Homotopy levels

Samuel Mimram

2025

École polytechnique

Homotopy levels

In topology, we have the **spheres**:

$S^0 =$ $S^1 =$  $S^2 =$  $\ldots$

and the disks

$D^0 =$ $D^1 =$  $D^2 =$  $\ldots$

which come equipped with a canonical inclusion

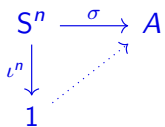
$$\iota^n : S^n \hookrightarrow D^n$$

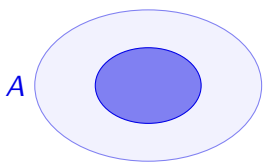
Note: all disks are contractile and thus homotopy equivalent to a point, i.e. $D^n = 1$.

Homotopy levels

Given a space A, an n-**sphere** in A is a map $\sigma : S^n \to A$.

An n-sphere is **contractible** when this maps extends to 1:

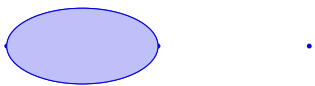




A type is an n-**type** (or n-**truncated**) when all its k-spheres are contractible for $k > n$.

Homotopy levels

For instance, in a 0-type all k-spheres are contractible:



A 0-type is thus a space in which between any two points there is at most one path (up to homotopy).

This kind of observation can be used in order to define n-types in a nice way.

Homotopy levels

In low dimensions, n-types have names

-2	contractible types
-1	propositions
0	sets
1	groupoids

We will begin by studying those before the general case.

Part I

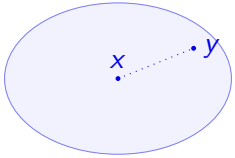
Contractible types

Contractible types

A type A is **contractible** when it satisfies

$$\mathsf{isContr}\ A \quad \hat{=} \quad \Sigma(x : A).\Pi(y : A).(x = y)$$

The point x is the *center of contraction*.



Contractible types

The canonical example of a contractible type is 1.

Namely, we can show

$$\mathsf{isContr}\ 1 \quad \hat{=} \quad \Sigma(x : 1).\Pi(y : 1).(x = y)$$

by

$$(\star, f)$$

with f defined by induction by

$$f : (y : 1) \to (\star = y)$$
$$\star \mapsto \mathsf{refl}$$

This is fact, essentially, the only contractible type.

Bool is not contractible

Lemma
Bool *is not contractible.*

Proof.
Suppose that Bool is contractible. There is a point $b_0 : \mathsf{Bool}$ and a family of paths

$$p : (b : \mathsf{Bool}) \to b_0 = b$$

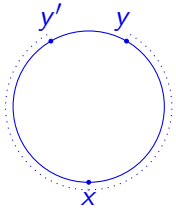
We thus have

$$\mathsf{true} \overset{(p\,\mathsf{true})^{-}}{=\!=} b_0 \overset{p\,\mathsf{false}}{=\!=} \mathsf{false}$$

but we know that this is not the case. □

S^1 is not contractible

It might seem that

$S^1 \quad \hat{=}$ 

is contractible, but this is not the case!

Namely, we have to construct a function

$$(y : S^1) \to (x = y)$$

which, as any function, has to be continuous.

In topology, this is the difference between a **contractible** and **path connected** space.

Path spaces of contractible types

You might have the false impression that everything can be solved with J / path induction:

Proposition?
Given a contractible type A and paths $p\,q : x = y$, we have $p = q$.

Proof.

By path induction on q, it is enough to show that for every path $p : x = x$ we have $p = \mathsf{refl}$. We cannot do a path induction on p! □

Path spaces of contractible types

Proposition ([Uni13, Lemma 3.11.10])
Given a contractible type A and $x\,y : A$, we have that $x = y$ is contractible.

Proof.

We have a center of contraction $x_0 : A$ and family of paths $p : (x : A) \to x_0 = x$.

For $x = y$, we choose as center of contraction $q_0 \hat{\equiv} (p\,x)^{-} \cdot (p\,y)$.

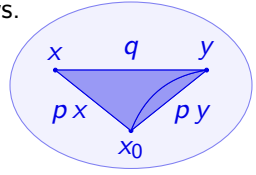
Suppose given a path $q : x = y$, we need to show $p_0 = q$.

