School on Univalent Mathematics
Univalent foundations

Paige Randall North
(adapted from slides of Benedikt Ahrens)

Outline

1. Interpreting type theory in spaces
2. Contractible types, equivalences, function extensionality
3. Logic in univalent type theory
4. Homotopy levels

Moving from classical foundations to univalent foundations

- Mathematics is the study of structures on sets and their higher analogs.
- Set-theoretic mathematics constitutes a subset of the mathematics that can be expressed in univalent foundations.
- Classical mathematics is a subset of univalent mathematics consisting of the results that require LEM and/or AC among their assumptions.

see Voevodsky, Talk at HLF, Sept 2016

Outline

Interpretation of identities as paths

Inhabitants of $\mathsf{Id}(a, a')$ behave like classical equality

- reflexivity, symmetry, transitivity
- $\mathsf{transport}^B : B(x) \times \mathsf{Id}(x, y) \to B(y)$

Inhabitants of $\mathsf{Id}(a, a')$ behave **un**like classical equality

- There can be two identities $p, q : \mathsf{Id}(x, y)$.
- There can be identities of identities

$$\alpha : \mathsf{Id}_{\mathsf{Id}(x,y)}(p, q), \tag{$*$}$$

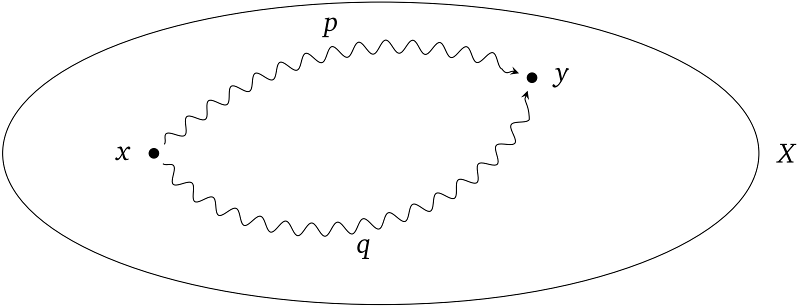
- but there don't always have to be.

We interpret terms of $\mathsf{Id}_X(x, y)$ as **paths from** x **to** y **in** X and sometimes write

$$x \rightsquigarrow_X y.$$

Identities interpreted as paths in a space

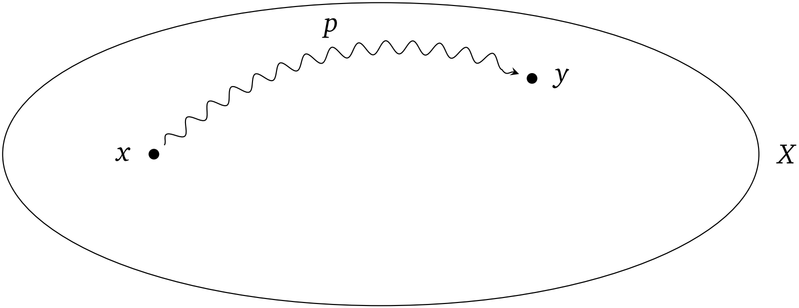
$p, q : x \rightsquigarrow_X y$



Reflexivity (refl) is interpreted as the constant path on a point x.

