# 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

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

*What does he reply?*

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)

`figures/isomorphism.jpg`

Figure: Monkeys looking at isomorphic monkeys.

# 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 homotopy groups

computed by looking at homotopy classes of *continous* 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_hollo

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

# Type theory's origin

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

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

- constructive logic and formation rules

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

# 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 for mathematics

Deductive system of judgements with typing rules:

$$\frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash b : B}$$

- judgments express wether a type is inhabited
- all judgements have contexts
- typing rules tell how to form and combine types and terms

# Equality in type theory

## Martin-Löf, 1984

Definitional equality in type theory is for type checking, "denoted =" in code:

```
data Nat : Set where
  zero : Nat
  suc  : (n : Nat) → Nat

_+_ : Nat → Nat → Nat
zero  + m = m
suc n + m = suc (n + m)
```

Figure: An example of definitional equality.

Mathematics needs a "softer" equality as in:

Example (Leibniz's extensionality principle)

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

# Identity type

Martin-Löf, 1984

> *... introduction of propositional equality in form of "identity type"*

## Definition (Introduction rule)

Given a $a : X$, $\mathtt{refl}(a) : a = a$.

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 $f(x, x, \mathtt{refl}_x) \equiv c(x)$.

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

# 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 $\mathtt{refl}_x : x = x$.*

Implications:

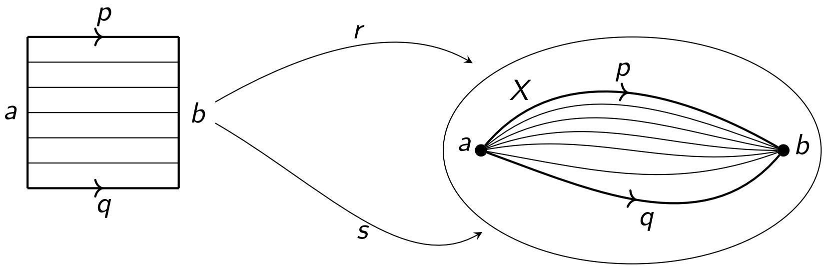
- proves transitivity, symmetry
- equality type can have multiple terms

# 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: Jeff Erickson, 2009

# Origin univalence

Grayson, 2018

figures/voevodsky.jpg

Figure: Vladimir Voevodsky (1966 - 2017)

Why is it called "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 (1966 - 2017)

Why is it called "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)

# Personal remark

The univalence axiom adds:

- intuitive explanation of equaity
- alternative foundations with types
- field of mathematics: HoTT

But does not:

- make all proofs easier or shorter
- eliminate proofs of equivalence

# 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)

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

# 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)

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

# Cubical type theory

Cohen et al., 2015

A constructive extension of HoTT with dimension variables $i, j, k : \mathbb{I}$ (cubes) 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 of *continous* embeddings:

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

*In HoTT, higher-dimensional eqalities behave like these embeddings*

| Level | Types | Cubes | Topology |
|---|---|---|---|
| 1 | $p, q : (a = b)$ | edge | line |
| 2 | $r, s : (p = q)$ | face | path homotopy |
| ... | ... | ... | \dots |
| n | ... | n-hypercube | n-dimensional homotopy |

# Cubes model homotopy

Altenkirch, Brunerie, Licata, et. al 2013

Homotopy groups are defined as equivalence classes of *continous* embeddings:

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

> *In HoTT, higher-dimensional eqalities behave like these embeddings*

| Level | Types | Cubes | Topology |
|---|---|---|---|
| 1 | $p, q : (a = b)$ | edge | line |
| 2 | $r, s : (p = q)$ | face | path homotopy |
| ... | ... | ... | \dots |
| n | ... | n-hypercube | n-dimensional homotopy |

# Operations on cubes

Bezem, Coquand, Huber et al., 2013

figures/kan.jpg

Figure: Daniel Kan (1927 — 2013)

Necessary for modelling HoTT:

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

figures/extension.png

# Presheaf model on $\mathscr{C}$

Dybjer, 1994

Give interpretation for stuff in type theory:

- base category $\mathscr{C}$ contains "extra tools" for model
- every context $\Gamma$ is modelled as presheaf on $\mathscr{C}$, denoted $\widehat{\mathscr{C}}$.
- types and terms also interpreted in $\widehat{\mathscr{C}}$

Goals:

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

Denoted as "presheaf model $\widehat{\mathscr{C}}$".

