Homotopy Type Theory

Kalavasis Alvertos - Zaravinos Georgios

National Technical University of Athens

kalabasis.al@gmail.com - georgezar13@gmail.com

October 31, 2018

Overview

General Framework

- Homotopy type theory (HoTT) connects homotopy theory (algebraic topology) and type theory (logic, computer science).
- Consider Curry-Howard isomorphism : *a proof is a program, and the formula it proves is the type for the program.*
- HoTT illuminates a new correspondance perspective.
- HoTT starts with the idea that the points of a geometric space behave like computer programs.
- Imagine that a type is a geometric object and that terms of that type are points of that object.
- Its great insight is in systematizing the way in which these fields have all, in some sense been studying the 'same' thing.

Type theory VS Set theory

- *Issue 1 : Before constructing a 'theory', we need to use a system upon which we can derive judgements.*

Type theory VS Set theory

- *Issue 1 : Before constructing a 'theory', we need to use a system upon which we can derive judgements.*
- A deductive system is a bunch of rules for deriving judgments. Think of a deductive system as a formal game, then the judgments are the states in the game that we reach by playing according to the rules.

Type theory VS Set theory

- *Issue 1 : Before constructing a 'theory', we need to use a system upon which we can derive judgements.*
- A deductive system is a bunch of rules for deriving judgments. Think of a deductive system as a formal game, then the judgments are the states in the game that we reach by playing according to the rules.
- *Issue 2 : Set theoretic foundation.*
- It has two layers : the deductive system of first-order logic,and, formulated inside this system, the axioms of a particular theory, such as ZFC. Thus, set theory is built on top of logic.

Type theory VS Set theory

- *Issue 1 : Before constructing a 'theory', we need to use a system upon which we can derive judgements.*
- A deductive system is a bunch of rules for deriving judgments. Think of a deductive system as a formal game, then the judgments are the states in the game that we reach by playing according to the rules.
- *Issue 2 : Set theoretic foundation.*
- It has two layers : the deductive system of first-order logic,and, formulated inside this system, the axioms of a particular theory, such as ZFC. Thus, set theory is built on top of logic.
- *Issue 3 : Type theoretic foundation.*
- Type theory is its own deductive system : it needs not be formulated inside any superstructure, such as first-order logic. Anything is encoded as type. Thus, the mathematical activity of proving a theorem is identified with the construction of an object – in this case, an inhabitant of a type - that represents a proposition.

Type theory VS Set theory

- *Issue 4 : So, who wins?*
- It depends on the "application".
- The main interest in type theory comes from computer science: the creators of type theory very intentionally stuck with rules that were "computationally meaningful". This means that we can think of proofs in type theory as programs being run in a computer. Thus type theory becomes very useful in understanding how computation works.

Type theory VS Set theory

- *Issue 4 : So, who wins?*
- It depends on the "application".
- The main interest in type theory comes from computer science: the creators of type theory very intentionally stuck with rules that were "computationally meaningful". This means that we can think of proofs in type theory as programs being run in a computer. Thus type theory becomes very useful in understanding how computation works.
- *Issue 5 : And what about Category Theory?*
- Reminder : Category :: Labeled directed graph, whose nodes are called objects, and whose labelled directed edges are called morphisms.
- Type theory and certain kinds of category theory are closely related. By a syntax-semantics duality one may view type theory as a formal syntactic language for category theory, and, in reverse, one may think of category theory as providing semantics for type theory.

Are these "the same" shape?

Are these the "same" shape?

Are these the "same" shape?

Which are the rules? What "same" means?

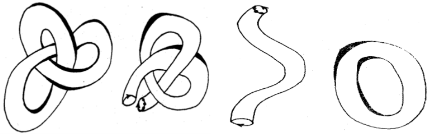
Are these "the same" shape?

Are these the "same" shape?

Which are the rules? What "same" means?

Continuous deformation and 'cuts' are only accepted.

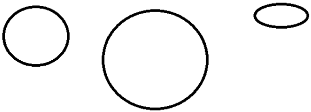
Topologically speaking



1. Knotted torus   2. Cut (the arrows show the direction of cut).   3. Unknot the cylinder.   4. Sew up again.

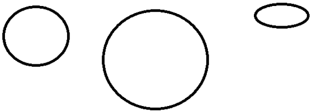
From a topologist's perspective

- Are these 'the same' or not?



From a topologist's perspective

- Are these 'the same' or not?



- In topology, we ignore the geometric "realization" of a shape. The essence of a shape is the class of all shapes it can be continuously deformed into.
- From a programming perspective, the notion of a shape is a data structure (a class) referring to that shape and a representation of the shape refers to an instance of that class.

From a programmer's (?) - mathematician's perspective

- Are these 'the same' or not?
  - $\backslash x- > x \ \ (\equiv \lambda x.x)$
  - $\backslash y- > y$
  - $(\backslash f\ x- > x)\ (\ )$
  - $(\backslash g\ y\ x- > x)\ (\ )\ (\ )$

From a programmer's (?) - mathematician's perspective

- Are these 'the same' or not?
  - $\backslash x- > x \;\; (\equiv \lambda x.x)$
  - $\backslash y- > y$
  - $(\backslash f\ x- > x)\ (\ )$
  - $(\backslash g\ y\ x- > x)\ (\ )\ (\ )$
- The notion of a function is the collections of all functions that it can be 'continuously' transformed into.
- Or maybe, we can think on how they act to the input. The function is all functions that act the same way (they have the exact response in the same input).
- Ignoring the geometric realization of a shape is like ignoring the domain of a function.
- From a programming perspective, a description of a function is a data structure (or a class) and as representation, we can enumerate functions (or objects of that class).

From a programmer's perspective

- Ignore Haskell's laziness for these data structures.
  - data Pair a = Pair a a
  - data Pair a = Pair (a,()) a
  - data Pair a = Either Void (a,a)

From a programmer's perspective

- Ignore Haskell's laziness for these data structures.
  - data Pair a = Pair a a
  - data Pair a = Pair (a,()) a
  - data Pair a = Either Void (a,a)
- The notion of a structure is the collection of all structures that it can be 'continuously' transformed into.
- Ignoring the geometric realization of a shape is like ignoring the construction of the data structure.