By path induction on q, it is enough to show $(p\,x)^{-} \cdot (p\,x) = \mathsf{refl}$ which does hold.

Alternatively, by apd [Uni13, Lemma 2.3.4] and transport in path types [Uni13, Lemma 2.11.2], we have

$$p\,y = \mathsf{transport}\,(\lambda x.x_0 = x)\,q\,(p\,x) = p\,x \cdot q$$

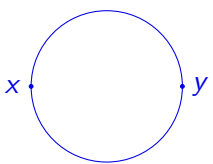
and we conclude by groupoid laws. □



Path spaces of contractible types

Proposition ([Uni13, Lemma 3.11.10])
Given a contractible space A and $x\,y : A$, we have that $x = y$ is contractible.

This also shows that S^1 is not contractible:



Clearly $x = y$ is not contractible!

Singletons

Given $x : A$, we define the type **singleton** at x as

$$\text{singl } x \quad \hat{=} \quad \Sigma(y : A).(x = y)$$

At first it might seem that singl x is the *connected component* of x in A.
This is not the case because we keep the information of the path!

TODO: circle..... (not the connected component) helix

Singleton

Proposition ([Uni13, Lemma 3.11.8])
Given $x : A$, the type $\mathsf{singl}\, x \hat{\equiv} \Sigma(y : A).(x = y)$ is contractible.

Proof.
As center of contraction, we take (x, refl).

Suppose given (y, p) with $p : x = y$, we need to show $(x, \mathsf{refl}) = (y, p)$ in $\Sigma(y : A).(x = y)$. This can be done by taking $p : x = y$ and constructing a path

$$q : \mathsf{transport}\,(\lambda y.x = y)\, p\, \mathsf{refl} = p$$

By transport in path types [Uni13, Lemma 2.11.2], we have

$$\mathsf{transport}\,(\lambda y.x = y)\, p\, \mathsf{refl} = \mathsf{refl} \cdot p$$

and we conclude by unitality. □

The 1 contractible type

Are there contractible types other than 1?

For now, we do not have a way to show that $\mathsf{isContr}\ A \to (A = 1)$ or even $\mathsf{I} = 1$.

But we will once we assume **univalence**, so that you can safely suppose this.

Part II

Propositions

Propositions

A type A is a **proposition** when it satisfies

$$\mathsf{isProp}\ A \quad \hat{=} \quad (x\, y : A) \to (x = y)$$

For instance,

- 0 is a proposition
- 1 is a proposition
- Bool and $\mathbb{N}$ are not propositions
- S^n are not propositions

Contractible propositions

Lemma
Every contractible type is a proposition.

Proof.

Given a contractible type A and $x\,y : A$, we know that $x = y$ is contractible and thus inhabited.

More explicitly, writing (x_0, p) for the contraction of A, with $p : (x : A) \to (x_0 = x)$, we can take

$$x \overset{(p\,x)^-}{=\!=} x_0 \overset{p\,y}{=\!=} y$$

□

Propositions as contractible types

What does a proposition look like? It is either empty or contractible. Excepting that we cannot say this like this in intuitionistic logic:

$$\mathsf{isProp}\ A \to (A = 0) \sqcup \mathsf{isContr}\ A$$

is not expected to hold.

Namely, take A to be "the n-th Turing machine halts": this is a proposition!

Propositions as contractible types

However one can show that

Lemma
For $A : \mathcal{U}$, we have that isProp A *is logically equivalent to* $A \to$ isContr A.

Proof.

Suppose $p :$ isProp A and $x : A$, then we can contract A with $(x, \lambda y.p\,x\,y)$.

Conversely suppose $f : A \to$ isContr A. Given $x\,y : A$, we have to show $x = y$.
By $f\,x$, we have that A is contractible, therefore $x = y$ is also contractible, and thus inhabited. □

Proposition thus deserve their name: they are formulas which can be proved in a unique way (when they can).

Path spaces of propositions

Lemma ([Uni13, Lemma 3.11.10])
A proposition A has contractible path spaces: $\mathsf{isContr}\,(x = y)$ *for* $x\,y : A$.

Proof.

Suppose given $x\,y : A$. By previous proposition, we have $\mathsf{isContr}\ A$. Therefore $\mathsf{isContr}(x = y)$. □

Proposition

General fact
Whenever you introduce something which looks like a predicate, you should check that this is a family of propositions: what matters is that $\mathsf{bla}(x)$ holds and not which proof of $\mathsf{bla}(x)$ we gave.

