Partial Univalence in n-truncated Type Theory

Christian Sattler [1] Andrea Vezzosi [2]

[1]University of Nottingham

[2]IT University of Copenhagen

Herrsching, 18 December 2019

UIP + UA $\vdash \bot$

Well known that Uniqueness of Identity Proofs contradicts Univalence:

- There are two distinct paths Bool = Bool:
- ua(id) and ua(not)
- ua(id)*true = true
- ua(not)*true = false

UIP + UA $\vdash \bot$

Well known that Uniqueness of Identity Proofs contradicts Univalence:

- There are two distinct paths Bool = Bool:
- ua(id) and ua(not)
- ua(id)*true = true
- ua(not)*true = false

Nevertheless UIP still appealing because it cuts down the complexity of paths.

- coherence data is at most finite.
- dealing with path algebra or building higher dimensional cubes is not very fun.

HoTT without UA

Even without univalence HoTT still contributes:

- The hierarchy of n-types.
  In particular propositions:
  $\mathrm{isProp}(A) := (x\ y : A) \to x = y$
- Higher Inductive Types: truncation, initial algebras, colimits.

These are very useful even in a type theory where we assume every type is truncated.

We still need *some* Univalence

When working with HITs we often use univalence for hprops.

- Membership for Kuratowski Finite Sets defined by recursion: e.g. need to prove $a \in (x \cup y) = a \in (y \cup x)$
- Effectiveness of quotients: $[a] =_{A/R} [b] \simeq R(a, b)$
- ...

So even with UIP it is desirable to have

$\mathsf{ua}_{-1} : \mathsf{isProp}(A) \to \mathsf{isProp}(B) \to A \simeq B \to A = B$

Partial Univalence is consistent with UIP

We build a model of a type theory with

- $\Pi, \Sigma, W, \mathbb{N}, \mathsf{U}_0, \mathsf{U}_1, \ldots$
- n-truncated Higher Inductive Types
- n-truncatedness for every type.
- univalence restricted to (n-1)-types.

The model interprets types as elements of universes V_ℓ which are constructed as HITs, and proven n-truncated by an encode/decode argument.

The UIP case is done by taking $\mathsf{n} = 0$.

Also in the following we assume $n \geq 0$.

Starting Small

In HoTT with Higher Induction Recursion:

data V : U

- $\overline{\Pi} : (A : \mathsf{V}) \to (B : \mathsf{El}(A) \to \mathsf{V}) \to \mathsf{V}$
- $\mathsf{univ}_{A,B:\mathsf{V}} : \mathsf{type}_{(n\text{-}1)}(\mathsf{El}(A)) \to \mathsf{type}_{(n\text{-}1)}(\mathsf{El}(B)) \to \mathsf{El}(A) \simeq \mathsf{El}(B) \to A = B$

$\mathsf{El} : \mathsf{V} \to \mathsf{U}$

- $\mathsf{El}(\overline{\Pi}\ A\ B) = (x : \mathsf{El}(A)) \to \mathsf{El}(B(x))$
- $\mathsf{El}(\mathsf{univ}_{A,B}\ A_{n\text{-}1}\ B_{n\text{-}1}\ e) = \mathsf{ua}(e)$

univ is enough to recover univalence for $(n\text{-}1)$-types:

$\mathsf{type}_{(n\text{-}1)}(\mathsf{El}(A)) \to \mathsf{isContr}(\Sigma\,(B : \mathsf{V}).\mathsf{El}(B) \simeq \mathsf{El}(A))$

Indexed Family InV

To discuss equality in V it's actually more convenient to work with an indexed type.

data $\mathsf{InV} : \mathsf{U}_0 \to \mathsf{U}_1$

- $\overline{\Pi}_{A,B} : \mathsf{InV}(A) \to ((a : A) \to \mathsf{InV}(B(a))) \to \mathsf{InV}((a : A) \to B(a))$
- $\mathsf{univ}_{A,B} : \mathsf{type}_{(n\text{-}1)}(A) \to \mathsf{type}_{(n\text{-}1)}(B) \to (\overline{A} : \mathsf{InV}(A))\ (\overline{B} : \mathsf{InV}(B)) \to (e : A \simeq B) \to \overline{A} =_{\mathsf{InV}(\mathsf{ua}(e))} \overline{B}$

$\mathsf{V} = \Sigma(A : \mathsf{U}_0).\mathsf{InV}(A)$

$\mathsf{El} : \mathsf{V} \to \mathsf{U}_0$
$\mathsf{El}(A, \_) = A$

Indexed Family InV

To discuss equality in V it's actually more convenient to work with an indexed type.