Type theory - 1st Building Block of HoTT

- Type theory is a deductive system based on two forms of judgement :
  1) $a : A$ *("a is an object of type A")*
  2) $a \equiv b : A$ *("a and b are definitionally equal objects of type A")*
- Construction : Given types A and B, we can construct the type $A \to B$ of mappings with domain A and codomain B.
- $\lambda-$abstraction for expression f : $(\lambda(x : A).f) : A \to B$
- Computation rule : A $\lambda-$abstraction is a function, so we can apply it to an argument $a : A$. We then have the following computation rule, which is a definitional equality: $(\lambda x.f)(a) \equiv F$ where F is the expression f in which all occurrences of x have been replaced by a.
- Uniqueness principle for function types : From any function $f : A \to B$, we can construct a $\lambda-$abstraction function $\lambda x.f(x)$ and we consider it to be definitionally equal to f.
- Currying : $f : A \to (B \to C)$

Type theory

- We need to formalize the expression "A is a type"

Universes

A universe is a type whose elements are types. To avoid the paradox of "a universe $\mathcal{U}_\infty$ of all types including itself", we introduce a hierarchy of universes $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \ldots$, where every $\mathcal{U}_i$ is an elemnent of the next universe and we assume that our universes are cumulative (if $A : \mathcal{U}_i$, then $A : \mathcal{U}_{i+1}$).

- When we say A is a type, we mean that it inhabits some universe $\mathcal{U}_i$.
- Families of types : To model a collection of types varying over a given type A, we use functions $\mathcal{B} : A \to \mathcal{U}$ whose codomain is a universe. F.e. the family of finite set $Fin : \mathbb{N} \to \mathcal{U}$, where $Fin(n)$ is a type with exactly n elements.

Type theory - Use the family *Fin* example.

- Dependent product types : Given a type $A : \mathcal{U}$ and a family $\mathcal{B} : A \to \mathcal{U}$, we may construct the type of dependent functions $\prod_{(x:A)} \mathcal{B}(x) : \mathcal{U}$. If $\mathcal{B}$ is a constant family, then $\prod_{(x:A)} \mathcal{B} \equiv (A \to B)$.
- An important class of dependent function types are functions that are polymorphic over a given universe. They take a type as one of its arguments and then act on elements of that type (take the "general" type). For example, the identity $id : \prod_{(A:\mathcal{U})} A \to A$
- Product types : Cartesian product
- Dependent pair (sum) types : We generalize product types. Given a type $A : \mathcal{U}$ and a family $\mathcal{B} : A \to \mathcal{U}$, the dependent pair type is $\sum_{(x:A)} B(x) : \mathcal{U}$. If $\mathcal{B}$ is a constant family, then $\sum_{(x:A)} \mathcal{B} \equiv (A \times B)$. The construction is done by pairing : we have $(a, b) : \sum_{(x:A)} B(x)$ for given $a : A$ and $b : B(a)$
- Coproduct types : Given $A, B : \mathcal{U}$, the coproduct type $A + B : \mathcal{U}$ corresponds to the disjoint union in set theory.

Dependent product/sum types

Informally :

- A dependent product type is just a function that maps the inhabitants $x : A$ of type A to an inhabitant within the type $B(x)$ (where type $B(x)$ depends on the input object).
  - Example : If $Vec(\mathbb{R}, n)$ for n-tuples of real numbers, then $\prod_{n:\mathbb{N}} Vec(\mathbb{R}, n)$ would be the type of a function which, given a natural number n, returns a tuple of real numbers of size n.
- The dual of the dependent product type is the dependent pair (sum) type. The dependent pair type captures the idea of an ordered pair where the type of the second term is dependent on the value of the first. If $(a, b) : \sum_{x:A} B(x)$, then $a : A$ and $b : B(a)$.

Type theory

- In type theory, proofs are mathematical objects specifically computer programs, so it makes perfect sense to ask whether two of them are equal.
- Identity type : According to the propositions-as-types conception, the proposition that two elements of the same type $a, b : A$ are equal must correspond to some type. So this identity type must be a type family Id_A dependent on two objects of the type A.
  $\prod_{(x:A)} \prod_{(y:A)} (x =_A y) : Id_A(x, y) : \mathcal{U}$
- $Id_A : A \to A \to \mathcal{U}$, so that $Id_A(a, b)$ is the type representing the proposition of equality between a and b.

Type theory

The type of unit $\mathbf{1}$

For any $x, y : \mathbf{1}$, we have $(x = y) \simeq \mathbf{1}$

The type of booleans

The type of booleans $\mathbf{2} : \mathcal{U}$ is intended to have exactly two elements $0_{\mathbf{2}}, 1_{\mathbf{2}} : \mathbf{2}$.

The type of Natural numbers $\mathbb{N} : \mathcal{U}$

We can construct them with
$0 : \mathbb{N}$ and the successor operation $succ : \mathbb{N} \to \mathbb{N}$.

Type theory

Propositions	Type Theory
True	$\mathbf{1}$
False	$\mathbf{0}$
A and B	$A \times B$
A or B	$A + B$
if A then B	$A \to B$
A iff B	$(A \to B) \times (B \to A)$
Not A	$A \to \mathbf{0}$
For all $x : A, P(x)$ holds	$\prod_{(x:A)} P(x)$
There exists $x : A$ s.t. P(x)	$\sum_{(x:A)} P(x)$

Table: Propositions as types

Homotopy theory - 2nd Building Block of HoTT

Topological space

A topological space (X, T) is a pair of a set X together with a collection of open subsets T that satisfies the four conditions:

1. The empty set $\emptyset \in T$.
2. $X \in T$.
3. The intersection of a finite number of sets in T is also in T.
4. The union of an arbitrary number of sets in T is also in T.

Path

A path in a topological space $\mathcal{X}$ is a continuous function $f : [0, 1] \to \mathcal{X}$. The initial point of the path is f(0) and the terminal point is f(1).

Homotopy theory

Homotopy *"Continuous deformation of f into g fixing endpoints"*

A homotopy between two continuous functions f and g from a top. space $\mathcal{X}$ to a top. space $\mathcal{Y}$ is defined to be a continuous function $\mathcal{H} : \mathcal{X} \times [0,1] \to \mathcal{Y}$ such that, if $x \in \mathcal{X}$, then $H(x,0) = f(x)$ and $H(x,1) = g(x)$.