In the following, we will assume this:

Temporary axiom
We assume **function extensionality**: given functions $f\ g : A \to B$ if

$$(x : A) \to f\ x = g\ x$$

then $f = g$.

It is "temporary", because it will follow from univalence.

Being a contractible is a proposition

Proposition ([Uni13, Lemma 3.3.5])
Being contractible is a proposition: $\mathsf{isProp}\,(\mathsf{isContr}\ A)$.

Proof.

Suppose given two proofs (x, p) and (y, q) of $\mathsf{isContr}\ A$. We have $p\,y : x = x$ and we need to show

$$p \quad =^{\lambda x.(y:A)\to x=y}_{p\,y} \quad q$$

By funext this amounts to show, for $z : A$

$$\mathsf{transport}\,(\lambda x.(y : A) \to x = y)\,(p\,y)\,p\,z = q\,z$$

in $y = z$. But A is contractile, therefore $x = z$ also, which is thus a proposition. □

Being a proposition is a proposition

Proposition ([Uni13, Lemma 3.3.5])
Being a proposition is a proposition: $\mathsf{isProp}\,(\mathsf{isProp}\ A)$.

Proof.

Suppose given proofs P and Q of $(x\,y : A) \to x = y$. By funext, given $x\,y : A$, we need to show

$$P\,x\,y = Q\,x\,y$$

in $x = y$. but A is a proposition, therefore $x = y$ is contractible, and thus a proposition. □

Subtypes

Given a type A and a family $P : A \to \mathcal{U}$ of propositions, we can form the **subtype** $\Sigma(x : A).P\,x$ of A.

Proposition ([Uni13, Lemma 3.5.1])
The canonical inclusion / first projection $\mathsf{fst} : \Sigma A.P \to A$ *is an "injection".*

Proof.

Suppose given (x, π) and (y, ρ) in $\Sigma A.P$ together with $p : x = y$. We can construct an equality between them by constructing

$$\mathsf{transport}\ P\,p\,\pi \quad = \quad \rho$$

which follows from the fact that P is a proposition. □

HProp

The type of **propositions** is

$$\mathsf{HProp} \quad \hat{=} \quad \Sigma(A : \mathcal{U}).\, \mathsf{isProp}\ A$$

This is a subtype of $\mathcal{U}$.

Closure properties: product

We have seen that 0 / 1 are propositions, understood as $\bot$ / $\top$.
What about connectives?

Proposition
Propositions are closed under $\times$. We thus have an induced operation

$$\wedge : \mathrm{HProp} \to \mathrm{HProp} \to \mathrm{HProp}$$

Proof.
Given A and B which are propositions and (x, y) and (x', y') in $A \times B$, we have

- $p : x = x'$ since A is a proposition
- $q : y = y'$ since B is a proposition

and thus $\mathrm{pair}^{=}\ p\, q : (x, y) = (x', y')$. □

Closure properties: implication

Proposition
The type $A \to B$ is a proposition when B is. We thus have an induced operation

$$\Rightarrow : \mathsf{HProp} \to \mathsf{HProp} \to \mathsf{HProp}$$

Proof.

Given $f\ g : A \to B$, we want to show $f = g$. By funext it is enough to show $f\ x = g\ x$ for an arbitrary $x : A$, which holds because B is a proposition. □

In particular negation is always a proposition:

$$\neg A \quad \hat{=} \quad A \to \bot$$

Closure properties: universal quantification

Previous theorem generalizes to depedent types:

Proposition
Given a type A and a family $B : A \to \mathcal{U}$ of propositions, the type

$$\Pi(x : A).B\,x$$

is a proposition. We thus have an induced operation

$$\forall : (A : \mathcal{U}) \to (B : A \to \mathsf{HProp}) \to \mathsf{HProp}$$

Propositional extensionality

We define equivalence as

$$A \Leftrightarrow B \quad \hat{=} \quad (A \Rightarrow B) \wedge (B \Rightarrow A)$$

The **proposition extensionality** principle [WR27, Chu40] states that

$$(A \Leftrightarrow B) \to (A = B)$$

This principle is compatible with the current theory, and in fact will be implied by **univalence** which is a generalization of it.

