] and Andrew Polonsky[2]

[1] Universitetet i Bergen
[2] University Paris Diderot

We present a type theory called $\lambda\simeq_2$ with an extensional equality relation; that is, the universe of types is closed by reflection into it of the logical relation defined by induction on the structure of types.

The type system has three universes:

- The universe **Prop** of *propositions*. An object of **Prop** is called a *proposition*, and the objects of a proposition are called *proofs*. There is a proposition **1**, which has a unique proof $*$.
- The universe **Set** of *sets*.
- The universe **Groupoid** of *groupoids*.

For each universe U, we have an associated relation of equality between U-types $\simeq$, and between objects of U-equal types $\sim$. The associated rules of deduction are:

$$\frac{A : U \quad B : U}{A \simeq B : U} \qquad \frac{a : A \quad e : A \simeq B \quad b : B}{a \sim_e b : U^-}$$

where U^- is the universe one dimension below U. Thus:

- Given two propositions ϕ and ψ, we have the proposition $\phi \simeq \psi$ which denotes the proposition 'ϕ if and only if ψ'. If $\delta : \phi$, $\epsilon : \psi$ and $\chi : \phi \simeq \psi$, then $\delta \sim_\chi \epsilon = \mathbf{1}$. (*Cf* In homotopy type theory, any two objects of a proposition are equal.)
- Given two sets A and B, we have the set $A \simeq B$, which denotes the set of all bijections between A and B. Given $a : A$, $f : A \simeq B$ and $b : B$, we have the proposition $a \sim_f b$: **Prop**, which denotes that a is mapped to b by the bijection f.
- Given two groupoids G and H, we have the groupoid $G \simeq H$, which denotes the groupoid of all groupoid isomorphisms between G and H. Given $g : G$, $\phi : G \simeq H$ and $h : H$, we have the set $g \sim_\phi h$: **Set**, which can be thought of as the set of all paths between $\phi(g)$ and h in H.

We have reflexivity provided by the following typing rules:

$$\frac{A : U}{1_A : A \simeq A} \qquad \frac{a : A}{r_a : a \sim_{1_A} a}$$

The relation $\sim_{1_A}$ thus behaves like an equality relation on each type A.

The introduction and elimination rules for $\simeq$ ensure that $A \simeq B$ is the type of equivalences between A and B:

$$\frac{e : A \simeq B \quad a : A}{e^+(a) : B} \qquad \frac{e : A \simeq B \quad b : B}{e^-(b) : A}$$

$$\frac{a : A \quad b : B \quad e : A \simeq B}{e^=(a,b) : (a \sim_{1_A} e^-(b)) \simeq (e^+(a) \sim_{1_B} b)}$$

$$\frac{\begin{array}{ll} \Gamma, x : A & \vdash b : B \\ \Gamma, y : B & \vdash a : A \\ \Gamma, x : A, y : B & \vdash e : (x \sim_{1_A} a) \simeq (b \sim_{1_B} y) \end{array}}{\Gamma \vdash \mathsf{univ}([x : A]b, [y : B]a, [x : A, y : B]e) : A \simeq B}$$

Each universe is itself an object of the next universe:

$$\mathbf{1} : \mathbf{Prop} : \mathbf{Set} : \mathbf{Groupoid}$$

and we have the following definitional equalities:

$$\phi \sim_{1_{\mathbf{Prop}}} \psi \stackrel{\text{def}}{=} \phi \simeq \psi, \quad A \sim_{1_{\mathbf{Set}}} B \stackrel{\text{def}}{=} A \simeq B$$

The following computation rules also hold in $\lambda{\simeq_2}$.

TODO

We therefore note the following features of $\lambda{\simeq_2}$:

- Univalence holds definitionally — an equality between types $A \simeq B$ is exactly (definitionally) the type of equivalences between A and B.
- Transport respects reflexivity and composition definitionally.

This type theory has been formalised in Agda, using the method of the system `Kipling` from McBride [5]. The method ensures that, if s and t are definitionally equal expressions in $\lambda \simeq_2$, then $[\![s]\!]$ and $[\![t]\!]$ are definitionally equal objects in Agda.

We have

- a type $\mathsf{data}\ \mathsf{Cx} : \mathsf{Set}_1$
- functions $[\_]\mathsf{C} : \mathsf{Cx} \to \mathsf{Set}_1$
  $\mathsf{EQC} : \forall\, \Gamma \to [\,\Gamma\,]\mathsf{C} \to [\,\Gamma\,]\mathsf{C} \to \mathsf{Set}$
  $\mathsf{EQC}_2 : \forall\, \{\Gamma\}\, \{a_1\ a_2\ b_1\ b_2 : [\,\Gamma\,]\mathsf{C}\} \to$
  $\quad \mathsf{EQC}\ \Gamma\ a_1\ a_2 \to \mathsf{EQC}\ \Gamma\ b_1\ b_2 \to \mathsf{EQC}\ \Gamma\ a_1\ b_1 \to \mathsf{EQC}\ \Gamma\ a_2\ b_2 \to \mathsf{Set}$
- an inductive data type $\mathsf{data}\ \_\vdash\_\ni\_\ (\Gamma : \mathsf{Cx}) : \forall\, \{n\}\ (T : \mathsf{Typeover}\ n\ \Gamma)\ (t : \mathsf{Section}\ T) \to \mathsf{Set}_1$

The formalisation is available online at `https://github.com/radams78/Equality2`.

Related Work An earlier version of this system was presented in [2]. In this talk, we also give semantics to this system in Agda's type theory extended with a native type of groupoids, and show how the syntax and semantics are formalised in Agda.

Our system is closely related to the system PHOML (Predicative Higher-Order Minimal Logic) presented in [1]. The system $\lambda{\simeq_2}$ can be seen as an extension of PHOML with groupoids, and with a univalent equality for both sets and groupoids.

Cubical type theory [3, 4] has a very similar motivation to this work, and also offers a type theory with univalence and a computational interpretation. One difference with our system is that the following equations hold judgementally:

TODO

The corresponding equations in cubical type theory would be $\mathsf{transp}^i\, A\, a = a$ where i does not occur in A, and $\mathsf{transp}^j\, B\, (\mathsf{transp}^i\, A\, a) = \mathsf{transp}^i (\mathsf{comp}^j\, U\, [(i = 0) \mapsto A[i := 0], (i = 1) \mapsto B])\, A\, a$. In cubical type theory, these equations do hold up to definitional equality for all terms a. They hold up to definitional equality when a is a canonical object, and up to propositional equality in general.

References

[1] Robin Adams, Marc Bezem, and Thierry Coquand. A normalizing computation rule for propositional extensionality in higher-order minimal logic. Submitted for publication in TYPES 2016. https://arxiv.org/abs/1610.00026, 2017.

[2] Robin Adams and Andrew Polonsky. A type system with native homotopy universes. Talk given at Workshop on Homotopy Type Theory / Univalent Foundations, Porto, Portugal, 2016. http://hott-uf.gforge.inria.fr/andrew.pdf.

[3] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. *CoRR*, abs/1611.02108, 2016.

[4] Simon Huber. Canonicity for cubical type theory. `arXiv:1607.04156`, 2016.

[5] Conor McBride. Outrageous but meaningful coincidences: Dependent type-safe syntax and evaluation. In Bruno C. d. S. Oliveira and Marcin Zalewksi, editors, *Proceedings of the 6th ACM SIGPLAN Workshop on Generic Programming (WGP 2010)*, pages 1–12. ACM, 2010.