$f|_A$ ::= restriction of function f from $\mathcal{D}om(f)$ to A.
$\mathcal{K}$ is compact if each of its open covers, say C and $\mathcal{K} = \cup_{x \in C} x$, has a finite subcover $F \subset C$ ($\mathcal{K} = \cup_{x \in F} x$).

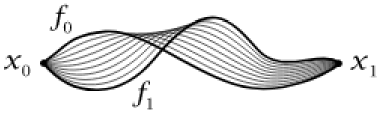


FIGURE 1. A picture of a homotopy between paths f_1 and f_2 from x_0 to x_1

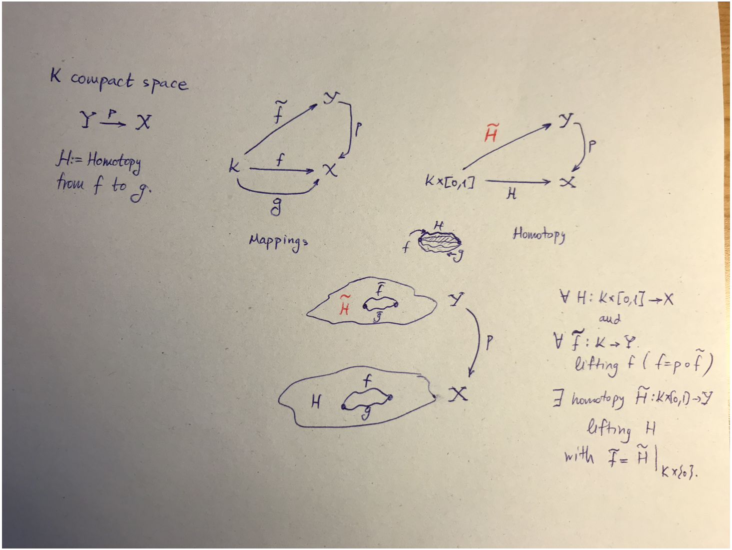
Homotopy lifting property

Homotopy lifting property

Given a projection map $p : Y \to X$ and a compact space $\mathcal{K}$, one says that $(\mathcal{K}, p)$ has the homotopy lifting property with respect to $\mathcal{K}$ if:

- for any homotopy $H : \mathcal{K} \times [0,1] \to X$ and,
- for any map $\bar{f} : \mathcal{K} \to Y$ lifting $f = H|_{\mathcal{K}\times\{0\}}$ (that is $f = p \circ \bar{f}$), there exists a homotopy $\bar{H} : \mathcal{K} \times [0,1] \to Y$ lifting H (that is $H = p \circ \bar{H}$) which also satisfies $\bar{f} = \bar{H}|_{\mathcal{K}\times\{0\}}$.

Homotopy theory



Homotopy theory

Loop - Loop Space

Loop : Paths from a point to itself (very interesting in Homotopy).
We define the loop space $\Omega(A, a)$ to be the type $a =_A a$.

We can then consider the loop space of the loop space of A, which is the space of 2-dim loops on the identity loop at a. This is $\Omega^2(A, a)$ and is represented in type theory by the type $refl_a =_{(a =_A a)} refl_a$.

Fibration

A fibration is a continuous mapping $p : Y \to X$ satisfying the homotopy lifting property with respect to any space $\mathcal{K}$.

Path space

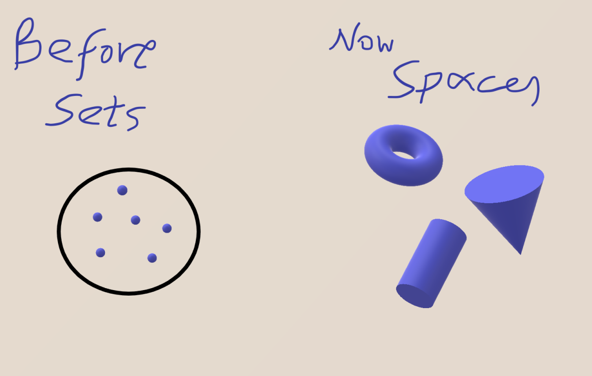
Let $I = [0, 1]$.Path space A^I is the space of all continuous maps $I \to A$ from the unit interval.

Homotopy type theory

Consider the model where :

- Types are (equivalence classes of) topological spaces (collections of points with a notion of continuity).
- Two points are equal if there is a continuous path between them.

1)Equality types are types, they have their own equality types, etc.
2)Paths between points are lines, paths between lines are surfaces, etc.



Homotopy type theory

Type Theory	Homotopy Theory
types A	spaces $\mathcal{A}$
terms a	points a
$a : A$	$a \in \mathcal{A}$
dependent type $x : A \vdash B(x)$	fibration $\mathcal{B} \to \mathcal{A}$
identity type $Id_A(a, b)$	path space
$p : Id_A(a, b)$	path $p : a \to b$
$H : Id_{Id_A(a,b)}(p, q)$	homotopy $H : p \Rightarrow q$

Table: Homotopy type theory correspondances.

Groupoids and the $\infty-$groupoid

In classical mathematics, the base point of set theory is the set. In HoTT, the base point is the $\infty - groupoid$.

Groupoid - Viewpoint 1

Group with a partial function replacing the binary operation.
$(G, *)$ where there may exist $a, b \in G$, with $a * b$ not always well defined.

Groupoid - Viewpoint 2

Category in which every morphism is invertible.

- Points (dim $= 0$), paths (dim $= 1$), paths of paths (dim $= 2$), ...
- Because homotopies are themselves a kind of 2-dimensional path, there is a natural notion of 3-dimensional homotopy between homotopies, and then homotopy between homotopies between homotopies, and so on.
- Smoothness of the $\infty-$gpd : We have to equip it with cohesion in the form of smooth structure (smooth manifolds, Lie groups).

$\infty-$groupoid

- This infinite stack of points, paths, homotopies, homotopies between homotopies, ..., equipped with algebraic operations such as the fundamental group, is an instance of an algebraic structure called a (weak) ∞-groupoid.

$\infty-$groupoid

- This infinite stack of points, paths, homotopies, homotopies between homotopies, ..., equipped with algebraic operations such as the fundamental group, is an instance of an algebraic structure called a (weak) ∞-groupoid.

