A Type Theory with Native Homotopy Universes

Robin Adams and Andrew Polonsky

May 31, 2017

The Programmers' Credo

"We do these things, not because they are easy, but because we thought they would be easy."

Introduction

Systems with univalence axiom:

- Book-HoTT
  - Univalence is just an axiom (constant)
  - Computation with univalence gets stuck.
- Cubical type theory
  - Not all the definitional equalities we want:

$$\mathsf{transp}^i (pi)(\mathsf{transp}^i (qi) a) \not\equiv \mathsf{transp}^i ((p \circ q) i) a$$

$$\mathsf{transp}^i A a \not\equiv a \qquad (i \notin A)$$

Idea (Coquand [Coq11]): Define equality on each type by induction on the type structure.

Logical Relations

Introduced by Gandy [Gan56].
The name 'logical relation' has been used for several different families of equivalence relations on the terms of A defined by recursion on A.

$$f \sim_{A \to B} g \stackrel{\text{def}}{=} \forall x, y : A.(x \sim_A y \Rightarrow fx \sim_B gy)$$

$$p \sim_{A \times B} q \stackrel{\text{def}}{=} \pi_1(p) \sim_A \pi_1(q) \wedge \pi_2(p) \sim_B \pi_2(q)$$

$$A \sim_{\mathcal{U}} B \stackrel{\text{def}}{=} A \simeq B$$

Not a New Idea?

Recall Martin-Löf's meaning explanation [ML82]:

> A canonical type A is defined by prescribing how a canonical object of type A is formed as well as how two equal canonical objects of type A are formed.

Suggests that equality is supposed to vary with the type.

Past Work

Several attempts:

- Observational Type Theory (OTT) [AMS07] A 1-dimensional theory (sets and propositions)
- 2-dimensional type theory [LH12] A 2-dimensional type theory with equality reflection
- $\lambda \simeq$ [Pol14] An inconsistent system with $* : *$
- PHOML [ABC16] A 1-dimensional theory with univalence

Attempts to do this beyond dimension 2 end up in 'setoid hell'

TDLR

Presenting TDLR (pronounced 'toddler') — Two Dimensional Logical Relation theory. (Previously called $\lambda \simeq_2$)

TDLR

Presenting TDLR (pronounced 'toddler') — Two Dimensional Logical Relation theory. (Previously called $\lambda \simeq_2$)
Idea: A disciplined approach to creating a type theory by:

1. Using a proof assistant
2. Formalising syntax and semantics in groupoids simultaneously

TDLR

Presenting TDLR (pronounced 'toddler') — Two Dimensional Logical Relation theory. (Previously called $\lambda \simeq_2$)
Idea: A disciplined approach to creating a type theory by:

1. Using a proof assistant
2. Formalising syntax and semantics in groupoids simultaneously

Work in progress — rules of deduction still in flux

TDLR

Presenting TDLR (pronounced 'toddler') — Two Dimensional Logical Relation theory. (Previously called $\lambda \simeq_2$)
Idea: A disciplined approach to creating a type theory by:

1. Using a proof assistant
2. Formalising syntax and semantics in groupoids simultaneously

Work in progress — rules of deduction still in flux
Four judgement forms:

- $\Gamma \vdash_3 G$ gpd — G is a groupoid
- $\Gamma \vdash_2 A : G$ — A is an object of groupoid G
- $\Gamma \vdash_1 t : A$ — t is an element of set A
- $\Gamma \vdash_0 \delta : t$ — δ is a proof of proposition t

Propositions

Let there be propositions.

Propositions

Let there be propositions. Let propositions have proofs.

Propositions

Let there be propositions. Let propositions have proofs. Given propositions ϕ and ψ, let there be a proposition $\phi \Leftrightarrow \psi$

$$\frac{\Gamma, p : \phi \vdash \delta : \psi \qquad \Gamma, q : \psi \vdash \epsilon : \phi}{\Gamma \vdash \mathsf{univ}_0(p.\delta, q.\epsilon) : \phi \Leftrightarrow \psi}$$

$$\frac{\Gamma \vdash \delta : \phi \Leftrightarrow \psi \quad \Gamma \vdash \epsilon : \psi}{\Gamma \vdash \delta^+ \epsilon : \psi}$$

$$\frac{\Gamma \vdash \delta : \phi \Leftrightarrow \psi \quad \Gamma \vdash \epsilon : \psi}{\Gamma \vdash \delta^- \epsilon : \phi}$$

Sets

Let there be sets.

Sets

Let there be sets.

Let sets have elements.

Sets

Let there be sets.

Let sets have elements.

Given $a, b : S$, let there be a proposition $a =_S b$ and proof $r(a) : a =_S a$.

Sets

Let there be sets.

Let sets have elements.

Given $a, b : S$, let there be a proposition $a =_S b$ and proof $r(a) : a =_S a$.

Given sets S, T, let there be a set $S \simeq T$ of *bijections*.

$$\frac{\Gamma, x : S \vdash t : T \quad \Gamma, y : T \vdash s : S \qquad \Gamma, x : S, y : T \vdash \delta : (x =_S s) \Leftrightarrow (t =_T y)}{\Gamma \vdash \mathsf{univ}_1(x.t, y.s, xy.\delta) : S \simeq T}$$

$$\frac{\Gamma \vdash e : S \simeq T \quad \Gamma \vdash s : S}{\Gamma \vdash e^+ s : T}$$

$$\frac{\Gamma \vdash e : S \simeq T \quad \Gamma \vdash t : T}{\Gamma \vdash e^- t : S}$$

$$\frac{\Gamma \vdash e : S \simeq T \quad \Gamma \vdash s : S \quad \Gamma \vdash t : T}{\Gamma \vdash e^=(s, t) : (s = e^- t) \Leftrightarrow (e^+ s = t)}$$

Groupoids

Let there be groupoids.

Groupoids

Let there be groupoids.

Let groupoids have objects.

Groupoids

Let there be groupoids.

Let groupoids have objects.

Given $a, b : G$, let there be a set $a =_G b$ and element $r(a) : a =_G a$.

Groupoids

Let there be groupoids.

Let groupoids have objects.

Given $a, b : G$, let there be a set $a =_G b$ and element $r(a) : a =_G a$.

Given groupoids G, H, let there be a groupoid $G \simeq H$.

$$\frac{\Gamma, x : G \vdash t : H \quad \Gamma, y : H \vdash s : G \qquad \Gamma, x : G, y : H \vdash e : (x =_G s) \simeq (t =_H y)}{\Gamma \vdash \mathsf{univ}_2(x.t, y.s, xy.e) : G \simeq H}$$

$$\frac{\Gamma \vdash \phi : G \simeq H \quad \Gamma \vdash s : G}{\Gamma \vdash \phi^+ s : H}$$

$$\frac{\Gamma \vdash \phi : G \simeq H \quad \Gamma \vdash t : H}{\Gamma \vdash \phi^- t : G}$$

$$\frac{\Gamma \vdash \phi : S \simeq T \quad \Gamma \vdash s : S \quad \Gamma \vdash t : T}{\Gamma \vdash \phi^=(s, t) : (s =_G \phi^- t) \simeq (\phi^+ s = t)}$$

Properties of Equality

Define symmetry:

$$\frac{\Gamma \vdash \delta : \phi \Leftrightarrow \psi}{\Gamma \vdash \mathrm{sym}_0(\delta) \stackrel{\mathrm{def}}{=} \mathsf{univ}_0(p.\delta^- p, p.\delta^+ p) : \psi \Leftrightarrow \phi}$$

Properties of Equality

Define symmetry:

$$\frac{\Gamma \vdash \delta : \phi \Leftrightarrow \psi}{\Gamma \vdash \mathrm{sym}_0(\delta) \stackrel{\text{def}}{=} \mathsf{univ}_0(p.\delta^- p, p.\delta^+ p) : \psi \Leftrightarrow \phi}$$

$$\frac{\Gamma \vdash e : S \simeq T}{\begin{array}{ll}\Gamma \vdash & \mathrm{sym}_1(e) \stackrel{\text{def}}{=} \mathsf{univ}_1(x.e^- x, y.e^+(y), xy.\mathrm{sym}_0(e^=(x, y))) \\ & : T \simeq S\end{array}}$$