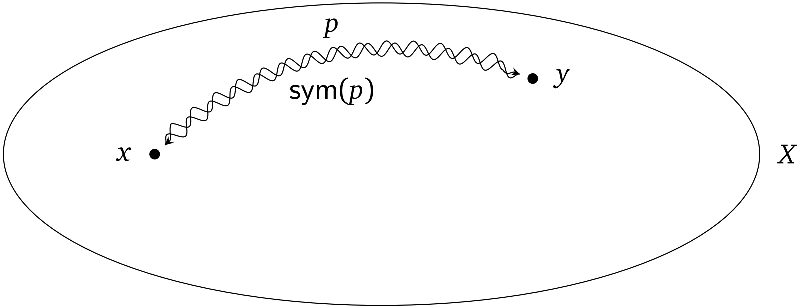
Operations on paths

- $p : x \rightsquigarrow y$



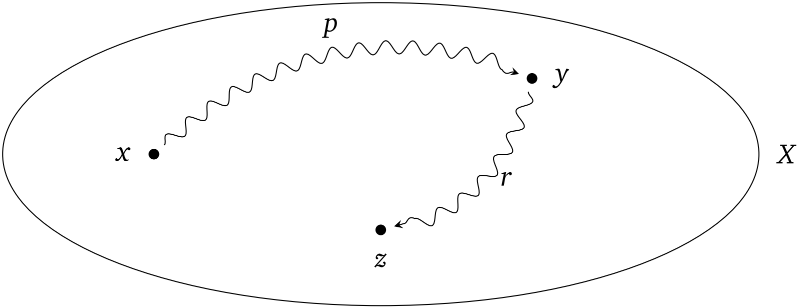
Operations on paths

- $p : x \rightsquigarrow y$
- $\mathsf{sym}(p) : y \rightsquigarrow x$



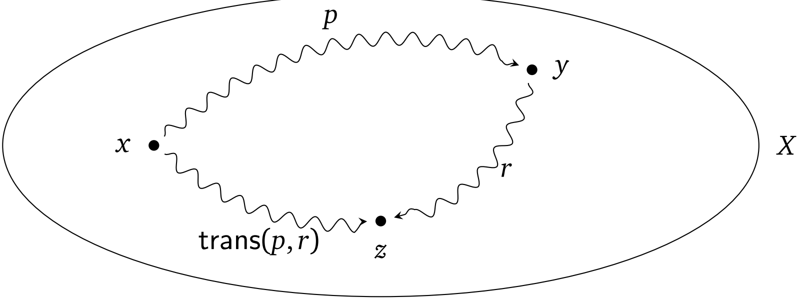
Operations on paths

- $p : x \rightsquigarrow y$
- $\mathsf{sym}(p) : y \rightsquigarrow x$
- $r : y \rightsquigarrow z$



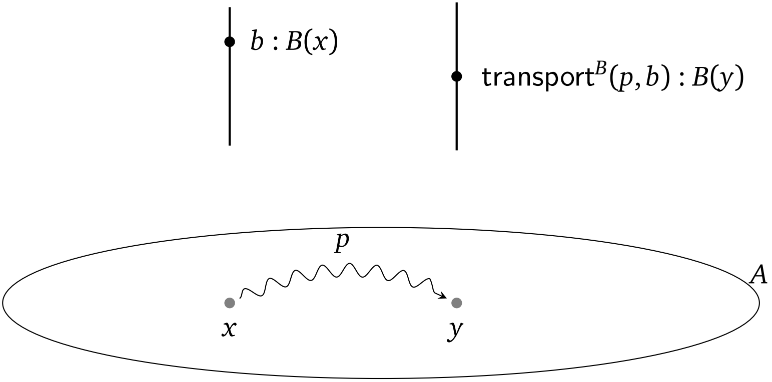
Operations on paths

- $p : x \rightsquigarrow y$
- $\mathsf{sym}(p) : y \rightsquigarrow x$
- $r : y \rightsquigarrow z$
- $\mathsf{trans}(p, r) : x \rightsquigarrow z$



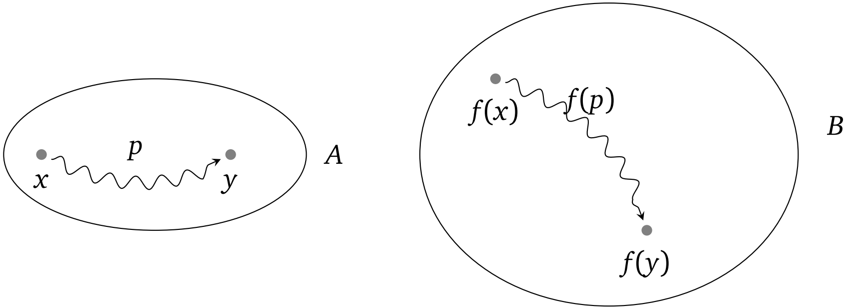
Transport in pictures

$\mathsf{transport}^B : x \rightsquigarrow y \to B(x) \to B(y)$



Functions map paths, not just points

$f : A \to B$



Exercise

Given $f : A \to B$, construct a term of type

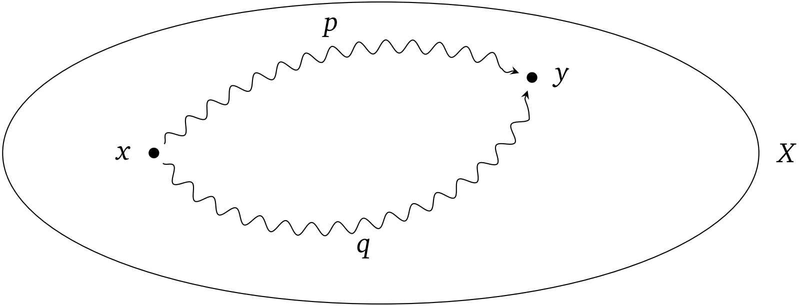
$$\prod_{x,y:A} x \rightsquigarrow_A y \to f(x) \rightsquigarrow_B f(y)$$

Paths between paths

What is a path

$$h : p \rightsquigarrow_{x \rightsquigarrow y} q$$

between paths?



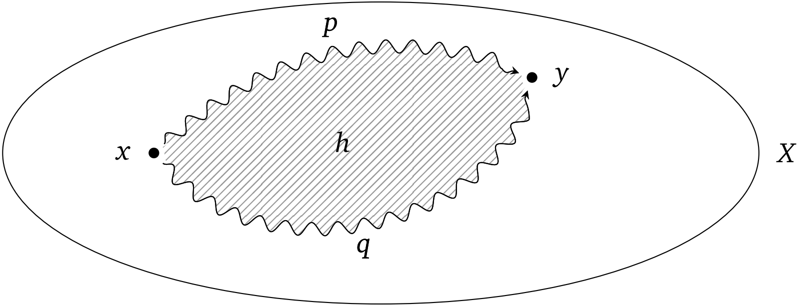
Paths between paths

What is a path

$$h : p \rightsquigarrow_{x \rightsquigarrow y} q$$

between paths?

Intuition: continuous deformation of the first into the second path, called a **homotopy**



Laws satisfied by path concatenation

Can construct homotopies

- $(p \cdot q) \cdot r \rightsquigarrow p \cdot (q \cdot r)$
- $p \cdot 1_y \rightsquigarrow p$
- $1_x \cdot p \rightsquigarrow p$
- $p \cdot p^{-1} \rightsquigarrow 1_x$
- $p^{-1} \cdot p \rightsquigarrow 1_y$
- $\ldots$

Theorem (Garner, van den Berg)

$$(A, \rightsquigarrow_A, \rightsquigarrow_{\rightsquigarrow_A}, \ldots)$$

forms ∞-groupoid, i.e., groupoid laws hold up to “higher” paths

Interpreting types as topological spaces?

We have not mentioned yet what a “space” or ∞-groupoid is.

Types as topological spaces?

It seems difficult (impossible?) to give a formal interpretation of type theory in the category of topological spaces.

Types as Kan complexes

Vladimir Voevodsky has given an interpretation of type theory in the category of Kan complexes.

There is a ‘Quillen equivalence’ between that category and the category of topological spaces, justifying the intuition of ‘types as (topological) spaces’.

Interpreting types as simplicial sets

Syntax	Simpl. set interpretation
$(A, \rightsquigarrow_A, \rightsquigarrow_{\rightsquigarrow_A}, \ldots)$	Kan complex A
$a : A$	$a \in A_0$
$A \times B$	binary product
$A \to B$	space of maps
$A + B$	binary coproduct
$x : A \;\vdash\; B(x)$	fibration $B \to A$ with fibers $B(x)$
$\sum_{x:A} B(x)$	total space of fibration $B \to A$
$\prod_{x:A} B(x)$	space of sections of fibration $B \to A$

Outline

1. Interpreting type theory in spaces
2. Contractible types, equivalences, function extensionality
3. Logic in univalent type theory
4. Homotopy levels

Contractible types

Definition

The type A is **contractible** if we can construct a term of type

$$\mathsf{isContr}(A) \quad :\equiv \quad \sum_{x:A} \prod_{y:A} y \rightsquigarrow x$$

A contractible type. . .

- is also called **singleton** type.
- has a point and a path from any point to that point.

By path inversion and concatenation, there is a path between any two points of a contractible type.

Equivalences

Definition

A map $f : A \to B$ is an **equivalence** if it has contractible fibers, i.e.,

$$\mathsf{isequiv}(f) \;\; :\equiv \;\; \prod_{b:B} \mathsf{isContr}\left(\sum_{a:A} f(a) \rightsquigarrow b \right)$$

The type of equivalences:

$$A \simeq B \;\; :\equiv \;\; \sum_{f:A\to B} \mathsf{isequiv}(f)$$

Exercise: Given an equivalence $f : A \simeq B$, define a function $g : B \to A$. Construct paths $f(g(y)) \rightsquigarrow y$ and $g(f(x)) \rightsquigarrow x$.

Exercises

- Show that $\mathbb{1}$ is contractible.
- Let A be a contractible type. Construct an equivalence $A \simeq \mathbb{1}$.
- Given types A and B, let $f : A \to B$ and $g : B \to A$. Suppose having families of paths $\eta_x : g(f(x)) \rightsquigarrow x$ and $\epsilon_y : f(g(y)) \rightsquigarrow y$. Show that f is an equivalence.

Path types of pairs

Exercise: construct equivalences

- for $(a, b) : A \times B$,

$$\Big((a, b) \rightsquigarrow (a', b')\Big) \;\simeq\; \Big((a \rightsquigarrow a') \times (b \rightsquigarrow b')\Big)$$

- for $(a, b) : \sum_{a:A} B(a)$,

$$\Big((a, b) \rightsquigarrow (a', b')\Big) \;\simeq\; \sum_{p:a \rightsquigarrow a'} \mathsf{transport}^B(p, b) \rightsquigarrow b'$$

Path types of function spaces

For $f, g : A \to B$ cannot show

$$\left(f \rightsquigarrow g\right) \;\simeq\; \left(\prod_{a:A} f(a) \rightsquigarrow g(a)\right)$$

Exercise: Define

$$\mathsf{toPointwisePath} : \prod_{f,g:A\to B} \left(f \rightsquigarrow g\right) \to \left(\prod_{a:A} f(a) \rightsquigarrow g(a)\right)$$

Axiom (function extensionality)

$$\mathsf{toPointwisePath}(f,g) : \left(f \rightsquigarrow g\right) \;\to\; \left(\prod_{a:A} f(a) \rightsquigarrow g(a)\right)$$

is an equivalence for any f, g.

Exercise: define $\mathsf{toPointwisePath}$ for Π-types.

Path types of identity types

We cannot show the following:

Axiom (uniqueness of identity proofs)

$$\prod_{a,b:A} \prod_{p,q:a \rightsquigarrow b} p \rightsquigarrow q.$$

Path types of the universe

Exercise: Define

$$\mathsf{idtoequiv} : \prod_{A,B:\mathsf{Type}} (A \rightsquigarrow B) \to (A \simeq B)$$

We cannot show the following:

Axiom (univalence)

$$\mathsf{idtoequiv}(A,B) : (A \rightsquigarrow B) \to (A \simeq B)$$

is an equivalence.

Characterization of path types

- Σ-types: provable characterization
- Π-types: axiom of function extensionality
- Id-types: axiom of uniqueness of identity proofs
- Type: axiom of univalence

- FE is consistent with both UIP and U. (Actually U $\to$ FE.)
- UIP and U are inconsistent.
- Type theory + UIP + FE has a logical interpretation and a set interpretation.
- Type theory + U has a space interpretation.

We choose type theory + U (univalent foundations), and recover logic and set theory from certain types that we call *propositions* and *sets*.

Outline

Some types are propositions

Curry-Howard

- Types are propositions.
- Terms are proofs.

Univalent logic

- **Some** types are propositions.
- Terms **of those types** are proofs.

Definition (Propositions in univalent type theory)

Type A is a **proposition** if

$$\mathsf{isProp}(A) \;:\equiv\; \prod_{x,y:A} x \rightsquigarrow y$$

is inhabited.

Examples of propositions

Exercise: show that

- 1 is a proposition.
- any contractible type is a proposition.
- 0 is a proposition.
- if A and B are propositions, then $A \times B$ is a proposition.
- if B is a proposition, then $A \to B$ is a proposition.

Connectives in univalent logic

Definition

$$\mathsf{Prop} \;:\equiv\; \sum_{X:\mathsf{Type}} \mathsf{isProp}(X)$$

We want logical connectives

$$\begin{aligned} \top, \bot &: \mathsf{Prop} \\ \vee, \wedge, \Rightarrow &: \mathsf{Prop} \to \mathsf{Prop} \to \mathsf{Prop} \\ \neg &: \mathsf{Prop} \to \mathsf{Prop} \\ \forall_X, \exists_X &: (X \to \mathsf{Prop}) \to \mathsf{Prop} \qquad \text{(binding a variable)} \end{aligned}$$

Univalent logic

- 1 and 0 are propositions. Hence

$$\top \;:\equiv\; 1 \qquad \bot :\equiv 0$$

- If A and B are propositions, so is $A \times B$. Hence

$$A \wedge B \;:\equiv\; A \times B$$

- If B is a proposition, so is $A \to B$. Hence

$$A \Rightarrow B \;:\equiv\; A \to B$$

- 0 is a proposition, hence $A \to 0$ is. Hence

$$\neg A :\equiv\; A \to 0$$

- If $B(a)$ (for any a) are propositions, so is $\prod_{a:A} B(a)$. Hence

$$\forall (a : A), B(a) \;:\equiv\; \prod_{a:A} B(a)$$

$\vee$ and $\exists$ in univalent logic

- Exercise: Find a type T that is a proposition such that $T+T$ is not a proposition.
  Conclusion: can **not** set

$$A \vee B \;\; :\equiv \;\; A + B$$

- $\Sigma_{n:\mathsf{Nat}}\mathsf{isEven}(n)$ is the type of all even natural numbers. It is not a proposition.
  Conclusion: can **not** set

$$\exists(a : A), B(a) \;\; :\equiv \;\; \Sigma_{a:A} B(a)$$

Solution: introduce a type former that makes propositions.

Propositional truncation

Formation If A is a type, then $||A||$ is a type

Introduction If $a : A$, then $\overline{a} : ||A||$

$$p(A) : \prod_{x,y:||A||} x \rightsquigarrow y$$

Elimination If $f : A \to B$ and B is a proposition, then $\bar{f} : ||A|| \to B$

Computation $\bar{f}(\overline{a}) \equiv f(a)$

- $p(A)$ turns $||A||$ into a proposition.
- Intuitively, $||A||$ is empty if A is, and contractible if A has at least one element.

$\vee$ and $\exists$ in univalent logic

- $$A \vee B \;\; :\equiv \;\; ||A + B||$$
- $$\exists (a : A), B(a) \;\; :\equiv \;\; ||\Sigma_{a:A} B(a)||$$

For example:

$$\mathsf{isSurjective}(f) :\equiv \prod_{b:B} ||\Sigma_{a:A} f(a) \rightsquigarrow b||$$

Propositional extensionality

We would like to consider two propositions to be equal if they are logically equivalent:

$$\prod_{P,Q:\mathsf{Prop}} (P \rightsquigarrow Q) \simeq (P \longleftrightarrow Q)$$

Propositional extensionality

We would like to consider two propositions to be equal if they are logically equivalent:

$$\prod_{P,Q:\mathsf{Prop}} (P \rightsquigarrow Q) \simeq (P \longleftrightarrow Q)$$

Axiom: propositional extensionality

Exercise: state the axiom of propositional extensionality, e.g., analogously to function extensionality.

Exercise

Given $f : A \to B$, show that $\mathsf{isequiv}(f)$ is a proposition.

Exercise

Show that propositional extensionality follows from univalence.