Closure properties: coproducts

Are propositions closed under coproducts?

No: $2 = 1 \sqcup 1$ is not a set.

Proposition
Given propositions A and B such that $\neg(A \times B)$, we have $A \sqcup B$ a proposition

Proof.
Given elements $x\, x' : A \sqcup B$, we want to show that $x = x'$:

x	x'	$x = x'$
a	a'	we have $a = a'$ by A proposition
b	b'	we have $b = b'$ by B proposition
a	b'	impossible by $\neg(A \times B)$
a'	b	impossible by $\neg(A \times B)$

□

Closure properties: dependent sums

Propositions are not closed under Σ-types: given a family $B : A \to \mathcal{U}$ of proposition,

$$\Sigma(x : A).B\,x$$

is not a proposition in general. For instance,

$$\mathbb{N} \quad = \quad \Sigma(x : \mathbb{N}).1$$

is not a proposition.

Propositional truncation

In order to correct this, we need an operation which turns a type A into a proposition $\|A\|_{-1}$:

- when $A = 0$, we expect $\|A\|_{-1} = 0$,
- otherwise, we expect $\|A\|_{-1}$ to be contractible.

Excepting that we cannot reason like this in intuitionistic logic.

Instead, we will introduce $\| - \|_{-1}$ as a new type constructor!

Constructor:

$$\| - \|_{-1} : \mathcal{U} \to \mathcal{U}$$

Introduction rules:

$$| - |_{-1} : A \to \|A\|_{-1} \qquad \text{pt} : \text{isProp}\,(\|A\|_{-1})$$

The recursor is (the eliminator is not more useful)

$$\text{rec} : (B : \mathcal{U}) \to (A \to B) \to \text{isProp}\ B \to \|A\|_{-1} \to B$$

Informally, if you have an $a : \|A\|_{-1}$ and you are trying to prove a proposition B, you can safely assume that you actually have $a : A$.

Otherwise said $a : \|A\|_{-1}$ is in a box that you are only allowed to open when proving a proposition. Computation rule is

$$\text{rec}\ B\ f\ \text{pt}\ |a|_{-1} \quad \hat{=} \quad f\ a$$

Merely

We say that A **merely** holds when we have $\|A\|_{-1}$.

The missing connectives

We can thus define, for A and B propositions,

$$A \vee B \quad \hat{=} \quad \|A \sqcup B\|_{-1}$$

and for $A : \mathcal{U}$ and $B : A \to \mathcal{U}$ a family of propositions,

$$\exists(x : A).B\,x \quad \hat{=} \quad \|\Sigma(x : A).B\,x\|_{-1}$$

The Curry-Howard correspondence

Logic	Type theory	Propositions
Formula	Type	Proposition
$\top$	1	1
$\bot$	0	0
$A \wedge B$	$A \times B$	$A \times B$
$A \Rightarrow B$	$A \to B$	$A \to B$
$A \vee B$	$A \sqcup B$	$\|A \sqcup B\|_{-1}$
$\forall(x : A).B(x)$	$(x : A) \to B(x)$	$(x : A) \to B(x)$
$\exists(x : A).B(x)$	$\Sigma(x : A).B(x)$	$\|\Sigma(x : A).B(x)\|_{-1}$

Path connected types

connected

connected component

the connected component is connected

S^1 is path connected $(y : A) \to \|x = y\|_{-1}$

connected component (subtypes are embeddings) [Uni13, Lemma 3.5.1]

the image

.....

Double negation

Propositional truncation is close to double negation: there is a canonical map

$$\|A\|_{-1} \to \neg\neg A$$

The converse does not hold unless A is "classical" in the sense that $\neg\neg A \to A$.

Propositional truncation is thus an "intuitionistic" variant of double negation.

The law of excluded middle.....

General excluded middle is inconsistent.....

The axiom of choice

The propositional truncation forgets abuts proofs only remember provability.

However, we can nevertheless sometimes extract information from it.

