Logic, homotopy levels, and equivalences

Michael Shulman

February 21, 2012

Intensional type theory

From now on, we work in a type theory with

1. Dependent products
2. Inductive type families (including identity types)
3. At least one "universe" Type

Basically: the fragment of Coq's type theory that ignores coinductive types and the sort Prop.

Function extensionality

We also assume:

Axiom (Function extensionality)

$$f, g \colon \prod_{x:A} B(x) \vdash \left(\prod_{x:A} (f(x) = g(x)) \right) \longrightarrow (f = g)$$

- Not provable in plain type theory.
- True in set theory: “If two functions are pointwise equal, then they are equal as functions.”
- True in homotopy theory: “If two functions are homotopic, they are connected by a path in the space of functions.”

The eta rule

Define $\eta(f) :\equiv (\lambda x.f(x))$. Then for any a,

$$\eta(f)(a) = (\lambda x.f(x))(a) \to_\beta f(a).$$

Hence, function extensionality implies

$$f = \eta(f)$$

This is a proof of a proposition, i.e. a term in the type $(f = \eta(f))$. It would be stronger to assert a computation rule $f \to_\eta \eta(f)$ or $\eta(f) \to_\eta f$; the upcoming Coq v8.4 will do the latter.

Paths

We think of terms $p\colon (x = y)$ as paths $x \rightsquigarrow y$.

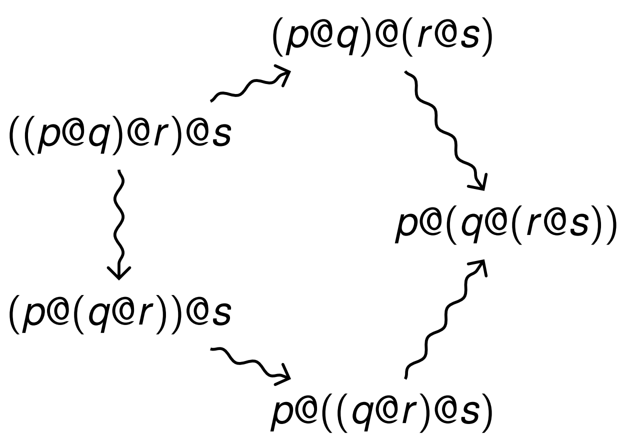
- Reflexivity becomes the constant path $\mathrm{refl}_x\colon x \rightsquigarrow x$.
- Transitivity becomes the concatenation $p@q\colon x \rightsquigarrow z$ of paths $x \overset{p}{\rightsquigarrow} y \overset{q}{\rightsquigarrow} z$.
- Symmetry becomes reversal of a path $!p\colon y \rightsquigarrow x$.

But now there is more to say.

- Concatenation is associative: $(p@q)@r = p@(q@r)$.
- Better: there is a path $\alpha_{p,q,r}\colon (p@q)@r \rightsquigarrow p@(q@r)$.

2-paths

The "associator" $\alpha_{p,q,r}$ is coherent:



...or more precisely, there is a path between those two concatenations...
...which then has to be coherent...

∞-groupoids

Definition (Grothendieck,Batanin,...)

An ∞-groupoid is a collection of points, together with paths between points, 2-paths between paths, and so on, with all possible coherent and consistent concatenation operations.

Theorem (Lusmdaine,Garner–van den Berg)

The tower of identity types of any type A in intensional type theory forms an ∞-groupoid.

- Basically this means that any reasonable fact about paths and higher paths is true.
- This is a theorem *in type theory*. (But not completely formalized internally, due to issues with infinity.)
- Separately (later), ∞-groupoids defined *in set theory* are a model of type theory.

Some homotopy types

- The circle S^1 has one point $b\colon S^1$, and one path $(b = b)$ for every integer.
- The sphere S^2 has one point $b\colon S^2$, the constant path $\mathsf{refl}_b\colon (b = b)$, and one 2-path $(\mathsf{refl}_b = \mathsf{refl}_b)$ for every integer.
- In the type Type, a path $p\colon (A = B)$ is an *equivalence* (or bijection). E.g. there are many terms in $(\mathbb{N} = \mathbb{Z})$.

BUT: We cannot prove any of this in our current type theory. Later, we'll extend the theory; for now, these are *intended* examples.

Some non-homotopy types

"Definition"

An h-set (or just a set) is a type that contains no nontrivial k-paths for any $k \geq 1$.

