Set-level mathematics

Joj Ahrens

Outline

Outline

Definition of H-Levels

$$\mathsf{isofhlevel}(n, X) : \mathsf{Nat} \to \mathsf{U} \to \mathsf{hProp}$$

$$\mathsf{isofhlevel}(0, X) := \mathsf{iscontr}(X)$$

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

Definition

A **set** is a type of hlevel 2.

Under the intended semantics, this means that for any two parallel paths in a *set*, the space of homotopies between them is contractible. This condition is equivalent to being homotopy equivalent to a discrete space.

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 property

Exercise

Do you know

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

Outline

Decidable equality

Definition

A type X is **decidable** if we can write a term of type

$$X + \neg X$$

Definition

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

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

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

Hedberg's theorem

Theorem

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.

Are all types sets?

Is there a type that is not a set?

Great question! It depends:

- In “spartan” type theory some types cannot be shown to be sets.
- In univalent type theory some types can shown not to be sets.

From now on, we consider **univalent type theory**.

Another set

Theorem

The type

$$\mathsf{hProp}_{\mathsf{U}} := \sum_{X:\mathsf{U}} \mathsf{isaprop}(X)$$

is a set.

The proof relies on the univalence axiom for the unviverse U.

Exercise

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

Types that are **not** sets

Exercise

Let U be a univalent universe that contains the type Bool. Why is U not a set?

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

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: given $f : X \to Y$,

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

is not a proposition in general, but it is if X and Y are sets.

Outline

Set-level quotient

Given type X and 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:

$$\begin{tikzcd} X \arrow[d, "p"'] \arrow[dr, "f"] & \\ X/R \arrow[r, "\exists!\, f'"'] & Y \end{tikzcd}$$

that is,

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

(more precisely, the map given by precomposition with p should be an equivalence).

Predicates on types

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

$$A : X \to \mathsf{hProp}_U$$

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

$$R : X \to X \to \mathsf{hProp}_U$$

Exercise

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

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

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

Exercise

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

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 $A : X \to \mathsf{hProp}_U$

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

Then we define

$$X/R := \sum_{A:X \to \mathsf{hProp}_U} \mathsf{iseqclass}(A)$$

Outline

Reminder: paths between pairs

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

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

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

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

Exercise

Why?

Groups

Traditionally (in set theory), a group is a quadruple (G, m, e, i) of

- a set G
- a multiplication $m : G \times G \to G$
- a unit $e \in G$
- an inverse $i : G \to G$

subject to the usual axioms.

Groups in type theory

In type theory, a group is a (dependent) pair $(\mathit{data}, \mathit{proof})$ where

- data is a quadruple (G, m, e, i) as above
- p is a proof that these satisfy the usual axioms.

Groups in type theory

In type theory, a group is a (dependent) pair $(data, proof)$ where

- $data$ is a quadruple (G, m, e, i) as above
- p is a proof that these satisfy the usual axioms.

We want to regard two groups $(data, proof)$ and $(data', proof')$ as being the same if $data$ is the same as $data'$.

Groups in type theory

In type theory, a group is a (dependent) pair $(\mathit{data}, \mathit{proof})$ where

- data is a quadruple (G, m, e, i) as above
- p is a proof that these satisfy the usual axioms.

We want to regard two groups $(\mathit{data}, \mathit{proof})$ and $(\mathit{data}', \mathit{proof}')$ as being the same if data is the same as data'.
This requires that the type encoding the group axioms be a *proposition*.

Groups in type theory

In type theory, a group is a (dependent) pair $(data, proof)$ where

- $data$ is a quadruple (G, m, e, i) as above
- p is a proof that these satisfy the usual axioms.

We want to regard two groups $(data, proof)$ and $(data', proof')$ as being the same if $data$ is the same as $data'$.

This requires that the type encoding the group axioms be a *proposition*.

This is in turn guaranteed as long as the underlying type G is required to be a *set*.

Exercise

Why?

Group isomorphisms

The type of groups is

$$\mathsf{Grp} := \sum_{X:\mathsf{hSet}} \mathsf{GrpStructure}(X)$$

A group isomorphism $G \to G'$ is

- a bijective function on the underlying sets $X \to X'$
- compatible with the group structures S and S' on X and X'.

Identity is isomorphism for groups

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