data $\mathsf{InV} : \mathsf{U}_0 \to \mathsf{U}_1$

- $\overline{\Pi}_{A,B} : \mathsf{InV}(A) \to ((a : A) \to \mathsf{InV}(B(a))) \to \mathsf{InV}((a : A) \to B(a))$
- $\mathsf{univ}_{A,\cancel{B}} : \mathsf{type}_{(n\text{-}1)}(A) \to \cancel{\mathsf{type}_{(n\text{-}1)}(B)} \to (\overline{A} : \mathsf{InV}(A))\ (\overline{B} : \mathsf{InV}(A)) \to \cancel{(e : A \simeq B)} \to \overline{A} =_{\mathsf{InV}(A)} \overline{B}$

$\mathsf{V} = \Sigma(A : \mathsf{U}_0).\mathsf{InV}(A)$

$\mathsf{El} : \mathsf{V} \to \mathsf{U}_0$
$\mathsf{El}(A, \_) = A$

Univalence: propositional codes

$\mathsf{data}\ \mathsf{InV} : \mathsf{U}_0 \to \mathsf{U}_1$

- $\overline{\Pi}_{A,B} : \mathsf{InV}(A) \to ((a : A) \to \mathsf{InV}(B(a))) \to \mathsf{InV}((a : A) \to B(a))$
- $\mathsf{univ}_A : \mathsf{type}_{(n\text{-}1)}(A) \to (\overline{A} : \mathsf{InV}(A))\ (\overline{B} : \mathsf{InV}(A)) \to \overline{A} =_{\mathsf{InV}(A)} \overline{B}$

$\mathsf{V} = \Sigma(A : \mathsf{U}_0).\mathsf{InV}(A)$

$\mathsf{El} : \mathsf{V} \to \mathsf{U}_0$
$\mathsf{El}(A, \_) = A$

Given $X : \mathsf{V}$ we will write $\overline{X}$ for the second projection.

Equality in V

We wish to show $\mathsf{type}_{(n\text{-}1)}(X =_{\mathsf{V}} Y)$

Of course $X =_{\mathsf{V}} Y \simeq \Sigma(p : \mathsf{El}(X) = \mathsf{El}(Y)).\ \overline{X} =_{\mathsf{InV}(p)} \overline{Y}$.

Two main cases for $\overline{X} =_{\mathsf{InV}(p)} \overline{Y}$:

1. $\mathsf{type}_{(n\text{-}1)}(\mathsf{El}(X))$
2. $\overline{\Pi}\, S_0\ T_0 =_{\mathsf{InV}(p)} \overline{\Pi}\, S_1\ T_1$

In case (1) we know $\mathsf{isContr}(\overline{X} =_{\mathsf{InV}(p)} \overline{Y})$ by univ.

Case (2) more involved.

$\overline{\Pi}_{A_0,B_0}\, S_0\; T_0 =_{\mathsf{InV}(p)} \overline{\Pi}_{A_1,B_1}\, S_1\; T_1$

Again two cases: equal structurally or equal (n-1)-types.

- $\mathrm{StrEq}(S_0, T_0, S_1, T_1, p) :=$
  - $a : A_0 = A_1$
  - $s : S_0 =_{\mathsf{InV}(a)} S_1$
  - $b : B_0 =_{a \to \mathsf{U}_0} B_1$
  - $t : T_0 =_{(x:a) \to \mathsf{InV}(b(x))} T_1$
  - $coh : p = (x : a) \to b(x)$
- $\mathrm{type}_{(n\text{-}1)}((x : A_0) \to B_0(x))$

The equality will be equivalent to the **join** of the two.

Note that the join is contractible when $\mathrm{type}_{(n\text{-}1)}((x : A_0) \to B_0(x))$.

encode/decode proof

Define:

- $\mathsf{Eq} : (X_0\, X_1 : \mathsf{V}) \to \mathsf{El}(X_0) =_{\mathsf{U}_0} \mathsf{El}(X_1)$
  $\to \Sigma(E : \mathsf{U}_1).\ \mathsf{type}_{(n\text{-}1)}(\mathsf{El}(X_0)) \to \mathsf{isContr}(E)$

  by double induction on the InV arguments, where the return type is contractible in the univ cases.

- $\mathsf{encode}_{X_0,X_1,p} : \overline{X}_0 =_{\mathsf{InV}(p)} \overline{X}_1 \to \mathsf{Eq}(X_0, X_1, p)$

  by path induction and then induction on the code.