Properties :

- Every topological space $\mathcal{X}$ has a fundamental ∞-groupoid whose k-morphisms are the k-dimensional paths in $\mathcal{X}$.
- The weakness of the ∞-groupoid corresponds directly to the fact that paths form a group only up to homotopy, with the (k+1)-paths serving as the homotopies between the k-paths.

$\infty-$groupoid and $\infty-$category

An $\infty-$groupoid (also $\omega-$groupoid) is an ∞-category in which all k-morphisms for all $k \in \mathbb{N}$ are equivalences (isomorphisms).

- An ∞-groupoid consists of a collection of objects, and then a collection of morphisms between objects, and then morphisms between morphisms,and so on, equipped with some complex algebraic structure; a morphism at level k is called a k-morphism.

An ∞-category is defined as follows :
In an ordinary category, one has morphisms going between objects.
In a 2-category, one has both 1-morphisms between objects and 2-morphisms going between 1-morphisms.
...
In an ∞-category, the objects are the 0-morphisms and there are k-morphisms going between $(k-1)$-morphisms for all $k \in \mathbb{N}$.

Types are higher groupoids

Equality	Homotopy	$\infty-$ Groupoid
reflexivity	constant path	identity morphism
symmetry	inversion of paths	inverse morphism
transitivity	concatenation of paths	composition of morphisms

Table: Homotopy type theory correspondances.

Types are higher groupoids

refl : $\prod_{a:A}(a =_A a)$

Paths can be reversed

For every type A and every $x, y : A$, there is a function $(x = y) \to (y = x)$, denoted $p \mapsto p^{-1}$, such that $refl_x^{-1} \equiv refl_x$ for each $x : A$.

This function is a type : $\prod_{(A:\mathcal{U})} \prod_{(x,y:A)}(x = y) \to (y = x)$.

Concatenation of paths

For every type A and every $x, y, z : A$, there is a function $(x = y) \to (y = z) \to (x = z)$ written $p \mapsto q \mapsto p * q$, such that $refl_x * refl_x \equiv refl_x$ for any $x : A$. We call $p * q$ the concatenation or composite of p and q.

Functions are functors (maps between categories).

Now we wish to establish that functions $f : A \to B$ behave functorially on paths. In traditional type theory, this is equivalently the statement that functions respect equality. Topologically, this corresponds to saying that every function is continuous, i.e. preserves paths.

Application of f to a path

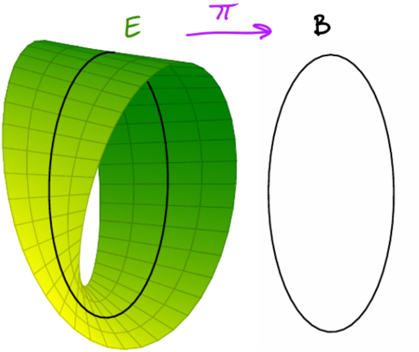
Suppose that $f : A \to B$ is a function. Then for any $x, y : A$ there is an operation $ap_f : (x =_A y) \to (f(x) =_B f(y))$. Moreover, for each $x : A$, we have $ap_f(refl_x) \equiv refl_{f(x)}$.

Type families are fibrations

What is a fibration? Fibration is a 'twisted' Cartesian product.

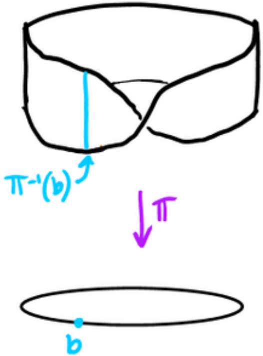
A total space, a base space

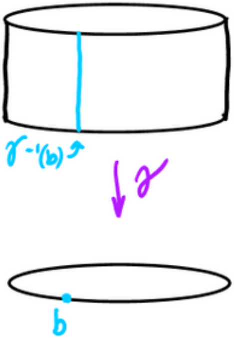
& a projection map.



Type families are fibrations

(Left) Total Space = Mobius band $\mathcal{M}$, Projection Space = Circle $\mathcal{C}$. Then the fiber is a vertical interval $\mathcal{I}$ (like a twisted Cartesian product). The fibration is $\mathcal{I} \to \mathcal{M} \to \mathcal{C}$.
(Right) Total Space = Simple Band. Then the fiber is a vertical interval (like a Cartesian product).





Type families are fibrations

1)Examples of Cart.Prods : $R \times S1 = Cylinder, S1 \times S1 = Torus.$
2)*Remark :* Fibration is a 'twisted' Cartesian product.
3)Understanding fibration via Cartesian product :
A cartesian product has a projection (to B): $(A \rightarrow)A \times B \rightarrow B$
So, for an element $b \in B, \exists$ space X s.t. $X \rightarrow b \in B$. and space X is of the form $A \times \{b\} \simeq A$ (fiber over b). A fibration is something that has a projection like this. For an element b in projection space B, if the subspace of the Total Space E is of the form $F \times \{b\}$, then F is the fiber over b. The fibration is the mapping $F \rightarrow E \rightarrow B$.

Transport Lemma

Suppose that P is a type family over A and that $p : x =_A y$. Then there is a function $p_* : P(x) \rightarrow P(y)$.

Homotopies and equivalences

So far, we have seen how the identity type $x =_A y$ can be regarded as a type of identifications, paths, or equivalences between two elements x and y of a type A. Let's recall the main question : When functions f, g are equal? Under the propositions-as-types interpretation, this suggests that two functions f and g (perhaps dependently typed) should be the same if the type $\prod_{(x:A)} (f(x) = g(x))$ is inhabited. Under the homotopical interpretation, this dependent function type consists of continuous paths or functorial equivalences, and thus may be regarded as the type of homotopies or of natural isomorphisms.

Let $f, g : \prod_{(x:A)} P(x)$ be two sections of a type family $P : A \to \mathcal{U}$. A homotopy from f to g is a dependent function of type $(f \sim g) :\equiv \prod_{(x:A)} (f(x) = g(x))$.

Note that a homotopy is not the same as an identification $(f = g)$. However, we will introduce an axiom making homotopies and identifications equivalent.