Examples

- $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$
- unit and $\emptyset$
- $A + B$, $A \times B$, $A \to B$, $\sum_{x:A} B$, and $\prod_{x:A} B$, as long as A and B are h-sets.
- list(A), if A is an h-set.
- All datatypes arising in everyday programming.
- Any type equivalent to an h-set is an h-set.

Paths for type constructors

- A path in $A \times B$ is a path in A and a path in B.
- A path in $A + B$ is a path in A *or* a path in B.
- Any two paths in unit are the same.
- A path in $\prod_{x:A} B$ is a "pointwise path" (using function extensionality).
- A path in $\sum_{x:A} B$ is. . . what?

Transporting along paths

Given $B\colon A \to \text{Type}$, $x, y\colon A$, and $p\colon (x = y)$, we have the operation of transporting along p:

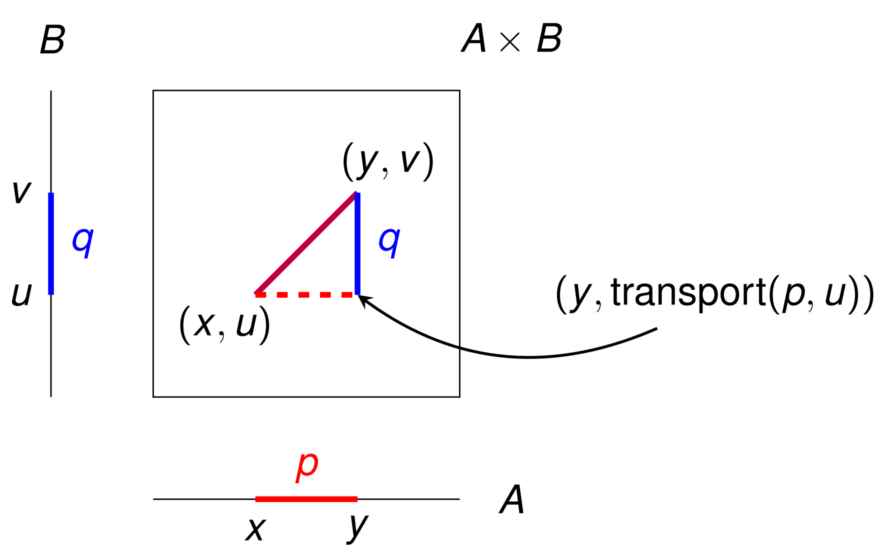
$$\text{transport}(p, -)\colon B(x) \to B(y).$$

A path $(x, u) = (y, v)$ in $\sum_{x\colon A} B(x)$ consists of

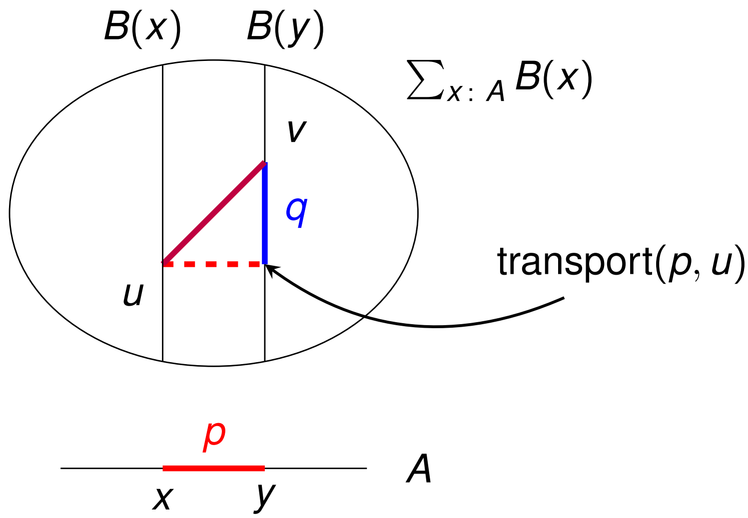
- A path $p\colon (x = y)$ in A, and
- A path $q\colon (\text{transport}(p, u) = v)$ in $B(y)$.

Note: If B is independent of x, then $\text{transport}(p, u) = u$.

Paths in cartesian products



Paths in dependent sums



Internalizing logic

Classically, mathematics consists of two distinct activities:

1. Defining things, and
2. Proving statements about them.

In homotopy type theory, the basic activity is constructing terms belonging to types.