- $(c : \mathsf{Eq}(X_0, X_1, p)) \to \mathsf{fiber}(\mathsf{encode}_{X_0,X_1,p}, c)$

  i.e., there is a decode function, right inverse of encode.

The whole equivalence then follows from contractibility of singletons.

$\mathrm{type}_n(\mathrm{V})$

Show $\mathrm{type}_{(n\text{-}1)}(\Sigma(p : \mathrm{El}(X_0) = \mathrm{El}(X_1)).\mathrm{Eq}(\overline{X}_0, \overline{X}_1, p))$.

We proceed by induction on $\overline{X}_0$ and $\overline{X}_1$.
Only relevant case when they are both $\overline{\Pi}$:

$$\mathrm{type}_{(n\text{-}1)}(\Sigma(p : \mathrm{El}(X_0) = \mathrm{El}(X_1)).\ \mathrm{type}_{(n\text{-}1)}(\mathrm{El}(X_0)) \star \mathrm{StrEq}(S_0, T_0, S_1, T_1, p))$$

where $X_i = (x : A_i) \to B_i(x)$

By abstract nonsense it is sufficient to show:

- $\mathrm{type}_{(n\text{-}1)}(\Sigma(p : \mathrm{El}(X_0) = \mathrm{El}(X_1)).\ \mathrm{type}_{(n\text{-}1)}(\mathrm{El}(X_0)))$
- $\mathrm{type}_{(n\text{-}1)}(\Sigma(p : \mathrm{El}(X_0) = \mathrm{El}(X_1)).\ \mathrm{StrEq}(S_0, T_0, S_1, T_1, p))$

The first follows from univalence.

$\mathrm{type}_n(\mathrm{V})$

To show

$\mathrm{type}_{(n\text{-}1)}(\Sigma(p : \mathrm{El}(X_0) = \mathrm{El}(X_1)).\mathrm{StrEq}(S_0, T_0, S_1, T_1, p))$

we observe the type itself consists of these five components:

- $p : \mathrm{El}(X_0) = \mathrm{El}(X_1)$
- $a : A_0 = A_1$
- $s : S_0 =_{\mathrm{InV}(a)} S_1$
- $b : B_0 =_{a \to \mathrm{U}_0} B_1$
- $t : T_0 =_{(x:a) \to \mathrm{InV}(b(x))} T_1$
- $coh : p = (x : a) \to b(x)$

p and coh form a contractible pair, while each of (a, s) and (b, t) are handled by induction hypothesis.

More type formers

The construction generalizes to an indexed container describing the type formers, and to different universes.

- $S : \mathsf{U}_0 \to \mathsf{U}_1$
- $P_A : S(A) \to \mathsf{U}_0 \to \mathsf{U}_1$

$$\mathrm{Ext}_{S,P}(X, A) := \Sigma(s : S(A)).((R : \mathsf{U}_0)(p : P_A(s, R)) \to X\ R)$$

$\mathsf{data}\ \mathsf{InV} : \mathsf{U}_0 \to \mathsf{U}_1$

- $\mathsf{con}_A : \mathrm{Ext}_{S,P}(\mathsf{InV}, A) \to \mathsf{InV}(A)$
- $\mathsf{univ}_A : \mathrm{type}_{(n\text{-}1)}(A) \to (\overline{A}_0 : \mathsf{InV}(A))\,(\overline{A}_1 : \mathsf{InV}(A))$
  $\to \overline{A}_0 =_{\mathsf{InV}(A)} \overline{A}_1$

As long as the extension of the container preserves total types being n-truncated, in a specific way.

Preserving truncatedness

During the proof that $\mathrm{type}_n(\mathrm{V})$ we need this:

Given

- $(s, t) : \mathrm{Ext}_{S,P}(\mathrm{InV}, A)$
- $(R_0\, R_1 : \mathrm{U}_0)\,(p_0 : P_A(s, R_0))\,(p_1 : P_A(s, R_1)) \to \mathrm{type}_{(n\text{-}1)}(\Sigma(r : R_0 = R_1).\ t\, R_0\, p_0 =_{\mathrm{InV}(r)} t\, R_1\, p_1)$.

It follows that

$$\mathrm{type}_{(n\text{-}1)}((A, s, t) =_{\Sigma(A:\mathrm{U}_0).\mathrm{Ext}_{S,P}(\mathrm{InV},A)} (A, s, t))$$

In the case for Π types this property corresponds to the step about $\mathrm{StrEq}(S_0, T_0, S_1, T_1, p)$.