About homotopies

Reflexive, Symmetric, Transmittive

Homotopy is an equivalence relation on each function type $A \to B$. That is, we have elements of the types

$R)\ \prod_{f:A\to B}(f \sim f)$

$S)\ \prod_{f,g:A\to B}(f \sim g) \to (g \sim f)$

$T)\ \prod_{f,g,h:A\to B}(f \sim g) \to (g \sim h) \to (f \sim h).$

Identity type

Just as the type $a =_A a'$ is characterized up to isomorphism, with a separate definition for each A, there is no simple characterization of the type $p =_{a=_A a'} q$ of paths between paths $p, q : a =_A a'$

Theorem (Application of equivalence f to paths is equivalence)

If $f : A \to B$ is an equivalence, then for all $a, a' : A$, so is $ap_f : (a =_A a') \to (f(a) =_B f(a'))$.

Introduction to the Univalence Axiom

Basic Idea

In type theory, one can have a universe (it's a type) $\mathcal{U}$, the terms $\{A_i\}$ of which are themselves types, $A_i : \mathcal{U}$. (We do not have $\mathcal{U} : \mathcal{U}$, so only some types are terms of $\mathcal{U}$) These types are called the small types. Like any type, $\mathcal{U}$ has an identity type $Id_{\mathcal{U}}$, which expresses the identity relation $A =_{\mathcal{U}} B$ among small types. Thinking of types as spaces, $\mathcal{U}$ is a space, the points of which are spaces.

Main Question

To understand its identity type, we must ask, "What is a path $p : A \to B$ between spaces in $\mathcal{U}$?"

The Univalence Axiom says

Such paths correspond to homotopy equivalences $A \simeq B$

Introduction to the Univalence Axiom

A bit more precisely :

Given any small types A and B, in addition to the type $Id_{\mathcal{U}}(A, B)$ of identities between A and B, there is the type $Eq(A, B)$ of equivalences from A to B. Since the identity map on any object is an equivalence, there is a canonical map $Id_{\mathcal{U}}(A, B) \to Eq(A, B)$. The Univalence Axiom states that this map is itself an equivalence :
Univalence Axiom: $(A \simeq B) \simeq (A = B)$. In other words, equivalence is equivalent to identity (equality).

Equivalence is equivalent to equality.
Think of the boolean space.

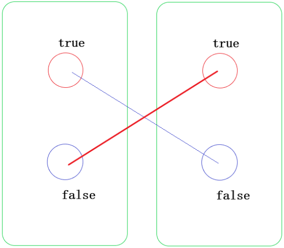


Figure: Vladimir Voevodsky (1966-2017)

Or think of the Finitely Generated Abelian Groups. We have that for $p, q \in \mathbb{Z}_{\geq 0}$ with $(p, q) = 1$, then $\mathbb{Z}_{pq} \cong \mathbb{Z}_p \times \mathbb{Z}_q$.

The univalence axiom

Given two types A and B, we may consider them as elements of some universe type $\mathcal{U}$, and thereby form the identity type $A =_{\mathcal{U}} B$. Univalence is the identification of $A =_{\mathcal{U}} B$ with the type $(A \simeq B)$ of equivalences from A to B.

idtoeqv

For types $A, B : \mathcal{U}$, there is a certain function :
idtoeqv : $(A =_{\mathcal{U}} B) \to (A \simeq B)$

Proof

Note that the identity function $id_{\mathcal{U}} : \mathcal{U} \to \mathcal{U}$ may be regarded as a type family indexed by the universe $\mathcal{U}$; it assigns to each type $X : \mathcal{U}$ the type X itself. Thus, given a path $p : A =_{\mathcal{U}} B$, we have a transport function $p_* : A \to B$. We claim that p_* is an equivalence. But by induction, it suffices to assume that p is $refl_A$, in which case $p_* \equiv id_A$, which is an equivalence. Thus, define $idtoeqv(p)$ to be p_*.

The univalence axiom

We would like to say that idtoeqv is an equivalence. the type theory described before is insufficient to guarantee this. We formulate this property as an axiom: Voevodskys univalence axiom.

Axiom

For types $A, B : \mathcal{U}$, the function idtoeqv : $(A =_{\mathcal{U}} B) \to (A \simeq B)$ is an equivalence.

Equivalent types may be identified.

The univalence axiom

It can be shown that corresponding statements looking like the classical law of double negation and law of excluded middle are incompatible with the univalence axiom.

Theorem

It is not the case that for all $A : \mathcal{U}$ we have $\neg(\neg A) \to A$.

Theorem

It is not the case that for all $A : \mathcal{U}$ we have $A + (\neg A)$.

The univalence axiom

Univalence axion from logical point of view...

... it is revolutionary: it says that isomorphic things can be identified! People are used to identify isomorphic structures in practice, but they generally do so knowing that the identification is not 'officially' justified by the foundations. But, working on this new foundational scheme, such structures are formally identified.

Inductive types

An inductive type X can be intuitively understood as a type freely generated by a certain finite collection of constructors, each of which is a function with codomain X.

- Type $\mathbb{N}$ of Naturals : $0 : \mathbb{N}, succ : \mathbb{N} \to \mathbb{N}$.
- Type $\mathbb{N}'$ of new Naturals : $0' : \mathbb{N}', succ' : \mathbb{N}' \to \mathbb{N}'$.
- Type $List(A)$ of finite lists of elements of some type A : $nil : List(A), cons : A \to List(A) \to List(A)$.

How to create an inductive type ? The elements of an inductive type are exactly what can be obtained by starting from nothing and applying the constructors repeatedly. In the case of $\mathbb{N}$, we should expect that every element is either 0 or obtained by applying succ to some previously constructed natural number.

$\mathbb{N}'$ has identical-looking induction and recursion principles to $\mathbb{N}$. But what is the relation between $\mathbb{N}$ and $\mathbb{N}'$?

Higher inductive types

Issue : How to construct homotopically non-trivial objects and spaces?
Use Higher(-dimensional) Inductive Types :
This is how you can generate a circle.

```
Circle : Type where
| base :: Circle
| loop :: Path base base
```
(start and end same point).