1. Defining a type = constructing a term in Type
2. Defining a function = constructing a term in $A \to B$
3. ...
4. Stating a theorem = constructing a type that is an h-prop
5. Proving a theorem = constructing a term in an h-prop

H-props

Definition

An h-proposition (or h-prop) is an h-set that is a *subsingleton* (any two points are equal).

- In classical logic, an h-prop is "either empty or contractible".
- These are the "truth values" for embedding logic in homotopy type theory.

Constructing h-props

Recall: propositions are built from "proposition constructors":

Types	$\longleftrightarrow$	Propositions
$A \times B$	$\longleftrightarrow$	P and Q
$A + B$	$\longleftrightarrow$	P or Q
$A \to B$	$\longleftrightarrow$	P implies Q
unit	$\longleftrightarrow$	$\top$ (true)
$\emptyset$	$\longleftrightarrow$	$\bot$ (false)
$\prod_{x:A} B(x)$	$\longleftrightarrow$	$(\forall x : A) P(x)$
$\sum_{x:A} B(x)$	$\longleftrightarrow$	$(\exists x : A) P(x)$

BUT: not all of these type constructors preserve h-props.

Constructing h-props

The following are h-props:

- unit and $\emptyset$ (true and false)
- $A \times B$, if A and B are h-props (and)
- $A \to B$, if A and B are h-props (implies)
- $\prod_{x:A} B(x)$, if each $B(x)$ is an h-prop (for all)

These are not:

- $A + B$, even if A and B are h-props (or)
- $\sum_{x:A} B(x)$, even if each $B(x)$ is an h-prop (there exists)
- $(x = y)$ for $x, y : A$, unless A is an h-set.

Supports

In set theory, subsingletons are a reflective subcategory of sets, and even of ∞-groupoids.

Definition

The support of A, denoted $\mathrm{supp}(A)$, is a subsingleton that contains a point precisely when A does.

Eventually, we'll need a type constructor that does this. But let's see how far we can get without it. (This will also tell us how to formulate that type constructor.)

Internalizing h-props

Let's try to internalize "A is an h-prop":

1. for all $x, y \colon A$, there exists a path $x \rightsquigarrow y$

$$\prod_{x \colon A} \prod_{y \colon A} \mathrm{supp}(x = y)$$

2. for all $x, y \colon A$ and paths $p, q \colon x \rightsquigarrow y$, there exists a 2-path $p \rightsquigarrow q$.

$$\prod_{x \colon A} \prod_{y \colon A} \prod_{p \colon (x=y)} \prod_{q \colon (x=y)} \mathrm{supp}(p = q)$$

3. for all $x, y \colon A$, paths $p, q \colon x \rightsquigarrow y$, and 2-paths $r, s \colon p \rightsquigarrow q$, there exists a 3-path $r \rightsquigarrow s$.
4. ...

Internalizing h-props

$$\prod_{x \colon A} \prod_{y \colon A} \mathrm{supp}(x = y)$$

is the h-prop "for all $x, y \colon A$ there exists a path from x to y"

$$\prod_{x \colon A} \prod_{y \colon A} (x = y)$$

is the type of functions which assign to any pair $x, y \colon A$ a path from x to y, "varying continuously" with x and y.

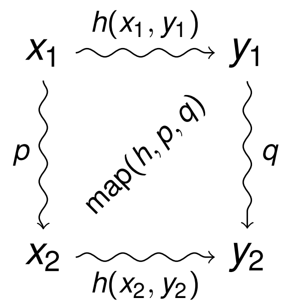
Such a function implies the former h-prop, but also more...

Internalizing h-props

Suppose $h\colon \prod_{x\colon A} \prod_{y\colon A} (x = y)$.

What does it mean that $h(x, y)$ "varies continuously" with x, y?

1. It takes paths to paths: for $p\colon (x_1 = x_2)$ and $q\colon (y_1 = y_2)$:



2. It takes 2-paths to 2-paths. . .

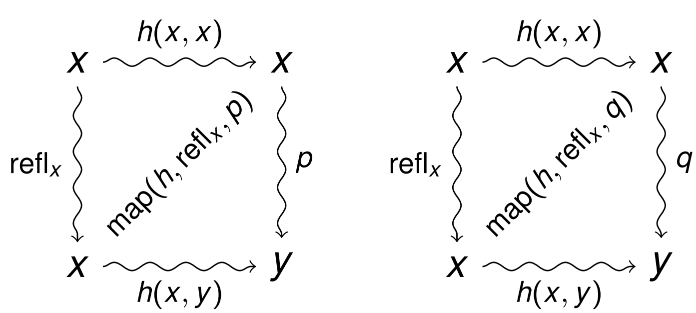
These are all things we expect to exist anyway in an h-prop!

