The Constructive Model of Univalence in Cubical Sets

Literature review

W. Vanhulle[1] A. Nuyts[2] D. Devriese[3]

[1]Student

[2]Supervisor

[3]Promoter

45 min. public seminar

Outline

Equality in mathematics

A mathematician is asked by a friend who is a devout Christian: "Do you believe in one God?"

What does he reply?

Equality in mathematics

A mathematician is asked by a friend who is a devout Christian: "Do you believe in one God?"

What does he reply?

He answers: "Yes - up to isomorphism." (© Michael Benjamin Stepp)
[Two isomorphic but not equal things]
[Two other isomorphic things]

Algebraic Topology

Isomorphism in algebraic topology is "homotopy equivalence"

figures/mug.jpg

Figure: A mug

figures/donut.jpg

Figure: A donut

homotopy equivalent spaces have the same "number of n-dimensional holes"

Holes are homtopy groups

computed by looking at homotopy classes of embeddings

$$S^n \to X, \quad \text{or } [0,1]^n \to X$$

figures/loops.png

Figure: A donut has two 1-dimensional homotopy classes.

figures/basketball_hollow.png

Figure: A ball without center has two 2-dimensional homotopy classes.

Type theory

Two meanings/subfields:

- verifying computation in programming languages

```
filter :: (a -> Bool) -> [a] -> [a]
filter _pred []    = []
filter pred (x:xs)
  | pred x         = x : filter pred xs
  | otherwise      = filter pred xs
```

Figure: A typed recursive function in Haskell

- alternative constructive foundation of mathematics

```
_∘_ :   (∀ {x} (y : B x) → C y) → (g : (x : A) → B x) →
        ((x : A) → C (g x))
f ∘ g = λ x → f (g x)
```

Figure: Definition of the topological space S^1 in Agda

Type theory

Two meanings/subfields:

- verifying computation in programming languages

```
filter :: (a -> Bool) -> [a] -> [a]
filter _pred []    = []
filter pred (x:xs)
  | pred x         = x : filter pred xs
  | otherwise      = filter pred xs
```

Figure: A typed recursive function in Haskell

- alternative constructive foundation of mathematics

```
_∘_ :   (∀ {x} (y : B x) → C y) → (g : (x : A) → B x) →
        ((x : A) → C (g x))
f ∘ g = λ x → f (g x)
```

Figure: Definition of the topological space S^1 in Agda

Type theory as foundation of mathematics

Russel, 1907

Invented to prevent paradox:

$$R = \{x \mid x \notin x\}, R \in R \Leftrightarrow \notin R$$

Solution was:

- replace sets (and propositions) by types and elements by terms,

$$x \in R \Rightarrow x : R$$

- types belong to universe hierarchy

$$\exists i, R : \mathscr{U}_i, \quad \mathscr{U}_0 : \mathscr{U}_1 : \ldots$$

- constructive logic and formation rules

$x \notin x$ is not a valid proposition anymore

equality type

Martin-Löf, 1984

Leibniz's principle (axiom):

$$f = g \Leftrightarrow f(x) = g(x), \forall x$$

Not an axiom in type theory for decidability reasons.

Equality "$=$" added as a type called *propositional equality*:

- weaker than equality "by definition".
- stronger than equivalence.

equality type

Martin-Löf, 1984

Leibniz's principle (axiom):

$$f = g \Leftrightarrow f(x) = g(x), \forall x$$

Not an axiom in type theory for decidability reasons.
Equality "$=$" added as a type called *propositional equality*:

- weaker than equality "by definition".
- stronger than equivalence.

Identity eliminator

Martin-Löf, 1984

Elimination rule of equality type:

Definition (path induction)

Given the following terms:

- a predicate $C : \prod_{x,y:A} (x =_A y) \to \mathscr{U}$
- the base step $c : \prod_{x:A} C(x, x, \mathtt{refl}_x)$

there is a function $f : \prod_{x,y:A} \prod_{p:x=_A y} C(x, y, p)$ such that $f(x, x, \mathtt{refl}_x) \equiv c(x)$.

(describes how to compute with the equality type)

Identity eliminator

Martin-Löf, 1984

Elimination rule of equality type:

Definition (path induction)

Given the following terms:

- a predicate $C : \prod_{x,y:A} (x =_A y) \to \mathscr{U}$
- the base step $c : \prod_{x:A} C(x, x, \mathtt{refl}_x)$

there is a function $f : \prod_{x,y:A} \prod_{p:x=_A y} C(x, y, p)$ such that $f(x, x, \mathtt{refl}_x) \equiv c(x)$.