Higher inductive types

```
(*Higher inductive types list.*)

(*For instance, a circle S1 is generated by a point base
  and a dependent path loop.*)
Inductive circle : Type :=
| base : circle
| loop : base == base

Inductive Interval : Type :=
| left : Interval
| right : Interval
| segment : Id Interval left right

Inductive sphere2 : Type :=
| base2 : sphere2
| surf2 : idpath base2 == idpath base2

(*The suspension is the universal way to make points into paths.*)
Inductive susp (X : Type) : Type :=
| north : susp X
| south : susp X
| merid : X -> north == south.
```

Higher inductive types

Higher inductive types are a general schema for defining new types generated by some constructors.

- We can consider the higher inductive type $\mathbb{S}^1$ generated by a point $\mathit{base} : \mathbb{S}^1$ and a path $\mathit{loop} : \mathit{base} =_{\mathbb{S}^1} \mathit{base}$.
- This can be regarded as entirely analogous to $\mathbf{2}$ generated by a point $0_2 : \mathbf{2}$ and a point $1_2 : \mathbf{2}$ or the definition of $\mathbb{N}$ as generated by a point $0 : \mathbb{N}$ and a function $\mathit{succ} : \mathbb{N} \to \mathbb{N}$. ...

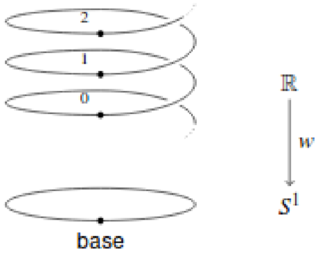
Example (How to use) : Prove that $\pi_1(\mathbb{S}^1) = \mathbb{Z}$.

- Let A be a HIT (f.e. Circle) and say we want to describe a type family $P : A \to \mathcal{U}$. Then for the point constructors (base), we have to provide types and for path constructors (loop), we have to provide paths between these types. Thanks to Univalence, it suffices to give equivalences between these types.
- Define the map from Circle to a Type $C : \mathbb{S}^1 \to \mathcal{U}$, where : $C(base) = \mathbb{Z}$ and to transport along loop we apply the equivalence $succ : \mathbb{Z} \to \mathbb{Z}$.
- We will use C to show that $(base =_{\mathbb{S}^1} base) \simeq \mathbb{Z}$

Assuming univalence, the loop space of Circle is homotopy-equivalent to the type Int.

Higher inductive types

Let $w : \mathbb{R} \to \mathbb{S}^1$ be the 'winding' map, projecting the helix to the circle. This map is a fibration (i.e. a dependent type) and the fiber over each point is isomorphic to $\mathbb{Z}$.
By walking around the loop, we either go up or down to the helix according to the direction of our walk.
As a result of univalence, the fibers of two fibrations are homotopy equivalent. Thus, since $\mathbb{R}$ and the helix $P_{base}\mathbb{S}^1$ are both contractible (that can be shrunk to a point), they are homotopy equivalent and thus the



fibers over base are isomorphic.

Homotopy theory (Topology)

One question homotopy theorists investigate is how to tell whether two spaces are the same, where the same means homotopy equivalence (continuous maps back and forth that compose to the identity up to homotopy).

Homotopy theory (Topology)

One question homotopy theorists investigate is how to tell whether two spaces are the same, where the same means homotopy equivalence (continuous maps back and forth that compose to the identity up to homotopy).

One common way to tell whether two spaces are the same is to calculate algebraic invariants associated with a space, which include its homotopy groups and homology and cohomology groups.

Homotopy theory (Topology)

One question homotopy theorists investigate is how to tell whether two spaces are the same, where the same means homotopy equivalence (continuous maps back and forth that compose to the identity up to homotopy).
One common way to tell whether two spaces are the same is to calculate algebraic invariants associated with a space, which include its homotopy groups and homology and cohomology groups.
We will see in short the technique of the fundamental group of a space.

Fundamental group

The simplest example of a homotopy group is the fundamental group of a space, which is written $\pi_1(X, x_0)$.

Fundamental group

The simplest example of a homotopy group is the fundamental group of a space, which is written $\pi_1(X, x_0)$.
Informally : The fundamental group is a mathematical group associated to any given pointed topological space that provides a way to determine when two paths, starting and ending at a fixed base point, can be continuously deformed into each other. It records information about the basic shape, or holes, of the topological space.

Fundamental group

The simplest example of a homotopy group is the fundamental group of a space, which is written $\pi_1(X, x_0)$.
Informally : The fundamental group is a mathematical group associated to any given pointed topological space that provides a way to determine when two paths, starting and ending at a fixed base point, can be continuously deformed into each other. It records information about the basic shape, or holes, of the topological space.
A path-connected topological space is simply connected iff its fundamental group is trivial.

Fundamental groups

Let $\mathcal{X}$ be a topological space, and let x_0 be a point of X.
We study the following set $\mathcal{L}$ of continuous functions called loops with base point x_0, that is :
$\mathcal{L} = \{f : [0,1] \to \mathcal{X} : f(0) = x_0 = f(1)\}$.

Fundamental groups

Let $\mathcal{X}$ be a topological space, and let x_0 be a point of X.
We study the following set $\mathcal{L}$ of continuous functions called loops with base point x_0, that is :
$\mathcal{L} = \{f : [0,1] \to \mathcal{X} : f(0) = x_0 = f(1)\}$.
Now we create the fundamental group of $\mathcal{X}$ with base point x_0 is this set $\mathcal{L}$ modulo homotopy h :
$\pi_1(\mathcal{X}, x_0) = \mathcal{L}/h$. (quotient group)

Fundamental groups

Let $\mathcal{X}$ be a topological space, and let x_0 be a point of X.
We study the following set $\mathcal{L}$ of continuous functions called loops with base point x_0, that is :
$\mathcal{L} = \{f : [0,1] \to \mathcal{X} : f(0) = x_0 = f(1)\}$.
Now we create the fundamental group of $\mathcal{X}$ with base point x_0 is this set $\mathcal{L}$ modulo homotopy h :
$\pi_1(\mathcal{X}, x_0) = \mathcal{L}/h$. (quotient group)
To make this set a group, we have to equip it with the group multiplication defined by :

$$(f * g)(t) = \begin{cases} f(2t), 0 \leq t \leq 1/2 \\ g(2t-1), 1/2 \leq t \leq 1 \end{cases}$$