Extracting from truncation

Proposition ([Uni13, Exercise 3.19])
Suppose that $P : \mathbb{N} \to \mathcal{U}$ is a decidable proposition: for $n : \mathbb{N}$, we have $P\,n \sqcup \neg(P\,n)$ and $\mathsf{isProp}\,(P\,n)$. Then

$$\exists(n : \mathbb{N}).P\,n \to \Sigma(n : \mathbb{N}).P\,n$$

Proof.
Typically: $P\,n \hat{=} (f\,n = 0)$ for some function $f : \mathbb{N} \to \mathbb{N}$.
Consider the predicate $\mathsf{isFirst}\,n \equiv (P\,n) \times ((m : \mathbb{N}) \to m < n \to \neg(P\,m))$. This is a proposition and thus $\Sigma(n : \mathbb{N}).\,\mathsf{isFirst}\,n$ is also a proposition. We can then construct a function which, given a solution, finds the first one:

$$\mathsf{findFirst} : (m : \mathbb{N}) \to P\,m \to \Sigma(n : \mathbb{N}).\,\mathsf{isFirst}\,n$$

Now, suppose given $\exists(n : \mathbb{N}).P\,n$ i.e. $\|\Sigma(n : \mathbb{N}).P\,n\|_{-1}$ and want $\Sigma(n : \mathbb{N}).\,\mathsf{isFirst}\,n$. Since this is a proposition, we can suppose given and element of $\Sigma(n : \mathbb{N}).P\,n$. By $\mathsf{findFirst}$, we can construct an element of $\Sigma(n : \mathbb{N}).\,\mathsf{isFirst}\,n$ and thus $\Sigma(n : \mathbb{N}).P\,n$. □

Agda notation

In Agda, $\|A\|_{-1}$ is noted

$$\| \ \mathtt{A} \ \|_1$$

(apparently "-" is difficult to type).

Propositional truncation: implementation

One way of implementing propositional truncation is axiomatically which we can do in Agda using rewriting rules:

```
postulate
  ∥_∥₁ : Type → Type
  |_|₁ : {A : Type} → A → ∥ A ∥₁
  isPropPropTrunc : {A : Type} → isProp ∥ A ∥₁
  propTrunc-rec :
    {A : Type} {B : Type} → isProp B → (A → B) → (∥ A ∥₁ → B)
  propTrunc-beta :
    {A : Type} {B : Type}
    (P : isProp B) (f : A → B) (x : A) → propTrunc-rec P f | x |₁ ≡ f x
```

Propositional truncation: implementation

Another (better) way is to use higher inductive types (once we have them):

```
data ‖_‖₁ (A : Type) : Type where
  |_|₁ : A → ‖ A ‖₁
  squash₁ : (x y : ‖ A ‖₁) → x ≡ y
```

Note: this is a recursive type.

$$\| \mathsf{Bool} \|_{-1} \quad = \quad$$

Propositional truncation: impredicative encoding

A last way to encode propositional truncation is the **impredicative encoding** which consists in defining truncation by its recursion principle:

$$\|A\|^{\ell}_{-1} \quad \hat{=} \quad (B : \mathcal{U}_{\ell}) \to \mathsf{isProp}\ B \to (A \to B) \to B$$

This can be shown to be a proposition (because B is) and the inclusion is

$$|x|_{-1}\, B\, \pi\, f \quad \hat{=} \quad f\, x$$

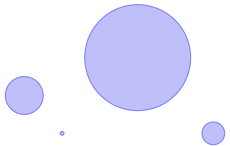
However, we can only eliminate at a specified level ℓ!

Part III

Sets

Sets

A set is a collection of points (or, rather, contractible types):



We thus define a **set** to be a type A satisfying

$$\mathsf{isSet}\ A \mathrel{\hat{=}} (x\ y : A) \to \mathsf{isProp}\,(x = y)$$

For instance:

- are sets: 0, 1, Bool, $\mathbb{N}$, HProp, ...
- are not sets: S^n with $n > 0$, ...

Booleans are sets

Proposition(?)
The type Bool is a set.

Proof.
We have to show $(x\, y : \mathsf{Bool}) \to (p\, q : x \to y) \to p = q$.

By induction on q, it is enough to show $(x : \mathsf{Bool}) \to (p : x \to x) \to p = \mathsf{refl}$.

With univalence, we will be able to show that $(x : \mathsf{Bool}) \to \mathsf{isContr}(x = x)$, but we do not have the tools to show this directly for now. Another proof follows. □

Propositions are sets

Proposition ([Uni13, Lemma 3.3.4])
Any proposition A is a set: $\mathsf{isProp}\ A \to \mathsf{isSet}\ A$.