Properties of Equality

Define symmetry:

$$\frac{\Gamma \vdash \delta : \phi \Leftrightarrow \psi}{\Gamma \vdash \mathrm{sym}_0(\delta) \stackrel{\text{def}}{=} \mathsf{univ}_0(p.\delta^- p, p.\delta^+ p) : \psi \Leftrightarrow \phi}$$

$$\frac{\Gamma \vdash e : S \simeq T}{\Gamma \vdash \begin{array}{l} \mathrm{sym}_1(e) \stackrel{\text{def}}{=} \mathsf{univ}_1(x.e^- x, y.e^+(y), xy.\mathrm{sym}_0(e^=(x,y)) \\ : T \simeq S \end{array}}$$

$$\frac{\Gamma \vdash e : G \simeq H}{\Gamma \vdash \begin{array}{l} \mathrm{sym}_2(e) \stackrel{\text{def}}{=} \mathsf{univ}_2(x.e^- x, y.e^+ y, xy.\mathrm{sym}_1(e^=(x,y)) \\ : H \simeq G \end{array}}$$

Can define transitivity.

Function types

Given propositions ϕ, ψ, there is a proposition $\phi \to \psi$:

$$\frac{\Gamma, p : \phi \vdash \delta : \psi}{\Gamma \vdash \lambda p.\delta : \phi \to \psi} \qquad \frac{\Gamma \vdash \delta : \phi \to \psi \quad \Gamma \vdash \epsilon : \phi}{\Gamma \vdash \delta\epsilon : \psi}$$

Function types

Given propositions ϕ, ψ, there is a proposition $\phi \to \psi$:

$$\frac{\Gamma, p : \phi \vdash \delta : \psi}{\Gamma \vdash \lambda p.\delta : \phi \to \psi} \qquad \frac{\Gamma \vdash \delta : \phi \to \psi \quad \Gamma \vdash \epsilon : \phi}{\Gamma \vdash \delta\epsilon : \psi}$$

Given sets S, T, there is a set $S \to T$ of *functions*:

$$\frac{\Gamma, x : S \vdash t : T}{\Gamma \vdash \lambda x.t : S \to T} \qquad \frac{\Gamma \vdash f : S \to T \quad \Gamma \vdash s : S}{\Gamma \vdash fs : T}$$

$$(f =_{S \to T} g) \equiv (\forall x, y : S. x =_S y \to fx =_T gy)$$

Function types

Given propositions ϕ, ψ, there is a proposition $\phi \to \psi$:

$$\frac{\Gamma, p : \phi \vdash \delta : \psi}{\Gamma \vdash \lambda p.\delta : \phi \to \psi} \qquad \frac{\Gamma \vdash \delta : \phi \to \psi \quad \Gamma \vdash \epsilon : \phi}{\Gamma \vdash \delta\epsilon : \psi}$$

Given sets S, T, there is a set $S \to T$ of *functions*:

$$\frac{\Gamma, x : S \vdash t : T}{\Gamma \vdash \lambda x.t : S \to T} \qquad \frac{\Gamma \vdash f : S \to T \quad \Gamma \vdash s : S}{\Gamma \vdash fs : T}$$

$$(f =_{S \to T} g) \equiv (\forall x, y : S.x =_S y \to fx =_T gy)$$

Given groupoids G, H, there is a groupoid $G \to H$ of *groupoid functors* and *natural isomorphisms*

$$\frac{\Gamma, x : G \vdash t : H}{\Gamma \vdash \lambda x.t : G \to H} \qquad \frac{\Gamma \vdash f : G \to H \quad \Gamma \vdash s : G}{\Gamma \vdash fs : H}$$

$$(f =_{G \to H} g) \equiv (\forall x, y : G.x =_G y \to fx =_T gy)$$

Path Substitution

Given $a : A$, $e : A \simeq B$ and $b : B$, write $a \sim_e b$ for $a =_A e^{-}(b)$.