Internalizing h-props

In fact, $\prod_{x\colon A} \prod_{y\colon A} (x = y)$ is also sufficient to make A an h-prop!

Example

Suppose $p, q\colon (x = y)$.



$$h(x, x) \mathbin{@} p = h(x, y) = h(x, x) \mathbin{@} q$$
$$p = q$$

Internalizing h-props

Thus, it would be enough to define

$$\mathrm{isProp}(A) :\equiv \mathrm{supp}\left(\prod_{x:A}\prod_{y:A}(x = y)\right).$$

But amazingly, $\prod_{x:A}\prod_{y:A}(x = y)$ is already an h-prop, even though $(x = y)$ is not!

Definition

$$\mathrm{isProp}(A) :\equiv \prod_{x:A}\prod_{y:A}(x = y)$$

Theorem

For any A, we can construct a term in

$$\mathrm{isProp}(\mathrm{isProp}(A)).$$

Homotopy levels

"Definition"

A type is *n*-truncated if it has no nontrivial k-paths for any $k > n$.

- Sets are the 0-truncated types.
- S^1 is 1-truncated.
- The type of sets (that is, the type whose points are sets) is 1-truncated.
- The type of n-truncated types is $(n + 1)$-truncated.
- S^2, and the type Type of all types, are not n-truncated for any n.

Negative thinking

Observations

- A k-path in A is a $(k-1)$-path in $(x = y)$ for some $x, y : A$.
- Thus A is n-truncated $\iff$ $(x = y)$ is $(n-1)$-truncated for all $x, y : A$.

We've seen that if A is 0-truncated, then $(x = y)$ is an h-prop. Thus it makes sense to define

Definition

A type is (-1)-truncated if it is an h-prop.

Internalizing truncation

By induction, starting with $n = (-1)$:

Definition

A type A is

- (-1)-truncated if it is an h-prop, and
- $(n+1)$-truncated if $(x = y)$ is n-truncated for all $x, y : A$.

```
Fixpoint isTrunc (n:nat) (A:Type) : Type :=
  match n with
    | -1   => isProp A
    | S n' => forall (x y:A), isTrunc n' (x == y)
  end.
```

More negative thinking

What can we say about $(x = y)$ if A is an h-prop?

- it is an h-prop.
- it is inhabited.

Definition

A type is contractible, or (-2)-truncated, if it is an inhabited h-prop.

(After this point, it's "turtles all the way down": (-3)-truncated is the same as (-2)-truncated.)

Alternative contractibility

Suppose A is contractible; thus we have $a\colon A$ and

$$h\colon \mathsf{isProp}(A) :\equiv \prod_{x\colon A}\prod_{y\colon A}(x = y).$$

Then

$$(a, h(a))\colon \sum_{x\colon A}\prod_{y\colon A}(x = y).$$

Conversely, if

$$(a, k)\colon \sum_{x\colon A}\prod_{y\colon A}(x = y)$$

then $a\colon A$ and

$$\lambda x^A y^A.\big(!k(x) \mathbin{@} k(y)\big) : \mathsf{isProp}(A).$$

Alternative contractibility

It turns out that

$$\mathrm{isContr}(A) :\equiv \sum_{x:\,A} \prod_{y:\,A} (x = y)$$

is also always an h-prop. So we can start the induction at -2:

Definition

A type A is

- (-2)-truncated if it is contractible, and
- $(n+1)$-truncated if $(x = y)$ is n-truncated for all $x, y : A$.

This is what we usually do in practice.

Definition

A type has h-level n if it is $(n-2)$-truncated.

Homotopy equivalences

Definition

A function $f : A \to B$ is a homotopy equivalence if there exists $g : B \to A$ and homotopies $g \circ f \sim \mathrm{id}_A$ and $f \circ g \sim \mathrm{id}_B$.

$$\mathrm{isEquiv}(f) :\equiv \mathrm{supp}\left(\sum_{g:\,B \to A} \big((g \circ f = \mathrm{id}_A) \times (f \circ g = \mathrm{id}_B) \big) \right)$$

This would not be an h-prop without supp. Can we avoid it?

