Eilenberg-MacLane Spaces in Homotopy Type Theory

Dan Licata

Wesleyan University

Eric Finster

Inria Paris Rocquencourt

Constructive Type Theory

Three senses of constructivity:

Constructive Type Theory

Three senses of constructivity:

- Non-affirmation of certain classical principles provides **axiomatic freedom**

Synthetic geometry

Euclid's postulates

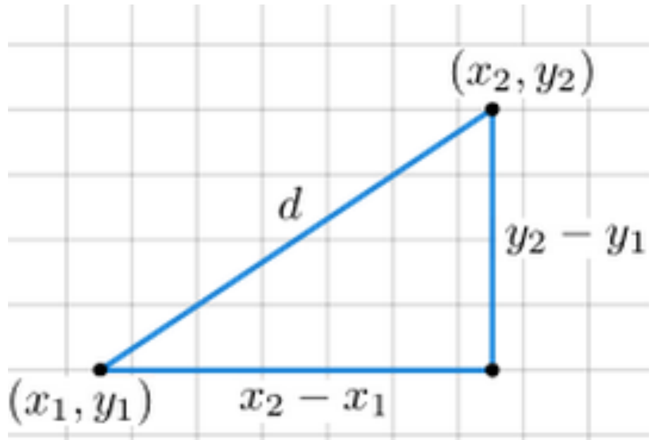
1. To draw a straight line from any point to any point.
2. To produce a finite straight line continuously in a straight line.
3. To describe a circle with any center and distance.
4. That all right angles are equal to one another.
5. Given a line and a point not on it, there is exactly one line through the point that does not intersect the line

Synthetic geometry

Euclid's postulates

1. **To draw a straight line from any point to any point.**
2. **To produce a finite straight line continuously in a straight line.**
3. **To describe a circle with any center and distance.**
4. **That all right angles are equal to one another.**
5. **Given a line and a point not on it, there is exactly one line through the point that does not intersect the line**

Cartesian



Synthetic geometry

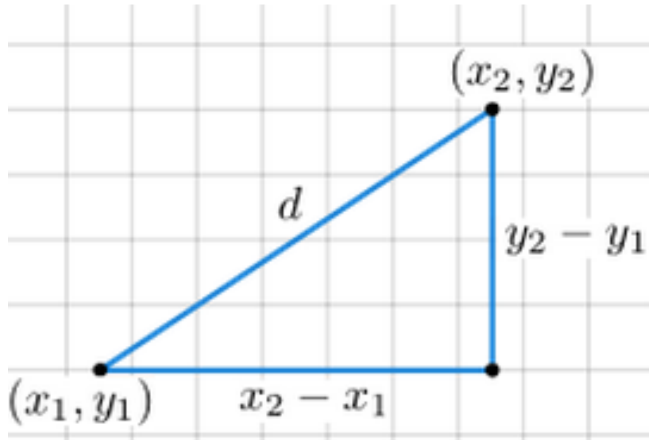
Euclid's postulates

1. To draw a straight line from any point to any point.
2. To produce a finite straight line continuously in a straight line.
3. To describe a circle with any center and distance.
4. That all right angles are equal to one another.
5. Given a line and a point not on it, there is exactly one line through the point that does not intersect the line

models

←

Cartesian



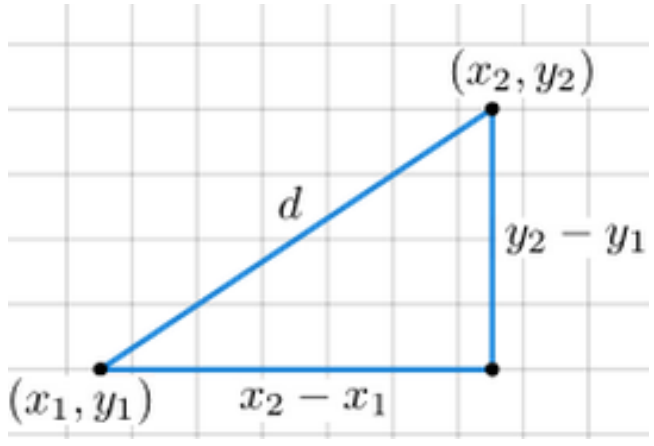
Synthetic geometry

Euclid's postulates

1. **To draw a straight line from any point to any point.**
2. **To produce a finite straight line continuously in a straight line.**
3. **To describe a circle with any center and distance.**
4. **That all right angles are equal to one another.**