(describes how to compute with the equality type)

Intuition identity eliminator

Role of eliminator:

> *To prove a property C that depends on terms x, y and equalities $p : x = y$ it suffices to consider all the cases where*
>
> - *x is definitionally equal to y*
> - *the term of the intensional equality type under consideration is $\texttt{refl}_x : x = x$.*

Implications:

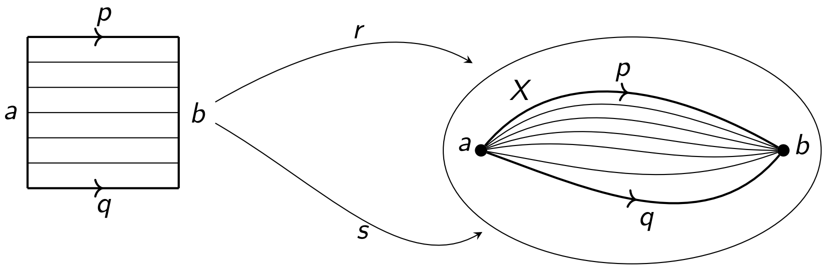
- proves transitivity, symmetry
- equality type can have multiple terms

Homotopy type theory (HoTT)

Awodey, 2006

Gives *homotopy* interpretation to equality type:

$$p, q : a =_X b \quad r, s : p =_{\mathtt{Id}_X(a,b)} q$$



$\Rightarrow$ alternative foundations for mathematics based on type theory and topology

Role of univalence

Voevodsky, 2009

Axiom (Univalence axiom)

Given types $X, Y : \mathscr{U}$ for some universe $\mathscr{U}$, the map $\Phi_{X,Y} : (X = Y) \to (X \simeq Y)$ is an equivalence of types.

- equivalence of types is a bijection for set-like types

$$\mathbb{N} \simeq \mathbb{N}_0$$

- univalence implies

$$\mathbb{N} \simeq \mathbb{N}_0 \Rightarrow \mathbb{N} = \mathbb{N}$$

- forces multiple terms of equality

$\Rightarrow$ *terms of equality are like paths*

Role of univalence

Voevodsky, 2009

Axiom (Univalence axiom)

Given types $X, Y : \mathscr{U}$ for some universe $\mathscr{U}$, the map $\Phi_{X,Y} : (X = Y) \to (X \simeq Y)$ is an equivalence of types.

- equivalence of types is a bijection for set-like types

$$\mathbb{N} \simeq \mathbb{N}_0$$

- univalence implies

$$\mathbb{N} \simeq \mathbb{N}_0 \Rightarrow \mathbb{N} = \mathbb{N}$$

- forces multiple terms of equality

$\Rightarrow$ *terms of equality are like paths*

Consequences of path interpretation

in general:

- equivalence behaves like homotopy equivalence
- mathematics "up to homotopy"
- lifting of path p as in algebraic topology: *transport* gives the other ending point of $p_\star$

figures/cover.png

Figure:

Origin univalence

Grayson, 2018

`figures/voevodsky.jpg`

Figure: Vladimir Voevodsky

Why "univalent"?

... these foundations seem to be faithful to the way in which I think about mathematical objects in my head ...

faithful = univalent in a Russion translation of Boardman (2006)

Origin univalence

Grayson, 2018

figures/voevodsky.jpg

Figure: Vladimir Voevodsky

Why "univalent"?

> *... these foundations seem to be faithful to the way in which I think about mathematical objects in my head ...*

faithful = univalent in a Russion translation of Boardman (2006)

Computing with univalence

Huber, 2015

What about:

- implementing HoTT?
- calculations with very simple types as $\mathbb{N}$?

Can we, given a term $t : \mathbb{N}$ constructed using the univalence axiom, construct two terms $u : \mathbb{N}$ and $p : t =_{\mathbb{N}} u$ such that u does not involve the univalence axiom?

$\Rightarrow$ canonicity of $\mathbb{N}$ in cubical type theory (CTT)

$$t \equiv ua(...) \rightsquigarrow u \equiv S(\dots(0)\dots) : \mathbb{N}$$

Computing with univalence

Huber, 2015

What about:

- implementing HoTT?
- calculations with very simple types as $\mathbb{N}$?