Back to bijections

A function $f\colon A \to B$ between sets is a bijection if

1. There exists $g\colon B \to A$ such that $g \circ f = \mathrm{id}_A$ and $f \circ g = \mathrm{id}_B$.
2. OR: For each $b \in B$, the set $f^{-1}(b)$ is a singleton.
3. OR: There exists $g\colon B \to A$ such that $g \circ f = \mathrm{id}_A$ and also $h\colon B \to A$ such that $f \circ h = \mathrm{id}_B$.

Better equivalences

Definition

The homotopy fiber of $f\colon A \to B$ at $b\colon B$ is

$$\mathrm{hfiber}(f, b) :\equiv \sum_{x\colon A} (f(x) = b).$$

Definition (Voevodsky)

f is an equivalence if each $\mathrm{hfiber}(f, b)$ is contractible:

$$\mathrm{isEquiv}(f) :\equiv \prod_{b\colon B} \mathrm{isContr}(\mathrm{hfiber}(f, b))$$

This is an h-prop.

H-isomorphisms

Definition (Joyal)

$f\colon A \to B$ is an h-isomorphism if we have $g\colon B \to A$ and a homotopy $g \circ f \sim \mathrm{id}_A$, and also $h\colon B \to A$ and a homotopy $f \circ h \sim \mathrm{id}_B$.

$$\mathrm{isHIso}(f) :\equiv \left(\sum_{g\colon B \to A} (g \circ f = \mathrm{id}_A) \right) \times \left(\sum_{h\colon B \to A} (f \circ h = \mathrm{id}_B) \right)$$

This is also an h-prop.

Adjoint equivalences

Given a homotopy equivalence, we can also ask for more coherence from $r\colon (g \circ f = \mathrm{id}_A)$ and $s\colon (f \circ g = \mathrm{id}_B)$.

(1a) For all $b\colon B$, we have $u(b)\colon (r(g(b)) = \mathrm{map}(g, s(b)))$.

(1b) For all $a\colon A$, we have $v(a)\colon (\mathrm{map}(f, r(a)) = s(f(a)))$.

(2a) For all $b\colon B$, we have $\ldots v(g(b) \ldots \mathrm{map}(g, u(b)) \ldots$

(2b) For all $a\colon A$, we have $\ldots u(f(a) \ldots \mathrm{map}(f, v(a)) \ldots$

$\vdots$

This gives an h-prop if we stop between any (n a) and (n b).

Definition

f is an adjoint equivalence if we have g, r, s, and u.

$$\mathrm{isAdjEquiv}(f) :\equiv \sum_{g\colon B \to A} \sum_{r\colon \ldots} \sum_{s\colon \ldots} \Big(r(g(b)) = \mathrm{map}(g, s(b)) \Big)$$

All equivalences are the same

Theorem

The following are equivalent:

1. *f is a homotopy equivalence.*
2. *f is a (Voevodsky) equivalence.*
3. *f is a (Joyal) h-isomorphism.*
4. *f is an adjoint equivalence.*

The last three are supp-free h-props, so we have equivalences

$$\mathrm{isEquiv}(f) \simeq \mathrm{isHIso}(f) \simeq \mathrm{isAdjEquiv}(f)$$

Definition

The type of equivalences between A, B: Type is

$$\mathrm{Equiv}(A, B) :\equiv \sum_{f\colon A \to B} \mathrm{isEquiv}(f).$$

The univalence axiom

For A, B: Type, we have

$$\mathrm{pathToEquiv}_{A,B} \colon (A = B) \to \mathrm{Equiv}(A, B)$$

defined by induction on paths.
Note: $(A = B)$ is a path-type of "Type".

Axiom (Univalence)

For all A, B, the function $\mathrm{pathToEquiv}_{A,B}$ is an equivalence.

$$\prod_{A\colon \mathrm{Type}} \prod_{B\colon \mathrm{Type}} \mathrm{isEquiv}(\mathrm{pathToEquiv}_{A,B})$$

In particular, every equivalence yields a path between types.

Remarks about univalence

1. Univalence implies function extensionality (Voevodsky).
2. Would like to formulate univalence (and, hence, function extensionality) "computationally". Some progress is being made (Harper-Licata).
3. In set-theoretic models, univalence should correspond to "object classifiers" in "$(\infty, 1)$-toposes" (Rezk, Lurie)
4. So far, only a few actual models known (coherence issues).
5. Many other uses.