models ←

Cartesian



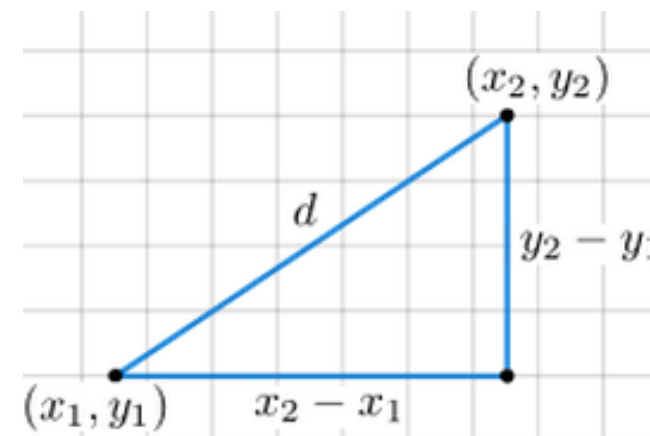
Synthetic geometry

Euclid's postulates

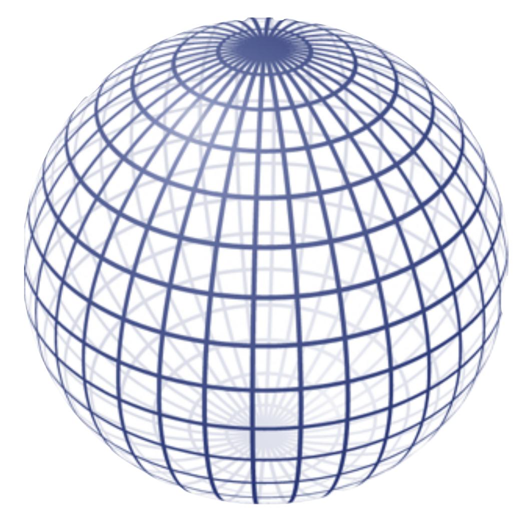
1. To draw a straight line from any point to any point.
2. To produce a finite straight line continuously in a straight line.
3. To describe a circle with any center and distance.
4. That all right angles are equal to one another.

models

Cartesian



Spherical



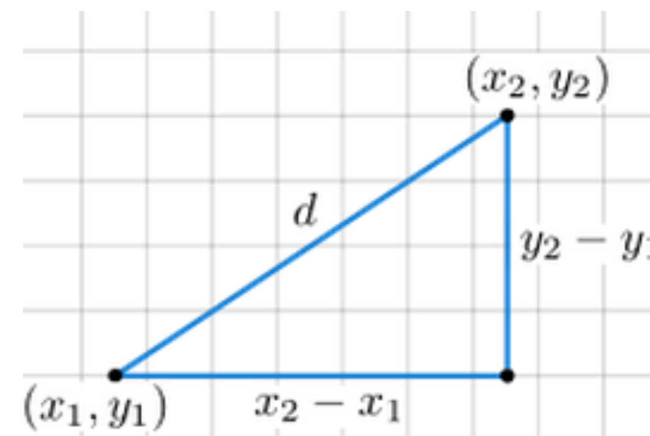
Synthetic geometry

Euclid's postulates

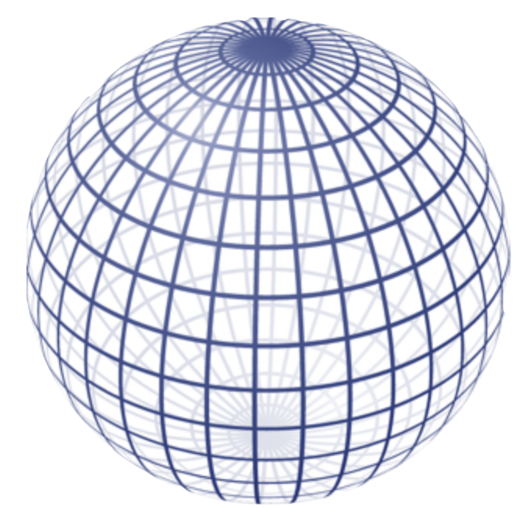
1. To draw a straight line from any point to any point.
2. To produce a finite straight line continuously in a straight line.
3. To describe a circle with any center and distance.
4. That all right angles are equal to one another.
5. Two distinct lines meet at two antipodal points.