> *Can we, given a term $t : \mathbb{N}$ constructed using the univalence axiom, construct two terms $u : \mathbb{N}$ and $p : t =_{\mathbb{N}} u$ such that u does not involve the univalence axiom?*

$\Rightarrow$ canonicity of $\mathbb{N}$ in cubical type theory (CTT)

$$t \equiv ua(\ldots) \rightsquigarrow u \equiv S(\ldots(0)\ldots) : \mathbb{N}$$

Cubical type theory

Cohen et al., 2015

Dimension variables $i, j, k : \mathbb{I}$ as primitives:

`figures/cubes.png`

Figure: Discrete "*n*-cubes". Huber (2016)

- univalence becomes *constructable*
- computational interpretation for univalence

Are cubes a good idea?

EnigmaChord, 2016

figures/cube-joke.jpg

Figure:

(yes, they model n-dimensional homotopies)

Are cubes a good idea?

EnigmaChord, 2016

figures/cube-joke.jpg

Figure:

(yes, they model n-dimensional homotopies)

Cubes model homotopy

Altenkirch, Brunerie, Licata, et. al 2013

Homotopy groups are defined as equivalence classes:

$$[0,1]^n \rightarrow X$$

In homotopy type theory:

higher-dimensional paths $\cong$ *higher-dimensional eqalities.*

Cubes give discrete description of higher-dimensional paths:

- equalities $\Rightarrow$ edges of cubes
- equalities between equalities (homotopies) $\Rightarrow$ faces of cubes, ...

Cubes model homotopy

Altenkirch, Brunerie, Licata, et. al 2013

Homotopy groups are defined as equivalence classes:

$$[0,1]^n \to X$$

In homotopy type theory:

> *higher-dimensional paths* $\cong$ *higher-dimensional eqalities.*

Cubes give discrete description of higher-dimensional paths:

- equalities $\Rightarrow$ edges of cubes
- equalities between equalities (homotopies) $\Rightarrow$ faces of cubes, ...

Operations on cubes

Bezem, Coquand, Huber et al., 2013

Necessary for modelling HoTT:

- composition $\Rightarrow$ equality type
- glueing $\Rightarrow$ univalence

figures/kan.jpg

Figure: Daniel Kan

figures/extension.png

Figure: Adding the lid with composition. Huber (2016)

Topics touched in my thesis

Literature review about:

- cubes as base categories for a model of HoTT
- the proof of univalence in this model
- some applications and alternative models

figures/groups.png

Presheaves on $\mathscr{C}$

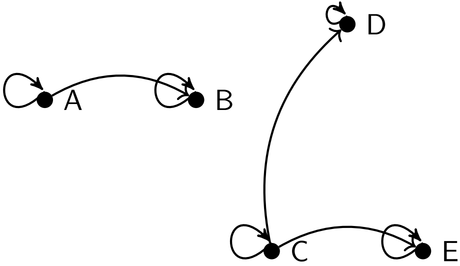
Hofstra, 2014

Maps (contravariant functors) $\mathscr{C} \to$ **Set** denoted by $\hat{\mathscr{C}}$

- generalize sheaves (see sheafication)
- model type theories, Dybjer (1994)

Example (Reflexive directed graphs)

Take $\mathscr{C} = \{0, 1\}$ and $\mathrm{Hom}_{\mathscr{C}} = \{B, E, R\}$



Presheaf model on $\mathscr{C}$

Dybjer, 1994

Gives interpretations for stuff in type theory:

- contexts Γ are the category $\widehat{\mathscr{C}}$
- types are a presheaf

$$\widehat{\int_{\mathscr{C}} \Gamma}$$

- terms are elements of

$$\prod_{I \in \mathscr{C}, \rho \in \Gamma(I)} A(I, \rho)$$

Bonuses:

- verify consistency of type theory in sets
- find primitives for implementations

Presheaf model on $\mathscr{C}$

Dybjer, 1994

Gives interpretations for stuff in type theory:

- contexts Γ are the category $\widehat{\mathscr{C}}$
- types are a presheaf

$$\widehat{\int_{\mathscr{C}} \Gamma}$$

- terms are elements of

$$\prod_{I \in \mathscr{C}, \rho \in \Gamma(I)} A(I, \rho)$$

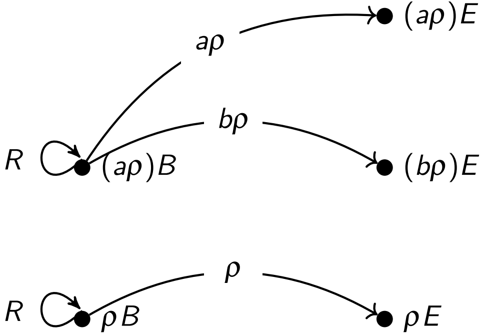
Bonuses:

- verify consistency of type theory in sets
- find primitives for implementations

Types with presheaves

Example

A type A over $\mathscr{C} = \{0,1\}$ is a refined graphs, terms $a, b : A$ are edges in refined graph:



Distributive lattice

Let $i, j, k, \ldots \in \mathbb{A}$ countable = "dimension variables"

Definition (Free De Morgan algebra)

Distributive lattice containing:

- $i \wedge j$ (min of i, j)
- $i \vee 0$ (computes to i)
- $\neg k$ (negation)
- de Morgan rules.