# Presheaf model on $\mathscr{C}$

Dybjer, 1994

Give interpretation for stuff in type theory:

- base category $\mathscr{C}$ contains "extra tools" for model
- every context $\Gamma$ is modelled as presheaf on $\mathscr{C}$, denoted $\widehat{\mathscr{C}}$.
- types and terms also interpreted in $\widehat{\mathscr{C}}$

Goals:

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

Denoted as "presheaf model $\widehat{\mathscr{C}}$".

# Contexts in presheaf model $\widehat{\mathscr{C}}$

## Definition (Presheaves $\widehat{\mathscr{C}}$)

Contravariant functors $\mathscr{C} \rightarrow \mathbf{Set}$

- generalize sheaves (see sheafication)
- model contexts

`figures/presheaf.png`

Figure: A representation of a preseheaf

# Contexts in presheaf model $\widehat{\{0,1\}}$

Hofstra, 2014

### Example (Reflexive directed graph)

Take $\mathscr{C} = \{0,1\}$ and $\mathrm{Hom}_{\mathscr{C}} = \{B, E, R\}$, $\Gamma \in \widehat{\{0,1\}}$, then
Applying functorial identities:

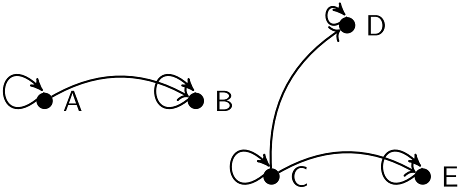



Figure: Reflexive graph

# Types in presheaf model $\widehat{\mathscr{C}}$

**Lemma (Types in a presheaf model)**

*If $\Gamma \in \widehat{\mathscr{C}}$ a context, then the types are*
$\left\{(\Delta,\sigma) \mid \Delta \in \widehat{\mathscr{C}}, \sigma \in Hom_{Ctx}(\Delta,\Gamma)\right\}$.

Helps to characterize types without using presheaves explicitly.

# Types in a presheaf model $\widehat{\{0,1\}}$

## Example (Dependent directed reflexive graph)

Applying previous lemma to the type $A$ in $\widehat{\{0,1\}}$:

figures/type_lemma.png

Figure: Modelled by two contexts and a surjective morphism

# CTT as a presheaf model

Dimension variables and cubes have an abstract representation as "hypercubes" in a base category:

figures/cube_presheaf.png

Figure: Presheaf acting on cubes

- proves consistency of CTT and HoTT
- justifies primitives used in implementations

# Distributive lattice

Necessary to make cubical model CTT

$\mathbb{A}$: countable set of "dimension variables"
CTT built with presheaves on "cube" category $\square$:

- objects: $\{I \mid |I| < \infty, I \subset \mathbb{A}\}$
- morphisms $J \to I$: maps $I \mapsto dM(J)$
  - distributive lattice
  - $x \wedge 0 = 0, x \vee 1 = 1$
  - $\neg 0 = 1$ and $\neg 1 = 0$

figures/lattice.png

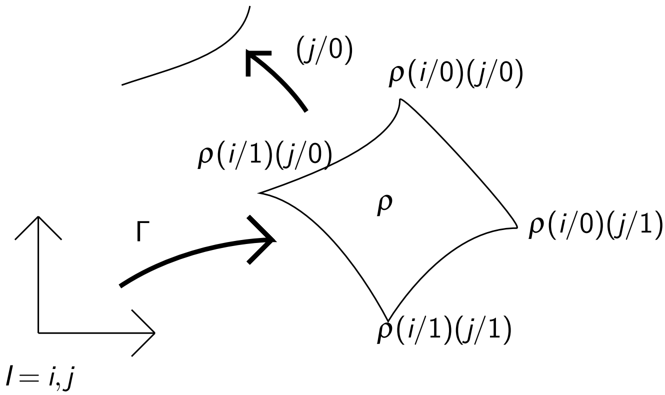
Figure: A simple lattice

# Contexts in presheaf model $\widehat{\square}$

## Example (Cubical contexts ("cubical sets"))

A presheaf $\Gamma \in \widehat{\square}$ is a functor $\square \to \mathbf{Set}$