Fundamental groups

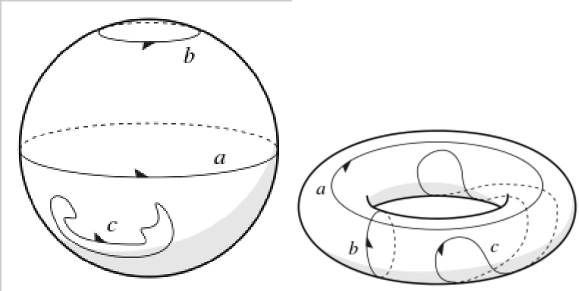
Let $\mathcal{X}$ be a topological space, and let x_0 be a point of X.
We study the following set $\mathcal{L}$ of continuous functions called loops with base point x_0, that is :
$\mathcal{L} = \{f : [0,1] \to \mathcal{X} : f(0) = x_0 = f(1)\}$.
Now we create the fundamental group of $\mathcal{X}$ with base point x_0 is this set $\mathcal{L}$ modulo homotopy h :
$\pi_1(\mathcal{X}, x_0) = \mathcal{L}/h$. (quotient group)
To make this set a group, we have to equip it with the group multiplication defined by :

$$(f * g)(t) = \begin{cases} f(2t), 0 \leq t \leq 1/2 \\ g(2t-1), 1/2 \leq t \leq 1 \end{cases}$$

So the set of all homotopy classes of loops with base point x_0 forms the fundamental group of $\mathcal{X}$ at the point x_0.

Fundamental group : Examples

- Sphere $\mathbb{S}_2$ and the torus $\mathbb{T}^2$ are not homotopy equivalent :
  The fundamental group of the 2-sphere is trivial ($\pi_1(\mathbb{S}_2) = \{h\}$) (using Jordan curve theorem), but the fundamental group of the torus has cardinality > 1 (not trivial).
  The intuition is that every loop on the sphere is homotopic to the identity, because its inside can be filled in. In contrast, a loop on the torus (see a,b) is not homotopic to the identity, so there are non-trivial elements in the fundamental group.



Fundamental group : Examples

- In $\mathbb{R}_n$ or any convex subset of $\mathbb{R}_n$, there is only one homotopy class of loops, and the fundamental group is therefore the trivial group with one element.

Fundamental group : Examples

- In $\mathbb{R}_n$ or any convex subset of $\mathbb{R}_n$, there is only one homotopy class of loops, and the fundamental group is therefore the trivial group with one element.
- For the infinite cyclic fundamental group : Each homotopy class consists of all loops which wind around the circle a given number of times (which can be positive or negative, depending on the direction of winding). So $\pi_1(\mathbb{S}_1) \simeq (\mathbb{Z}, +)$. (You can use that to prove Brouwer fixed-point theorem on dim 2).

Homotopy theory (Topology)

The first and simplest homotopy group is the fundamental group, that we already saw.

Homotopy theory (Topology)

The first and simplest homotopy group is the fundamental group, that we already saw.
To define the n-th homotopy group, the base-point-preserving maps from an n-dimensional sphere (with base point) into a given space (with base point) are collected into equivalence classes, called homotopy classes. These homotopy classes form a group, called the n-th homotopy group, $\pi_n(\mathcal{X})$, of the given space $\mathcal{X}$ with base point x_0.

Homotopy theory (Topology)

The first and simplest homotopy group is the fundamental group, that we already saw.

To define the n-th homotopy group, the base-point-preserving maps from an n-dimensional sphere (with base point) into a given space (with base point) are collected into equivalence classes, called homotopy classes. These homotopy classes form a group, called the n-th homotopy group, $\pi_n(\mathcal{X})$, of the given space $\mathcal{X}$ with base point x_0.

Why? The higher homotopy groups provide additional information about a space.

Homotopy theory (Topology)

The first and simplest homotopy group is the fundamental group, that we already saw.

To define the n-th homotopy group, the base-point-preserving maps from an n-dimensional sphere (with base point) into a given space (with base point) are collected into equivalence classes, called homotopy classes. These homotopy classes form a group, called the n-th homotopy group, $\pi_n(\mathcal{X})$, of the given space $\mathcal{X}$ with base point x_0.

Why? The higher homotopy groups provide additional information about a space.

Informally : Fix a point x_0 in X, and consider the constant path $refl_{x_0}$. Then the homotopy classes of homotopies between $refl_{x_0}$ and itself form a group $\pi_2(X, x_0)$, which tells us something about the two-dimensional structure of the space. Then $\pi_3(X, x_0)$ is the group of homotopy classes of homotopies between homotopies, and so on.

n-th homotopy group

In the $n-$sphere $\mathbb{S}^n$ we choose a base point a. For a space $\mathcal{X}$ with base point b, we define $\pi_n(\mathcal{X})$ to be the set of homotopy classes of maps $f : \mathbb{S}^n \to \mathcal{X}$ that map the base point a to the base point b.

Homotopy groups of spheres

	S^0	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8
π_1	0	$\mathbb{Z}$	0	0	0	0	0	0	0
π_2	0	0	$\mathbb{Z}$	0	0	0	0	0	0
π_3	0	0	$\mathbb{Z}$	$\mathbb{Z}$	0	0	0	0	0
π_4	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0	0	0
π_5	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0	0
π_6	0	0	$\mathbb{Z}_{12}$	$\mathbb{Z}_{12}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	0
π_7	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}\times\mathbb{Z}_{12}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$	0
π_8	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_2^2$	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}$
π_9	0	0	$\mathbb{Z}_3$	$\mathbb{Z}_3$	$\mathbb{Z}_2^2$	$\mathbb{Z}_2$	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$	$\mathbb{Z}_2$
π_{10}	0	0	$\mathbb{Z}_{15}$	$\mathbb{Z}_{15}$	$\mathbb{Z}_{24}\times\mathbb{Z}_3$	$\mathbb{Z}_2$	0	$\mathbb{Z}_{24}$	$\mathbb{Z}_2$
π_{11}	0	0	$\mathbb{Z}_2$	$\mathbb{Z}_2$	$\mathbb{Z}_{15}$	$\mathbb{Z}_2$	$\mathbb{Z}$	0	$\mathbb{Z}_{24}$
π_{12}	0	0	$\mathbb{Z}_2^2$	$\mathbb{Z}_2^2$	$\mathbb{Z}_2$	$\mathbb{Z}_{30}$	$\mathbb{Z}_2$	0	0
π_{13}	0	0	$\mathbb{Z}_{12}\times\mathbb{Z}_2$	$\mathbb{Z}_{12}\times\mathbb{Z}_2$	$\mathbb{Z}_2^3$	$\mathbb{Z}_2$	$\mathbb{Z}_{60}$	$\mathbb{Z}_2$	0

Table 8.1: Homotopy groups of spheres [Wik13]. The k^{th} homotopy group π_k of the n-dimensional sphere S^n is isomorphic to the group listed in each entry, where $\mathbb{Z}$ is the additive group of integers, and $\mathbb{Z}_m$ is the cyclic group of order m.