models

Cartesian



Spherical



Synthetic mathematics

Type theory

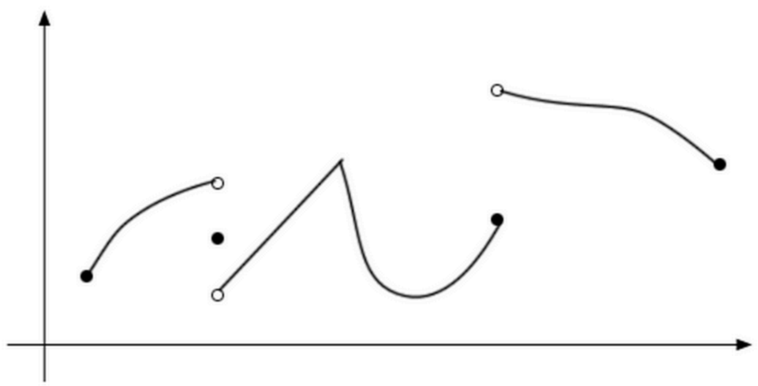
1. $\tau ::= b \mid \tau_1 \to \tau_2$
2. $e ::= x \mid e_1\, e_2 \mid \lambda x.e$
3. $(\lambda x.e)e_2 = e[e_2/x]$

Synthetic mathematics

Set-theoretic functions

Type theory

1. $\tau ::= b \mid \tau_1 \to \tau_2$
2. $e ::= x \mid e_1\, e_2 \mid \lambda x.e$
3. $(\lambda x.e)e_2 = e[e_2/x]$

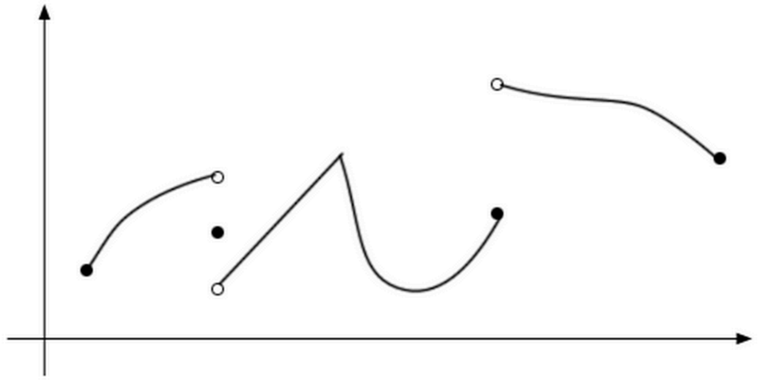


Synthetic mathematics

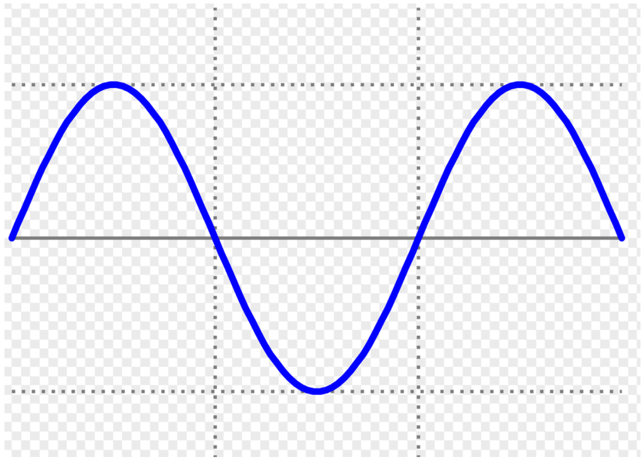
Type theory

1. $\tau ::= b \mid \tau_1 \to \tau_2$
2. $e ::= x \mid e_1\ e_2 \mid \lambda x.e$
3. $(\lambda x.e)e_2 = e[e_2/x]$