- The Programmers' Credo
- Introduction
- Logical Relations
- Not a New Idea?
- Past Work
- TDLR
- Propositions
- Sets
- Groupoids
- Properties of Equality
- Function types
- Path Substitution
- KIPLING
- KIPLING
- The Formalisation
- Results Verified
- Meaning Explanation for Homotopy Types?
- Conclusion
- References

Path Substitution

Given $a : A$, $e : A \simeq B$ and $b : B$, write $a \sim_e b$ for $a =_A e^-(b)$.

Given substitutions $\rho, \sigma : \Gamma \to \Delta$, a *path substitution* or *path* τ from ρ to σ $\tau : \rho \sim \sigma$, is given by:

- for every $x : T$ in Δ, a term $\Gamma \vdash \tau(x) : \rho(x) \sim_{T[[\tau]]} \sigma(x)$

Simultaneously, given $\tau : \rho \sim \sigma$ and $\Delta \vdash t : T$, define $\Gamma \vdash t[[\tau]] : t[\rho] \sim_{T[[\tau]]} t[\sigma]$.

KIPLING

McBride [McB10] introduced KIPLING:

- A standard approach: syntax first, then semantics

$$\mathsf{data}\ \mathsf{Context} : \mathsf{Set}$$
$$[\![\_]\!]\mathsf{C} : \mathsf{Context} \to \mathsf{Set}$$

$$\mathsf{data}\ \mathsf{Type} : \mathsf{Context} \to \mathsf{Set}$$
$$[\![\_]\!]\mathsf{T} : \forall\, \{\Gamma\} \to \mathsf{Type}\ \Gamma \to [\![\ \Gamma\ ]\!]\mathsf{C} \to \mathsf{Set}$$

$$\mathsf{data}\ \mathsf{Term} : \forall\ \Gamma \to \mathsf{Type}\ \Gamma \to \mathsf{Set}$$
$$[\![\_]\!]\mathsf{t} : \forall\, \{\Gamma\}\, \{A\} \to \mathsf{Term}\ \Gamma\ A \to (\gamma : [\![\ \Gamma\ ]\!]\mathsf{C}) \to [\![\ A\ ]\!]\mathsf{T}\ \gamma$$

KIPLING

McBride [McB10] introduced KIPLING:

- From [McB10]:

$$\mathsf{data}\ \mathsf{Context}_2 : \mathsf{Set}$$
$$[\![\_]\!]\mathsf{C}_2 : \mathsf{Context}_2 \to \mathsf{Set}$$

$$\mathsf{data}\ \mathsf{Type}_2 : \mathsf{Context}_2 \to \mathsf{Set}$$
$$[\![\_]\!]\mathsf{T}_2 : \forall\, \Gamma \to \mathsf{Type}_2\, \Gamma \to [\![\, \Gamma\,]\!]\mathsf{C}_2 \to \mathsf{Set}$$

$$\mathsf{data}\ \mathsf{Term}_2 : \forall\, \Gamma \to ([\![\, \Gamma\,]\!]\mathsf{C}_2 \to \mathsf{Set}) \to \mathsf{Set}$$
$$[\![\_]\!]\mathsf{t}_2 : \forall\, \{\Gamma\}\, \{S\} \to \mathsf{Term}_2\, \Gamma\, S \to (\gamma : [\![\, \Gamma\,]\!]\mathsf{C}_2) \to S\, \gamma$$

  Think of $\mathsf{Term}_2\, \Gamma\, S$ as the type of all terms t such that $\Gamma \vdash t : A$ for some A such that $[\![A]\!]\mathsf{T}_2 = S$.
- If s and t are definitionally equal in the object theory, then $[\![s]\!]$ and $[\![t]\!]$ are *definitionally* equal in Agda!

Postulate universes of groupoids. setoids and propositions.