This condition only refers to the arguments of the type former, so truncated HITs can be added without worrying about how complex the constructors are.

Example: code for truncated PushOuts

Having (truncated) pushouts in V would mean adding a code with this signature:

$$\overline{\mathrm{PO}} : (A\ B\ C : \mathrm{V}) \to \\ (f : \mathrm{El}(A) \to \mathrm{El}(B)) \to (g : \mathrm{El}(A) \to \mathrm{El}(C)) \to \mathrm{V}$$

Equalities regarding the fields A, B, C are handled by induction hypothesis as in the case for Π types.

Equalities regarding f and g are already $(n\text{-}1)$-truncated because the types are n-truncated.

A Universe Hierarchy

Given a selection of type formers,

repeat $\mathsf{InV} : \mathsf{U}_0 \to \mathsf{U}_1$ construction for universe U_ℓ:

$\mathsf{InV}_\ell : \mathsf{U}_\ell \to \mathsf{U}_{\ell+1}$

adding codes for the previous universes:

$v_0 : \mathsf{InV}_\ell(\mathsf{V}_0), \ldots, v_{\ell-1} : \mathsf{InV}_\ell(\mathsf{V}_{\ell-1})$

here we rely on $\mathsf{V}_0, \ldots, \mathsf{V}_{\ell-1}$ being n-truncated.

A Universe Hierarchy

Given a selection of type formers,

repeat $\mathsf{InV} : \mathsf{U}_0 \to \mathsf{U}_1$ construction for universe U_ℓ:

$\mathsf{InV}_\ell : \mathsf{U}_\ell \to \mathsf{U}_{\ell+1}$

adding codes for the previous universes:

$v_0 : \mathsf{InV}_\ell(\mathsf{V}_0), \ldots, v_{\ell-1} : \mathsf{InV}_\ell(\mathsf{V}_{\ell-1})$

here we rely on $\mathsf{V}_0, \ldots, \mathsf{V}_{\ell-1}$ being n-truncated.

We obtain a hierarchy of universes $\mathsf{V}_0, \mathsf{V}_1, \ldots$

with corresponding $\mathsf{lift}_\ell : \mathsf{V}_\ell \to \mathsf{V}_{\ell+1}$ functions providing explicit cumulativity.

Model Partially Univalent n-Truncated type theory

Let $\mathcal{C} = (\mathsf{Cxt}_{\mathcal{C}}, \mathsf{Ty}^{\ell}_{\mathcal{C}}(\Gamma), \mathsf{Tm}_{\mathcal{C}}(\Gamma, A))$ be a stratified CwF.

Assume that $\mathcal{C}$ models univalence and Indexed HITs.

Then the preceding construction gives types V_{ℓ} in $\mathcal{C}$.

We define a stratified CwF $\mathcal{D}$ where types are elements of V_{ℓ}:

- $\mathsf{Cxt}_{\mathcal{D}} := \mathsf{Cxt}_{\mathcal{C}}$
- $\mathsf{Ty}^{\ell}_{\mathcal{D}}(\Gamma) := \mathsf{Tm}_{\mathcal{C}}(\Gamma, \mathsf{V}_{\ell})$
- $\mathsf{Tm}_{\mathcal{D}}(\Gamma, X) := \mathsf{Tm}_{\mathcal{C}}(\Gamma, \mathsf{El}(X))$

Then $\mathcal{D}$ models univalence restricted to (n-1)-types and the axiom that every type is n-truncated.

Consistency follows by picking a $\mathcal{C}$, e.g., a variant of cubical sets.

Does it compute?

If we construct V in a Cubical Type Theory with

$\mathsf{Glue}\, A\, [\varphi \vdash (T, e)] : \mathsf{U}$

we automatically get Glue restricted to (n-1)-types in V:

$\mathsf{Glue}\, A\, A_{n\text{-}1}\, [\varphi \vdash (T, e)] : \mathsf{V}$

where A and T are in V and $A_{n\text{-}1} : \mathsf{type}_{n\text{-}1}(\mathsf{El}(A))$

Introducing a witness of n-truncatedness for the universes will require more work.

Conclusion

- It's fine to assume both h-prop extensionality and UIP.
- In fact it's fine to assume (n-1)-univalence and n-truncatedness.

Conclusion

- It's fine to assume both h-prop extensionality and UIP.
- In fact it's fine to assume (n-1)-univalence and n-truncatedness.

Probably one more model construction and we can close this Agda issue: