Set-level mathematics

Benedikt Ahrens

School on Univalent Mathematics, Cortona, 2022

Outline

1. Reminder: homotopy levels
2. How to show that something is (not) a set?
3. Subtypes, relations, set-level quotient
4. Algebraic structures

Outline

1. Reminder: homotopy levels
2. How to show that something is (not) a set?
3. Subtypes, relations, set-level quotient
4. Algebraic structures

Definition of homotopy levels

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

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

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

Definition

$$\mathsf{Set} \quad :\equiv \quad \sum_{X:\mathsf{Type}} \mathsf{isofhlevel}(2)(X)$$

- Any two parallel paths in a set are homotopic.
- Any closed path (loop) on x is homotopic to the constant path $\mathsf{refl}(x)$.

Outline

1. Reminder: homotopy levels
2. How to show that something is (not) a set?
3. Subtypes, relations, set-level quotient
4. Algebraic structures

Decidable equality

Definition

A type X is **decidable** if there is a term of type

$$X + \neg X$$

Definition

A type X has **decidable path-equality** if we can write a term of type

$$\prod_{x,x':A} (x \rightsquigarrow x') + \neg(x \rightsquigarrow x')$$

(that is, if all its paths types are decidable)

Hedberg's theorem

Theorem (Hedberg)

If a type X has decidable equality, then it is a set.

In the problem session, we will show that Bool and Nat are sets.

Closure properties

- $\sum_{x:A} B(x)$ is a set if A and all $B(x)$ are
- $A \times B$ is a set if A and B are
- $\prod_{x:A} B(x)$ is a set if all $B(x)$ are
- $A \to B$ is a set if B is
- A is a set if it is a proposition

Exercise

Do you know

- a type that is a set?
- a type for which you don't know (yet) whether it is a set?
- a type for which you know it is not a set?

Another set

Theorem

The type

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

is a set.

The proof relies on the univalence axiom for the universe Type.

Exercise

How would you generalize the above statement to any h-level?
How would you attempt proving it?

Remark

Prop does not have decidable equality.

Are all types sets?

Is there a type that is not a set?

It depends:

- In Martin-Löf type theory some types can not be shown to be sets.
- In univalent type theory some types can be shown not to be sets.

Types that are **not** sets

Suppose that Type is a univalent universe containing the type Bool.

Exercise

Show that Type not a set.

Which property of Bool does the proof of the above result exploit?

Exercise

Show that

$$\mathsf{Set} \quad :\equiv \quad \sum_{X:\mathsf{Type}} \mathsf{isofhlevel}(2)(X)$$

is not a set. Does it have an h-level?

Sets and propositions

- It is often useful for types representing “properties” to be propositions (as we’ll see later).
- Properties involving equality are usually propositions when the types involved are *sets*, but in general care is needed.

Example

Given $f : X \to Y$,

$$\mathsf{isInjective}(f) \;:\equiv\; \prod_{x,x':X} f(x) \rightsquigarrow f(x') \to x \rightsquigarrow x'$$

is not a proposition in general, but it is if X is a set.

Exercise

Define $\mathsf{isInjective}(f)$ in a such a way that it is a proposition for X and Y of any level.

Isomorphism vs. equivalence

Given $f : A \to B$,

$$\mathsf{isiso}(f) \quad :\equiv \quad \sum_{g:B\to A} (g \circ f \rightsquigarrow 1_A) \times (f \circ g \rightsquigarrow 1_B)$$

is **not** a proposition in general, but it is if A and B are sets.

Warning

Stating the univalence axiom with isomorphisms instead of equivalences yields an inconsistency.

When A and B are sets, then $\mathsf{isiso}(f) \simeq \mathsf{isequiv}(f)$.

Outline

Predicates on types

A **subtype** A on a type X is a map

$$A : X \to \mathsf{Prop}$$

Exercise

Show that the type of subtypes of X is a set.

The **carrier** of a subtype A is the type of elements satisfying A:

$$\mathsf{carrier}(A) := \sum_{x:X} A(x)$$

Relations on a type

A **binary relation** R on a type X is a map

$$R : X \to X \to \mathsf{Prop}$$

Exercise

Show that the type of binary relations on X is a set.

Properties of such relations are defined as usual, e.g.,

$$\mathsf{reflexive}(R) \quad :\equiv \quad \prod_{x:X} R(x)(x)$$

Exercise

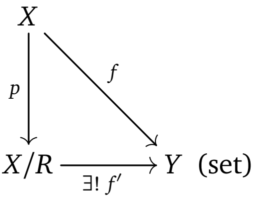
Formulate the properties of being symmetric, transitive, an equivalence relation.

Set-level quotient

Given type X and an equivalence relation R on X, the **quotient**

$$X \xrightarrow{p} X/R$$

is defined as the unique pair $(X/R, p)$ such that any compatible map f **into a set** Y factors via p:



I.e., the map given by precomposition with p is an equivalence

$$\sum_{f:X\to Y} \mathsf{iscompatible}(f) \quad \simeq \quad X/R \to Y$$

The quotient set

To define the quotient X/R of a set by an equivalence relation, we proceed as usual in set theory:

- First we define for a subtype $A : X \to \mathsf{Prop}$

$$\begin{aligned} \mathsf{iseqclass}(A,R) \quad &:\equiv \quad \|\mathsf{carrier}(A)\| \\ &\times \prod_{x,y:A} Rxy \to Ax \to Ay \\ &\times \prod_{x,y:A} Ax \to Ay \to Rxy \end{aligned}$$

- Then we define

$$X/R \quad :\equiv \quad \sum_{A:X\to\mathsf{Prop}} \mathsf{iseqclass}(A,R)$$

Exercise

Show that X/R is a set. Show that it has the desired universal property of a quotient.

Outline

Reminder: paths between pairs

Given $B : A \to \mathsf{Type}$ and $a, a' : A$ and $b : B(a)$ and $b' : B(a')$,

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

If $B(x)$ is a proposition for any $x : A$, then this can be simplified to

$$(a, b) \rightsquigarrow (a', b') \quad \simeq \quad a \rightsquigarrow a'$$

Exercise

Why?

Monoids

Traditionally (in set theory), a monoid is a triple (M, μ, e) of

- a set M
- a multiplication $\mu : M \times M \to M$
- a unit $e \in M$

subject to the usual axioms: associativity, and left and right neutrality.

Monoids in type theory

In type theory, a monoid is a tuple $(M, \mu, e, \alpha, \lambda, \rho)$ where

1. $M : \mathsf{Set}$
2. $\mu : M \times M \to M$ (multiplication)
3. $e : M$ (neutral element)
4. $\alpha : \Pi_{(a,b,c:M)} \mu(\mu(a,b),c) \rightsquigarrow \mu(a,\mu(b,c))$ (associativity)
5. $\lambda : \Pi_{(a:M)} \mu(e,a) \rightsquigarrow a$ (left neutrality)
6. $\rho : \Pi_{(a:M)} \mu(a,e) \rightsquigarrow a$ (right neutrality)

Monoids in type theory

In type theory, a monoid is a tuple $(M, \mu, e, \alpha, \lambda, \rho)$ where

1. $M : \mathsf{Set}$
2. $\mu : M \times M \to M$ (multiplication)
3. $e : M$ (neutral element)
4. $\alpha : \Pi_{(a,b,c:M)} \mu(\mu(a,b),c) \rightsquigarrow \mu(a,\mu(b,c))$ (associativity)
5. $\lambda : \Pi_{(a:M)} \mu(e,a) \rightsquigarrow a$ (left neutrality)
6. $\rho : \Pi_{(a:M)} \mu(a,e) \rightsquigarrow a$ (right neutrality)

Why $M : \mathsf{Set}$?

Monoids in type theory

In type theory, a monoid is a tuple $(M, \mu, e, \alpha, \lambda, \rho)$ where

1. $M : \mathsf{Set}$
2. $\mu : M \times M \to M$ (multiplication)
3. $e : M$ (neutral element)
4. $\alpha : \Pi_{(a,b,c:M)} \mu(\mu(a,b),c) \rightsquigarrow \mu(a, \mu(b,c))$ (associativity)
5. $\lambda : \Pi_{(a:M)} \mu(e,a) \rightsquigarrow a$ (left neutrality)
6. $\rho : \Pi_{(a:M)} \mu(a,e) \rightsquigarrow a$ (right neutrality)

Why $M : \mathsf{Set}$?