Add computation rules to Agda

The Formalisation

data Cx : Set
$[\![_]\!]$C : Cx $\to$ Groupoid

data $\_\vdash_2\_$: $\forall\ \Gamma \to$ Fibration$_2$ $[\![\ \Gamma\ ]\!]$C $\to$ Set where

$[\![\_]\!]_2$: $\forall\ \{\Gamma\}\ \{T\} \to \Gamma \vdash_2 T \to$ Section$_2$ T

data Sub Γ : Cx $\to$ Set
$[\![_]\!]$S : $\forall\ \{\Gamma\}\ \{\Delta\} \to$ Sub $\Gamma\ \Delta \to$ Groupoid-Functor $[\![\ \Gamma\ ]\!]$C $[\![\ \Delta\ ]\!]$C

data PathSub $\{\Gamma\}$: $\forall\ \{\Delta\} \to$ Sub $\Gamma\ \Delta \to$ Sub $\Gamma\ \Delta \to$ Set
$[\![_]\!]$PS : $\forall\ \{\Gamma\ \Delta\}\ \{\rho\ \sigma$: Sub $\Gamma\ \Delta\} \to$ PathSub $\rho\ \sigma \to$
Groupoid-NatIso $[\![\ \rho\ ]\!]$S $[\![\ \sigma\ ]\!]$S

Results Verified

1. TLDR has sound semantics in Agda extended by the new computation rules.

Results Verified

1. TLDR has sound semantics in Agda extended by the new computation rules.
2. Path substitution is well defined.

Results Verified

1. TLDR has sound semantics in Agda extended by the new computation rules.
2. Path substitution is well defined.
3. If Agda with the new computation rules is consistent, then TDLR is consistent.

Results Verified

1. TLDR has sound semantics in Agda extended by the new computation rules.
2. Path substitution is well defined.
3. If Agda with the new computation rules is consistent, then TDLR is consistent.
4. Corollary: if Agda with equality reflection is consistent, then TDLR is consistent.

Results Verified

1. TLDR has sound semantics in Agda extended by the new computation rules.
2. Path substitution is well defined.
3. If Agda with the new computation rules is consistent, then TDLR is consistent.
4. Corollary: if Agda with equality reflection is consistent, then TDLR is consistent.

Future work:

- Normalisation by evaluation
- Calculation involving univalence: $K(G, 1)$
- Three-dimensional version

Meaning Explanation for Homotopy Types?

To know a proposition is to know how to construct a canonical proof.

Meaning Explanation for Homotopy Types?

To know a proposition is to know how to construct a canonical proof.
To know a set is to know how to consturct a canonical element, and to know the proposition $a = b$ for canonical elements a, b

Meaning Explanation for Homotopy Types?

To know a proposition is to know how to construct a canonical proof.

To know a set is to know how to consturct a canonical element, and to know the proposition $a = b$ for canonical elements a, b

To know a groupoid is to know how to construct a canonical object, and to know the set $a = b$ for canonical objects a, b

Etc.

Gives a meaning explanation for n-types for all finite n.

Meaning Explanation for Homotopy Types?

To know a proposition is to know how to construct a canonical proof.

To know a set is to know how to consturct a canonical element, and to know the proposition $a = b$ for canonical elements a, b

To know a groupoid is to know how to construct a canonical object, and to know the set $a = b$ for canonical objects a, b

Etc.

Gives a meaning explanation for n-types for all finite n.

To know a type is to know how to construct a canonical object, and to know the type $a = b$ for canonical objects a, b.

Conclusion

Thank you!

Source code available at: `github.com/radams78/TLDR`

References

[ABC16] Robin Adams, Marc Bezem, and Thierry Coquand. A normalizing computation rule for univalence in higher-order minimal logic. *CoRR*, abs/1610.00026, 2016. Submitted to proceedings of TYPES 2016.

[AMS07] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! In *PLPV'07*, 2007.

[Coq11] Thierry Coquand. Equality and dependent type theory, 2011.

[Gan56] Robin O. Gandy. On the axiom of extensionality — part i. *J. Symb. Logic*, 21(1):36–48, 1956.

[LH12] Daniel R. Licata and Robert Harper. Canonicity for 2-dimensional type theory. In *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '12, pages 337–348, New York, NY, USA, 2012. ACM.