Set-theoretic functions

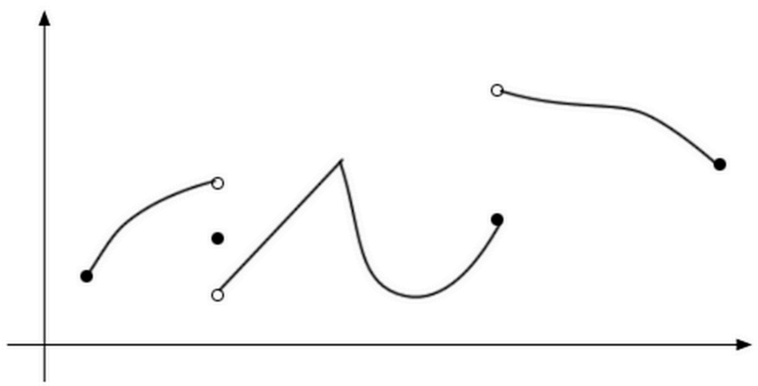


Domain-theoretic functions



Synthetic mathematics

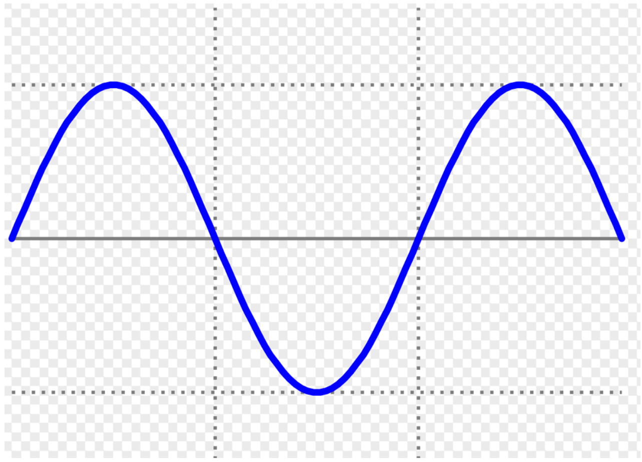
Set-theoretic functions



Type theory

1. $\tau ::= b \mid \tau_1 \to \tau_2$
2. $e ::= x \mid e_1\ e_2 \mid \lambda x.e$
3. $(\lambda x.e)e_2 = e[e_2/x]$
4. $Y(f) = f(Y(f))$

Domain-theoretic functions



Constructive Type Theory

Three senses of constructivity:

Constructive Type Theory

Three senses of constructivity:

- Non-affirmation of certain classical principles provides **axiomatic freedom**

Constructive Type Theory

Three senses of constructivity:

- Non-affirmation of certain classical principles provides **axiomatic freedom**
- **Computational interpretation** supports software verification and proof automation

Computational Interpretation

There is an algorithm that,
given a closed term `e : bool`,
computes either
an equality `e = true`, or
an equality `e = false`.

Constructive Type Theory

Three senses of constructivity:

- Non-affirmation of certain classical principles provides **axiomatic freedom**
- **Computational interpretation** supports software verification and proof automation

Constructive Type Theory

Three senses of constructivity:

- Non-affirmation of certain classical principles provides **axiomatic freedom**
- **Computational interpretation** supports software verification and proof automation
- Allows **proof-relevant mathematics**

Proof relevance

```
x : A
```

Proof relevance

$x : A$

$x =_A y$ equality type

Proof relevance

$x : A$

$p : x =_A y$ equality type

Proof relevance

`x : A`

`p : x =`$_A$` y` equality type

Any structure or property `C` can be transported along an equality

Proof relevance

$x : A$

$p : x =_A y$ equality type

Any structure or property C can be transported along an equality

$$\mathrm{transport}_C(p) : C(x) \to C(y)$$

Proof relevance

$x : A$

$p : x =_A y$ equality type

Any structure or property C can be transported along an equality

Leibniz's indiscernability of identicals

$transport_C(p) : C(x) \to C(y)$

Proof relevance

$x : A$

$p : x =_A y$ equality type

Any structure or property C can be transported along an equality ← **Leibniz's indiscernability of identicals**

$\mathrm{transport}_C(p) : C(x) \to C(y)$

by a function: can it do real work?