Abstractly, a monoid is a (dependent) pair $(\mathit{data}, \mathit{proof})$ where

- *data* is a triple (M, μ, e) as above
- *proof* is a triple (α, λ, ρ) saying that (data) satisfy the usual axioms.

The type of monoids

- We want to regard two monoids $(data, proof)$ and $(data', proof')$ as being the same if $data$ is the same as $data'$.
- This is ensured if the type encoding the monoid axioms is a **proposition**.
- This is in turn guaranteed as long as the underlying type M is required to be a **set**.

$$\mathsf{Monoid} \;:\equiv\; \sum_{(M:\mathsf{Set})}\ \sum_{(\mu,e):\mathsf{MonoidStr}(M)} \mathsf{MonoidAxioms}(M,(\mu,e))$$

with

$$\mathsf{isProp}(\mathsf{MonoidAxioms}(M,(\mu,e)))$$

Monoid isomorphisms

Given monoids $\mathbf{M} \equiv (M, \mu, e, \alpha, \lambda, \rho)$ and $\mathbf{M}' \equiv (M', \mu', e', \alpha', \lambda', \rho')$, a **monoid isomorphism** is a bijection $f : M \cong M'$ preserving multiplication and neutral element.

$$
\begin{aligned}
\mathbf{M} \rightsquigarrow \mathbf{M}' &\simeq (M, \mu, e) \rightsquigarrow (M', \mu', e') \\
&\simeq \sum_{p : M \rightsquigarrow M'} (\mathsf{transport}^{Y \mapsto (Y \times Y \to Y)}(p, \mu) \rightsquigarrow \mu') \\
&\qquad\qquad \times (\mathsf{transport}^{Y \mapsto Y}(p, e) \rightsquigarrow e') \\
&\simeq \sum_{f : M \cong M'} \left(f \circ m \circ (f^{-1} \times f^{-1}) \rightsquigarrow m'\right) \\
&\qquad\qquad \times (f \circ e \rightsquigarrow e') \\
&\simeq \mathbf{M} \cong \mathbf{M}'
\end{aligned}
$$

Paths are isomorphisms for groups

$$
\begin{aligned}
\mathbf{G} \rightsquigarrow \mathbf{G}' &\simeq (G,S) \rightsquigarrow (G',S') \\
&\simeq \sum_{p:G\rightsquigarrow G'} \mathsf{transport}^{\mathsf{GrpStructure}}(p,S) \rightsquigarrow S' \\
&\simeq \sum_{p:G\rightsquigarrow G'} (\mathsf{transport}^{Y\mapsto(Y\times Y\to Y)}(p,m) \rightsquigarrow m') \\
&\qquad \times (\mathsf{transport}^{Y\mapsto(Y\to Y)}(p,i) \rightsquigarrow i') \\
&\qquad \times (\mathsf{transport}^{Y\mapsto Y}(p,e) \rightsquigarrow e') \\
&\simeq \sum_{f:G\simeq G'} \left(f\circ m\circ(f^{-1}\times f^{-1}) \rightsquigarrow m'\right) \\
&\qquad \times \left(f\circ i\circ f^{-1} \rightsquigarrow i'\right) \\
&\qquad \times (f(e) \rightsquigarrow e') \\
&\simeq (\mathbf{G}\cong\mathbf{G}')
\end{aligned}
$$

Transport along group isomorphism

We now have two ingredients:

1. $(\mathbf{G} \rightsquigarrow \mathbf{G}') \quad \simeq \quad (\mathbf{G} \cong \mathbf{G}')$
2. $\mathsf{transport}^T : (\mathbf{G} \rightsquigarrow \mathbf{G}') \to T(\mathbf{G}) \to T(\mathbf{G}')$ for any structure T on the type of groups

Composing these, we get

$$\mathsf{transport}^T : (\mathbf{G} \cong \mathbf{G}') \to T(\mathbf{G}) \to T(\mathbf{G}')$$

In other words, any property or structure on groups that can be expressed in univalent type theory can be transported along isomorphism of groups.

Structure Identity Principle

The *Structure Identity Principle* (Coquand, Aczel) says

> *Isomorphic mathematical structures are structurally identical; i.e. have the same structural properties.*

The Structure Identity Principle holds in Univalent Foundations for many algebraic structures: isomorphic such structures have **all** the same (definable) properties.