… denoted by $dM(i, j, k, \ldots)$

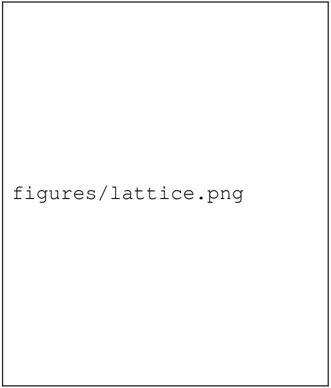


Figure: A distributive lattice

Definition (cube category $\Box$)

objects are $\{I \mid |I| < \infty, I \subset \mathbb{A}\}$ morphisms $J \to I$ are maps $I \mapsto dM(J)$

Example (presheaf model on $\Box$)

contexts Γ are presheaves $\Box \to \mathbf{Set}$ shaped by lattice structure $\Rightarrow$ complicated!

figures/context.png

Figure: a context Γ applied to $\{i, j\}$

Types in $\widehat{\square}$

figures/types.png

Figure: A type A within context Γ. Huber (2016)

Types in presheaf models

In the presheaf model on $\square$:

- types more complicated
- types no longer simply nested graphs

Lemma (Characterization of types)

If $\Gamma \in \widehat{\mathscr{C}}$*, then*

$$Ty(\Gamma) \cong \left\{ (\Delta, \sigma) \mid \Delta \in \widehat{\mathscr{C}}, \sigma \in Hom_{Ctx}(\Delta, \Gamma) \right\}$$

Interpreting types in presheaf model $\square$ hard but possible!

Types in presheaf models

In the presheaf model on $\square$:

- types more complicated
- types no longer simply nested graphs

Lemma (Characterization of types)

If $\Gamma \in \widehat{\mathscr{C}}$, *then*

$$Ty(\Gamma) \cong \left\{(\Delta, \sigma) \mid \Delta \in \widehat{\mathscr{C}}, \sigma \in Hom_{Ctx}(\Delta, \Gamma)\right\}$$

Interpreting types in presheaf model $\square$ hard but possible!

Path type

Bezem, Coquand, 2013

Syntactical definition of `Path` type with typing rules:

$$\frac{i : \mathbb{I} \vdash t : A \qquad i : \mathbb{I} \vdash t(i/0) = a : A \qquad i : \mathbb{I} \vdash t(i/1) = b : A}{() \vdash \langle i \rangle\, t : \texttt{Path}\ a\ b}$$

- almost models equality type
- not necessarily transitive $\Rightarrow$ composition operation

$$\begin{tikzcd}
a \arrow[r, dashed] & c \\
a \arrow[u, "refl"] \arrow[r, "p\ i"] & b \arrow[u, "q\ j"']
\end{tikzcd}$$

Figure: Transitivity can be proven with composition operation.

Constructive model of type theory

Other types can be interpreted in the presheaf model $\widehat{\square}$

- product, sum types
- natural numbers

$\Rightarrow$ Constructive model for type theory

What about univalence and HoTT?

$\Rightarrow$ Simplicial sets model univalence + glueing construction, Kapulkin (2012)

figures/simplex.png

Figure: Simplicial complex

Constructive model of type theory

Other types can be interpreted in the presheaf model $\widehat{\square}$

- product, sum types
- natural numbers

$\Rightarrow$ Constructive model for type theory

What about univalence and HoTT?

$\Rightarrow$ Simplicial sets model univalence + glueing construction, Kapulkin (2012)

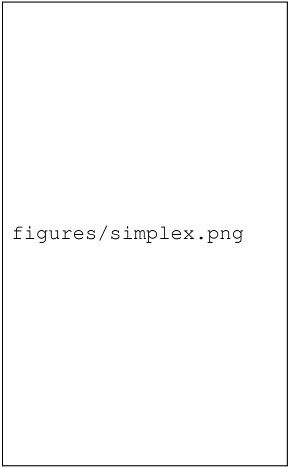


Figure: Simplicial complex

Partial types

Partial types are only defined on subpolyhedra of cubes:

figures/types_side.png

Figure: The partial type $A(i/0)$ in type A, syntactically:
$A\ [(i=0) \mapsto A(i/0)]$.