Proof relevance

$x : A$

$p : x =_A y$ equality type

Proof relevance

$x : A$

$p : x =_A y$ equality type

$p_1 =_{x=y} p_2$

Proof relevance

$x : A$

$p : x =_A y$ equality type

$q : p_1 =_{x=y} p_2$

Proof relevance

$x : A$

$p : x =_A y$ equality type

$q : p_1 =_{x=y} p_2$

$q_1 =_{p1=p2} q_2$

Proof relevance

$x : A$

$p : x =_A y$ equality type

$q : p_1 =_{x=y} p_2$

$r : q_1 =_{p1=p2} q_2$

Proof relevance

$x : A$

$p : x =_A y$ equality type

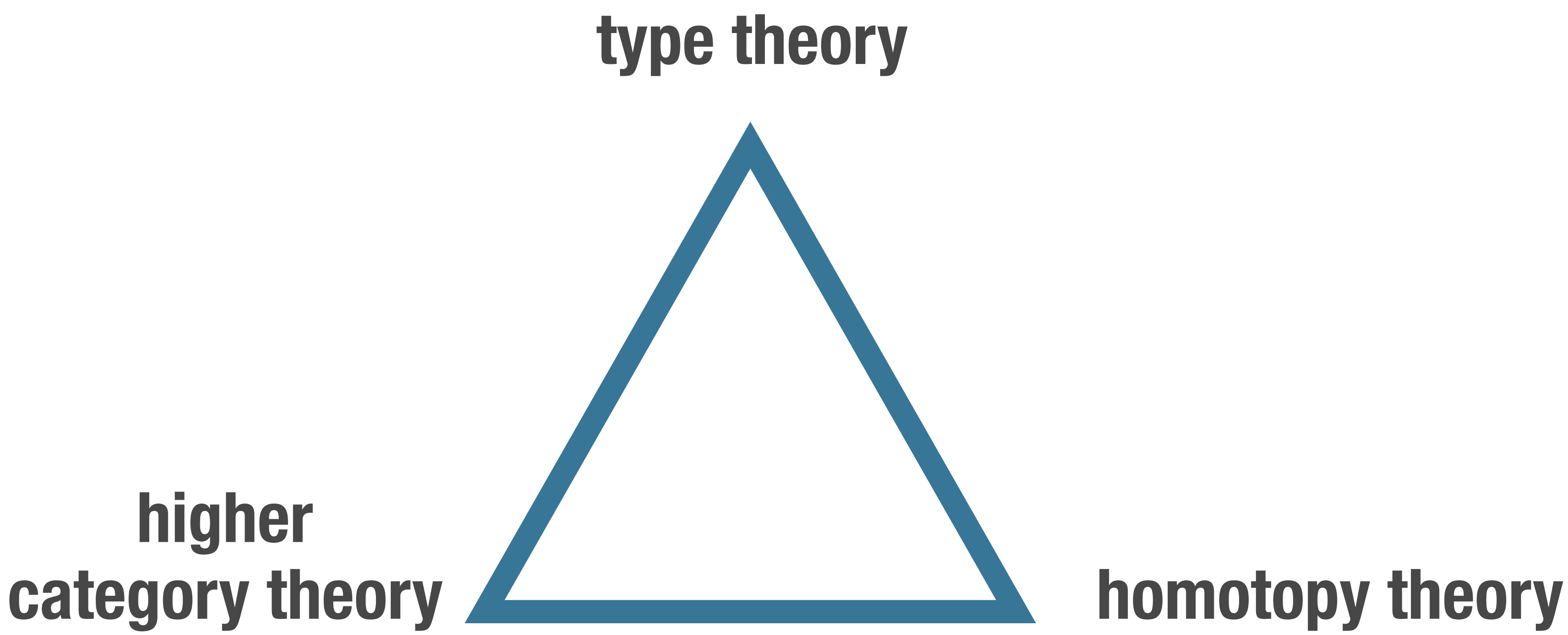
$q : p_1 =_{x=y} p_2$

$r : q_1 =_{p1=p2} q_2$

$\vdots$

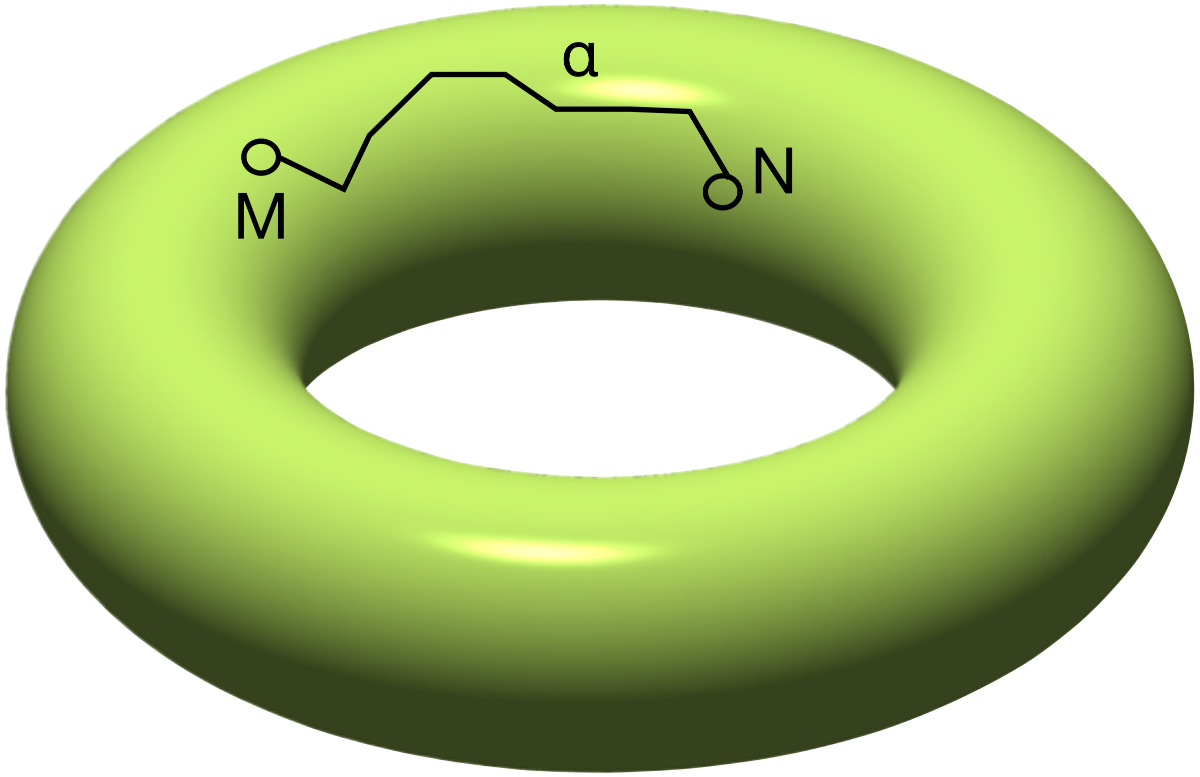
higher equalities radically expand the kind of math that can be done synthetically…