Homotopy groups

For $n : \mathbb{N}$, the $n-$fold iterated loop space of a pointed type (A, a) is defined recursively by:
$\Omega^0(A, a) = (A, a)$
$\Omega^{n+1}(A, a) = \Omega^n(\Omega(A, a))$
This gives a space (a type) of $n-$dimensional loops, which itself has higher homotopies.

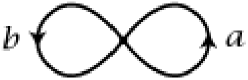
Homotopy Groups

Given $n \geq 1$ and (A, a) a pointed type, we define the homotopy groups of A at a by $\pi_n(A, a) :\equiv ||\Omega^n(A, a)||_0$

Properties for homotopy groups $\pi_n(A, a)$

Properties for $n = 1$. ([Fundamental group)

- Recall the fibration $F \to E \to B$. Then when all the spaces are connected, this has the following consequences for the fundamental groups: $\pi_1(B)$ and $\pi_1(E)$ are isomorphic if F is simply connected (that is path-connected and every path between two points can be continuously transformed into any other such path while preserving the two endpoints in question).
- Van Kampen's Theorem : Space decomposition into simpler spaces. Let's see this theorem using an example. Let $(A, *)$ and $(B, *)$ be two free groups generated by a and b respectively (equiv. the grammar $\{a^n\}$ and $((a^4 b^{-1}) * (b^3) = a^4 b^2)$). We know that $\pi_1(A) \simeq \mathbb{Z}$. Similarly, for B. Then consider the space *Infty*. The theorem implies that $\pi_1(Infty) \simeq \pi_1(A) * \pi_1(B) \simeq \mathbb{Z} * \mathbb{Z}$.



Properties for homotopy groups $\pi_n(\mathbb{S}_n, a)$

- Case $i < n$: $\pi_i(\mathbb{S}_n) = 0(= \{h\} =$trivial group$)$.
- Case $i = n$: $\pi_n(\mathbb{S}_n) = H_n(\mathbb{S}_n) = \mathbb{Z}$.

Theorem (Hurewicz theorem)

For any space $\mathcal{X}$ and $i \in \mathbb{N}$, there exists a group homomorphism $h_* : \pi_i(\mathcal{X}) \to H_i(\mathcal{X})$

Remark $i = 1$

The Hurewicz homomorphism for $i = 1$ and for $\mathcal{X}$ is path-connected is equivalent to the canonical abelianization map (commutator subgroup $G/[G, G]$).

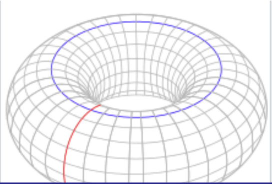
where $H_i(\mathcal{X})$ is the i-th homology group.

Properties for homotopy groups $\pi_n(\mathbb{S}_n, a)$

Homology group

Let $\mathcal{X}$ a topological space and fix $k \in \mathbb{N}_0$. Then, the k-th homology group $H_k(\mathcal{X})$ describes the k-th dimensional holes in $\mathcal{X}$. $H_0(\mathcal{X})$ describes the path-connected components of the space.

- $H_k(\mathbb{S}_1) = \begin{cases} \mathbb{Z}, k = 0, 1 \\ \{0\}, k > 1 \end{cases}$ $\quad H_k(\mathbb{S}_n) = \begin{cases} \mathbb{Z}, k = 0, n \\ \{0\}, \mathit{otherwise} \end{cases}$
- $H_k(\mathbb{T}^2) = \begin{cases} \mathbb{Z}, k = 0, 2 \\ \mathbb{Z} \times \mathbb{Z}, k = 1 \\ \{0\}, \mathit{otherwise} \end{cases}$



Properties for homotopy groups $\pi_n(\mathbb{S}_n, a)$

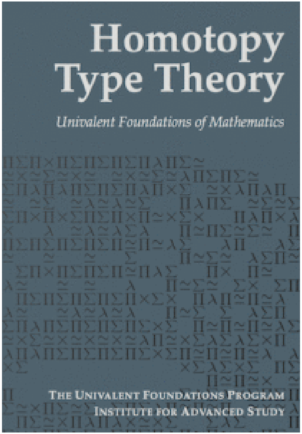
Construction of homology groups

Chain complex of abelian groups $C(\mathcal{X}) = \{C_i\}_{i \geq 0}$ and boundary operators $\partial_n : C_n \to C_{n-1}$ with $\partial_n \circ \partial_{n+1} = 0_{n+1,n-1}$, then we can create the quotient group $H_n(\mathcal{X}) = Ker(\partial_n)/Im(\partial_{n+1})$. ($Ker(\partial_n)$ is abelian and $Im(\partial_{n+1})$ is normal subgroup of the kernel).

- Case $i > n$: The homology groups $H_i(\mathbb{S}_n)$ are all trivial. The corresponding homotopy groups are not trivial. They are suprisingly complex and difficult to compute.

References

- *Homotopy Type Theory: Univalent Foundations of Mathematics*, The Univalent Foundations Program Institute for Advanced Study



Mathematics is security. Certainty. Truth. Beauty. Insight. Structure. Architecture. I see mathematics, the part of human knowledge that I call mathematics, as one thing-one great, glorious thing. Whether it is differential topology, or functional analysis, or homological algebra, it is all one thing. ... They are intimately interconnected, they are all facets of the same thing. That interconnection, that architecture, is secure truth and is beauty. That's what mathematics is to me.

Paul Halmos

$\mathcal{T}he\ \mathcal{E}nd$