Outline

1. Interpreting type theory in spaces
2. Contractible types, equivalences, function extensionality
3. Logic in univalent type theory
4. Homotopy levels

Contractible types, propositions and sets

- A is **contractible** if we can construct a term of type

$$\mathsf{isContr}(A) \;:\equiv\; \sum_{x:A} \prod_{y:A} y \rightsquigarrow x$$

- A is a **proposition** if $\prod_{x,y:A} x \rightsquigarrow y$ is inhabited

$$\mathsf{isProp}(A) \;:\equiv\; \prod_{x,y:A} x \rightsquigarrow y$$

- A is a **set** if, for any $x, y : A$, the type $x \rightsquigarrow y$ is a proposition

$$\mathsf{isSet}(A) \;:\equiv\; \prod_{x,y:A} \mathsf{isProp}(x \rightsquigarrow y)$$

Contractible types, propositions and sets

- A is **contractible** if we can construct a term of type

$$\mathsf{isContr}(A) \;:\equiv\; \sum_{x:A} \prod\nolimits_{y:A} y \leadsto x$$

- A is a **proposition** if $\prod_{x,y:A} \mathsf{isContr}(x \leadsto y)$ is inhabited

$$\mathsf{isProp}(A) \;:\equiv\; \prod_{x,y:A} \mathsf{isContr}(x \leadsto y)$$

- A is a **set** if, for any $x, y : A$, the type $x \leadsto y$ is a proposition

$$\mathsf{isSet}(A) \;:\equiv\; \prod_{x,y:A} \mathsf{isProp}(x \leadsto y)$$

Exercises

- For a type A, show that $\prod_{x,y:A} \mathsf{isContr}(x \rightsquigarrow y) \longleftrightarrow \prod_{x,y:A} x \rightsquigarrow y$.
- Show that Bool is a set. Is it contractible? Is it a proposition?
- Show that Nat is a set. Is it contractible? Is it a proposition?

Homotopy level of a type

Definition

$$\mathsf{isofhlevel} : \mathsf{Nat} \to \mathsf{Type} \to \mathsf{Type}$$

$$\mathsf{isofhlevel}(0)(X) :\equiv \mathsf{isContr}(X)$$

$$\mathsf{isofhlevel}(S(n))(X) :\equiv \prod_{x,y:X} \mathsf{isofhlevel}(n)(x \rightsquigarrow y)$$

Homotopy level of a type

Definition

$$\mathsf{isofhlevel} : \mathsf{Nat} \to \mathsf{Type} \to \mathsf{Prop}$$
$$\mathsf{isofhlevel}(0)(X) :\equiv \mathsf{isContr}(X)$$
$$\mathsf{isofhlevel}(S(n))(X) :\equiv \prod_{x,y:X} \mathsf{isofhlevel}(n)(x \rightsquigarrow y)$$

Exercise: Show that $\mathsf{isofhlevel}(n)(X)$ is a proposition.

Preservation of levels

. . . by type constructors

- If A and B are of level n, then so is $A \times B$.
- If B is of level n, then so is $A \to B$.
- If A and $B(a)$ (for any $a : A$) are of level n, then so is $\sum_{a:A} B(a)$.
- If $B(a)$ (for any $a : A$) are of level n, then so is $\prod_{a:A} B(a)$.

. . . under equivalence of types

If A is of level n and $A \simeq B$ then B is of level n.

Cumulativity

If type A is of h-level n, then it is also of h-level $S(n)$.

Set extensionality

We would like to consider two sets to be equal if they are in bijection:

$$\prod_{S,T:\mathsf{Set}} (S \rightsquigarrow T) \simeq (S \cong T)$$

Set extensionality

We would like to consider two sets to be equal if they are in bijection:

$$\prod_{S,T:\mathsf{Set}} (S \rightsquigarrow T) \simeq (S \cong T)$$

Axiom: set extensionality

Exercise: state the axiom of set extensionality, e.g., analogously to propositional extensionality.

Exercise

Show that set extensionality follows from univalence.

Summary: Univalent Foundations

- Univalent type theory with an interpretation in spaces (precisely: Kan complexes)

Type theory	Interpretation
A type	space A
$a : A$ (term a of type A)	point a in space A
$f : A \to B$	map from A to B
$p : a \rightsquigarrow b$	path (1-morphism) from a to b in A
$\alpha : p \rightsquigarrow_{a \rightsquigarrow b} q$	homotopy from p to q in A

- “World” of **logic** (propositions and proofs) given by Prop
- “World” of **sets** given by Set