Homotopy Type Theory



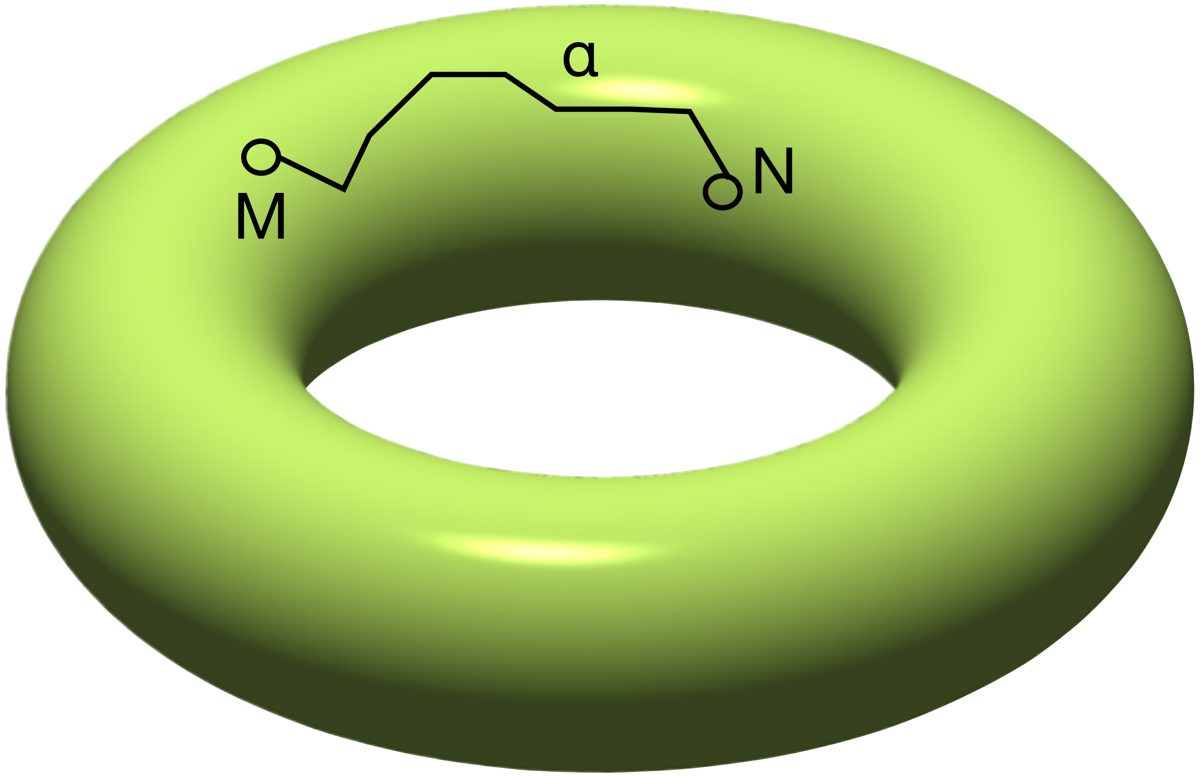
[Awodey,Warren,Voevodsky,,Streicher,Hofmann
Lumsdaine,Gambino,Garner,van den Berg]