Proof.
Suppose given a proposition A, $x\,y : A$ and $p\,q : x = y$. We have seen that $x = y$ is contractible and thus a proposition. Thus $p = q$. □

Being a set is a proposition

Proposition ([Uni13, Lemma 3.3.5])
Being a set for a given type is a proposition: $\mathsf{isProp}\,(\mathsf{isSet}\ A)$.

Proof.
We want to show $\mathsf{isProp}((x\,y : \mathsf{Bool}) \to \mathsf{isProp}(x = y))$.

This amounts to show $\mathsf{isProp}(\mathsf{isProp}(x = y))$ for fixed x and y.

Which does hold. □

Sets

We write

$$\mathsf{HSet} \quad \hat{=} \quad \Sigma(A : \mathcal{U}).\, \mathsf{isSet}\ A$$

for the type of sets.

Closure properties

We have that

- $A \times B$ is a set when A and B are sets
- $A \sqcup B$ is a set when A and B are sets
- $A \to B$ is a set when B is a set
- $(x : A) \to B\,x$ is a set when the $B\,x$ are sets
- $\Sigma(x : A).B\,x$ is a set when A and the $B\,x$ are sets

These operations thus induce operations on HSet.

In particular, given a set A and a predicate P, we have

$$\Sigma(x : A).P\,x$$

which plays the role of a subset.

Hedberg's theorem

A type A is **decidable** when $A \sqcup \neg A$ holds.

A type A is **discrete** when it has decidable equality: $x = y$ is decidable for every $x\, y : A$.

The following is known as **Hedberg's theorem**:

Theorem ([Hed98],[Uni13, Theorem 7.2.5])
Every discrete type is a set.

Note: see [Esc04] if you are curious about topological terminology.

Hedberg's theorem

A type A is **stable** when $\neg\neg A \to A$.

A type A is **separated** when it has stable equality: $x = y$ is stable for every $x\, y : A$.

Lemma
Any decidable type is stable.

Proof.
Suppose A is decidable, i.e. $A \sqcup \neg A$. Supposing $\neg\neg A$, we have to show A.

- If A holds then we conclude immediately.
- If $\neg A$ holds then we deduce $\bot$ and thus A. □

Corollary
Any discrete type is separated.

Lemma
Any separated type is a set.

Proof.
Suppose given two types $p\,q : x = y$ in A. We want to show that they are equal. Instead, we will show that both are equal to a "canonical" one.

Since we have $p : x = y$, we have $\overline{p} : \neg\neg(x = y)$ and thus $\tilde{p} : x = y$ by separation. This path is canonical in the following sense. Because $\neg\neg(x = y)$ is a proposition, we have $\overline{p} = \overline{q}$ and thus $\tilde{p} = \tilde{q}$ (for arbitrary p and q in $x = y$).

Our aim is now to show that $p = \tilde{p}$. By induction on p, we have to show $\mathsf{refl} = \tilde{\mathsf{refl}}$, which has no reason to hold!

Instead, we show

$$p = \tilde{p} \cdot \tilde{\mathsf{refl}}^{-} = \tilde{q} \cdot \tilde{\mathsf{refl}}^{-} = q$$

The first equality is shown by induction on p, which amounts to $\mathsf{refl} = \tilde{\mathsf{refl}} \cdot \tilde{\mathsf{refl}}^{-}$. □

Examples of sets

Corollary
The types Bool, $\mathbb{N}$, Fin n *are sets.*

Proof.
For instance, for Bool, it is enough to show that for $x\, y : \mathsf{Bool}$, we have $(x = y) \sqcup \neg(x = y)$, which can be done by case analysis:

x	y	$(x = y) \sqcup \neg(x = y)$
false	false	$x = y$ by refl
true	true	$x = y$ by refl
false	true	$\neg(x = y)$
true	false	$\neg(x = y)$

□

Set truncation

.....