- $\Gamma \in \widehat{\square}$ applied to $\{i, j\}$ gives square $\rho \in \Gamma(i, j)$
- morphisms in lattice $dM(i, j)$ give corners of $\rho$



# Types in presheaf model $\widehat{\square}$

Type $A$ is presheaf $\widehat{\int_{\mathscr{C}} \Gamma}$, a functor $\int_{\mathscr{C}} \Gamma \to \mathbf{Set}$:

- $\rho \in \Gamma(i)$ is a line
- endpoints $\rho(0), \rho(1)$ lifted to $u(0), u(1) \in A(i, \rho)$

`figures/types.png`

Figure: A type $A$ within context $\Gamma$. Huber (2016)

# Types in general presheaf models

In the presheaf model on $\square$:

- types more complicated
- types no longer simply nested graphs

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

*... transition from interpretation in model to syntax of types*

# 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 $\widehat{\square}$ for constructive model for type theory:

- product, sum types3
- natural numbers

Univalence proven with:

- concepts from simplicial set model (Streicher, Voevodsky, Kapulkin et al. 2006 – 2012)
- partial types and glueing construction

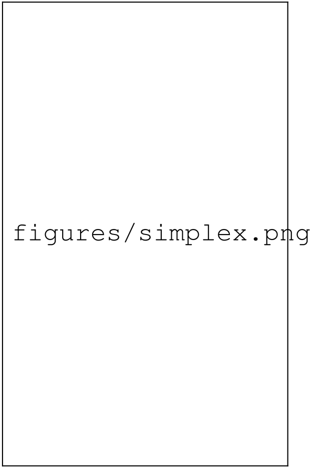



Figure: Related concept of a simplicial complex

# Partial types

Bezem, Coquand, Huber 2013

Partial types only defined on subpolyhedra of cubes.

- $u$ 1-dimensional cube, line and type $A$:
- partial type $A(i/0)$ in type $A$:

`figures/types_side.png`

Figure: Huber, 2015

- red $= A\ [(i=0) \mapsto A(i/0)]$ (syntax)

# Proving univalence

Cohen, Coquand, Huber, Moertberg (2015)

## 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.

1. existence $\mathtt{ua} : (X \simeq Y) \to (X = Y)$ with `Glue` construction.
2. for any partial $f : X \to Y$, $\mathtt{Glue}\ [\varphi \mapsto (X, f)]\ Y \simeq Y$
3. for any type $Y$, $\sum_X X \simeq Y$ contractible
4. existence of an equivalence "eliminator".
5. the trivial map $p : X = Y \to X \simeq Y$ is an inverse "up-to-path" of `ua`

$\Rightarrow$ map `ua` is equivalence $\square$

# Constructing `ua`

## Definition

$\texttt{ua} : \forall\ X\ Y : \mathscr{U}, X \simeq Y \to X = Y$

$$\begin{tikzcd} X \arrow[r, dashed, dash, "\texttt{ua}\ f"] \arrow[d, "f"] & Y \arrow[d, "\texttt{idEquiv}\ Y"] \\ Y \arrow[r, dash] & Y \end{tikzcd}$$

Input:

- partial equivalence $f$, type $X$
- a dimension variable or parameter $i$

Output:

- $\texttt{ua}\ f \equiv \texttt{Glue}\ [(i = 0) \mapsto (X, f), (i = 1) \mapsto (X, \texttt{idEquiv}\ Y)]\ Y$
- $\texttt{ua}\ f$ is path with endpoints $X$ and $Y$.

# 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

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 (set-like) types
- univalence/`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 $s_1 \equiv s_2$.

# Higher homotopies of spheres

Types in HoTT and CTT are topological spaces. Higher homotopy groups compute number of higher-dimensional holes in $S^n$:

figures/groups.png

Figure: HoTT Book, 2013

Homotopy groups can be defined in CTT as datatypes

# Implementation in CTT

## Brunerie, 2016

**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: The case $S^1$ is simply $\mathbb{Z}$ (drawing 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