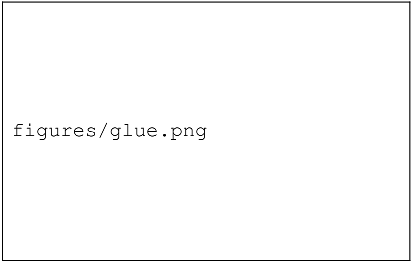
Glueing equivalences

Equivalences are a crucial ingredient for the univalence axiom:

- maps $f\colon T \to A$ that correspond to homotopy equivalences
- have inverse up-to some paths

The `Glue` type was introduced to prove univalence:

- very complicated typing rules
- glue equivalences over partial types together:



Proving univalence

Axiom (Univalence axiom)

Given types $X, Y : \mathscr{U}$ *for some universe* $\mathscr{U}$*, the map* $\Phi_{X,Y} : (X = Y) \to (X \simeq Y)$ *is an equivalence of types.*

Proof.

- The existence of a map $\mathtt{ua} : (X \simeq Y) \to (X = Y)$ proven with `Glue` construction:

  $$i : \mathbb{I} \vdash E = \mathtt{Glue}\,[(i = 0) \mapsto (X, f), (i = 1) \mapsto (Y, \mathtt{id}_Y)]\; Y$$

  E is a path (equality) from X to Y.
- Remainder proven with "contractibility of singletons".

□

Applying `ua` from univalence

Example (Monoids)

$$M_1 \equiv (\mathbb{N}, (m,n) \mapsto m+n, 0)$$

and

$$M_2 \equiv (\mathbb{N}_0, (m,n) \mapsto m+n-1, 1)$$

- are isomorphic by
$$\lambda\, n \to n+1$$
- (path-) equal in CTT

Definition of a ~~monoid~~ magma

setoid encoding uses operator "·" and equivalence "≈":

```
notZero n = Σ ℕ (λ m → (n ≡ (suc m)))
ℕ₀ = Σ ℕ (λ n → notZero n)

op₂ : Op₂ ℕ₀
op₂ (x , p)  (y , q) =
    (predℕ (x + y) , (predℕ (predℕ (x + y)) , sumLem x y p q) )

M₂ : Algebra.Magma _ _
M₂ = record {
  Carrier = ℕ₀ ;
  _≈_ = (_≡_) ;
  _•_ = op₂ ;
  isMagma = ... ,
  }
```

Equality of carrier sets

$\mathbb{N} \to \mathbb{N}_0 : n \mapsto n+1$ is bijection

- is equivalence of types
- `ua` returns equality `ℕ ≡ ℕ₀`

```
f : ℕ → ℕ₀
f n = (suc n , ( n , refl ) )
...
fEquiv : ℕ ≃ ℕ₀
fEquiv = (f ,  isoToIsEquiv (iso f g l' r'))

fEq : ℕ ≡ ℕ₀
fEq i = ua fEquiv i
```

Equality of ~~monoids~~ magmas

Defined for every component of record type:

```
mPath : s₁ ≡ s₂
mPath = λ i → record {
   Carrier = (fEq i) ;
   _≈_ =  _≡_;
   _•_ = transOp' i ;
   isMagma = record {
    isEquivalence = ≡equiv  ;
     •-cong   = ?
     }
   }
```

`transOp'` defined by transporting along $\mathbb{N} \equiv \mathbb{N}_0$, proofs can be transported over $s_1 \equiv s_2$.

In algebraic topology

Homotopy groups compute number of higher-dimensional holes

Theorem

$\pi_4(S^3) \cong \mathbb{Z}_n$ *for* $n = 2$

- proven in HoTT with univalence
- n implemented in CTT as a function
- canonicity predicts termination

(bug in Agda or CTT prevents evaluation)

figures/loops.png

Figure: from science4all

Other research

Licata, Harper, Cavallo, Orton et al., 2018

- computational type theory is an alternative implementation
- composition operation may not be necessary
- alternatives to complicated glue types: fundamental axioms and language of topoi

Summary

- HoTT redefines equality
- CTT implements HoTT
- HoTT can be verified in computers

Thanks for watching!

For Further Reading I