Connected components

We have

$$\pi_0(X) = \|X\|_0$$

The fundamental group

with set truncation

$$\pi_1(A) = \|\Omega A\|_0$$

Part IV

n-types

Groupoids

A **groupoid** is a type which has sets of paths:

$$\mathsf{isGroupoid}\ A \quad \hat{=} \quad (x\ y : A) \to \mathsf{isSet}(x = y)$$

For instance,

- sets are groupoids (Bool, $\mathbb{N}$, etc.)
- S^1 is a groupoid
- S^2 is not a groupoid
- HSet is a groupoid

$\mathsf{S}^1 =$ 

$\mathsf{S}^2 =$ 

n-types [Uni13, Definition 7.1.1]

We define n-**types** as

- a (-2)-type is a contractible type,
- an $(n+1)$-type is a type A in which $x = y$ is an n-type for every $x\,y : A$.

Formally,

$$\mathsf{isType}\ n\,A \quad \hat{=} \quad \begin{cases} \mathsf{isContr}\ A & \text{if } n \hat{=} -2, \\ (x\,y : A) \to \mathsf{isType}\,(n-1)\,A & \text{otherwise.} \end{cases}$$

Proposition

Lemma ([Uni13, Lemma 3.11.10])
A (-1)-type is the same as a proposition.

Proof.
A proposition satisfies

$$(x\, y : A) \to (x = y)$$

a (-1)-type satisfies

$$(x\, y : A) \to \mathsf{isContr}(x = y)$$

Clearly, a (-1)-type is a proposition (project to the contraction center).
Conversely, we have seen that a proposition has contractible path spaces. □

Being an n-type is a proposition

Theorem ([Uni13, Theorem 7.1.10])
Being an n-type is a proposition.

Proof.
By induction on n.

For the base case, we already know that being contractible is a proposition.

For the inductive case, we have to show

$$\mathsf{isProp}((x\,y : A) \to \mathsf{isType}\,(n-1)\,(x = y))$$

for which it is enough to show $\mathsf{isProp}(\mathsf{isType}\,(n-1)\,(x = y))$ which is the induction hypothesis. □

Theorem ([Uni13, Theorem 7.1.7])
Every n-type is an $(n+1)$-type.

Proof.
For the base case, we have to show that a contractible type has contractible path types, which we already did.

For the inductive case, we apply the induction hypothesis on $x = y$. □

Closure of n-types

Theorem ([Uni13, Theorems 7.1.8 and 7.1.9])
We have that

- $A \times B$ *is an* n*-type when* A *and* B *are*
- $A \sqcup B$ *is an* n*-type when* A *and* B *are*
- $A \to B$ *is an* n*-type when* B *is*
- $\Sigma A.B$ *is an* n*-type when* A *and the* $B\,x$ *are*
- $\Pi A.B$ *is an* n*-type when the* $B\,x$ *are*

Proof.
Not enough tools for now □

The type of n-types

Theorem ([Uni13, Theorem 7.1.11])
The type of n-types is an $(n+1)$-type.

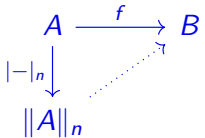
Proof.
Not enough tools for now □

Higher truncations

We can define the n-truncation $\|A\|_n$ of a type, which comes equipped with

$$|-|_n : A \to \|A\|_n$$

Given an n-type B, a map $f : A \to B$ extends uniquely as a map $\|A\|_n \to B$:



Bibliography i

[Chu40] Alonzo Church.

A formulation of the simple theory of types.

The journal of symbolic logic, 5(2):56–68, 1940.
doi:10.2307/2266170.

[Esc04] Martín Escardó.

Synthetic topology of data types and classical spaces.

Electronic Notes in Theoretical Computer Science, 87:21–156, 2004.
https://martinescardo.github.io/papers/entcs87.pdf,
doi:10.1016/j.entcs.2004.09.017.

[Hed98] Michael Hedberg.
A coherence theorem for Martin-Löf's type theory.
Journal of Functional Programming, 8(4):413–436, 1998.
doi:10.1017/S0956796898003153.

[Uni13] The Univalent Foundations Program.
Homotopy Type Theory: Univalent Foundations of Mathematics.
Institute for Advanced Study, 2013.
https://homotopytypetheory.org/book, arXiv:1308.0729.

[WR27] Alfred North Whitehead and Bertrand Russell.
Principia Mathematica to ∗56.
Cambridge University Press, 1927.
https://archive.org/details/
alfred-north-whitehead-bertrand-russel-principia-mathematica.-1/.