Types as spaces



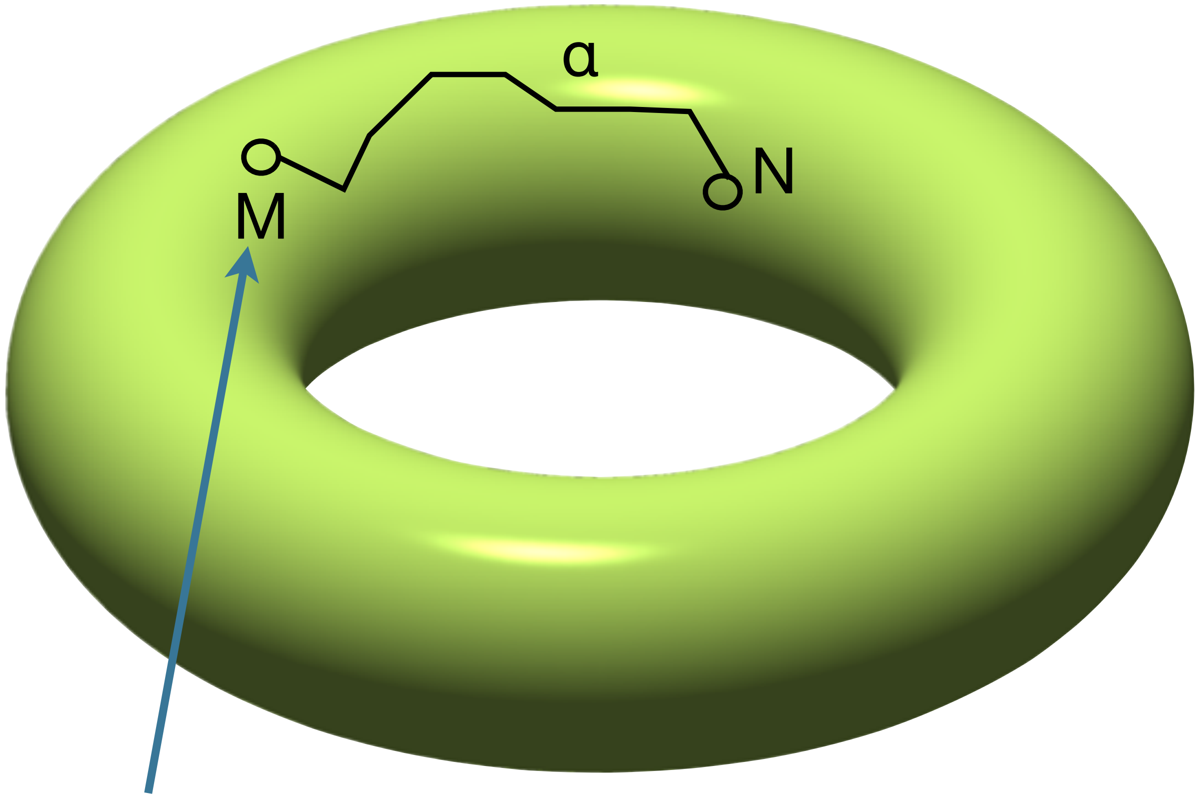
Types as spaces

type A is a space



Types as spaces

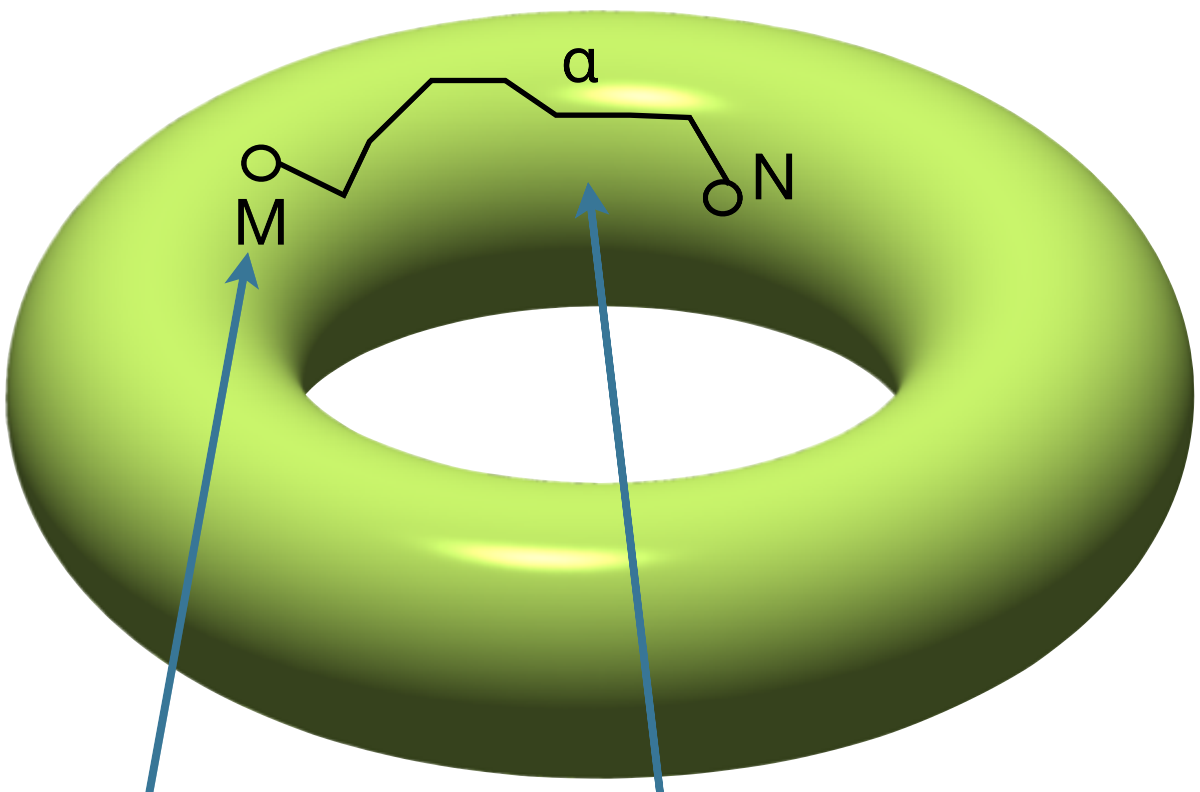
type A is a space



programs
M : A
are points

Types as spaces

type A **is a space**

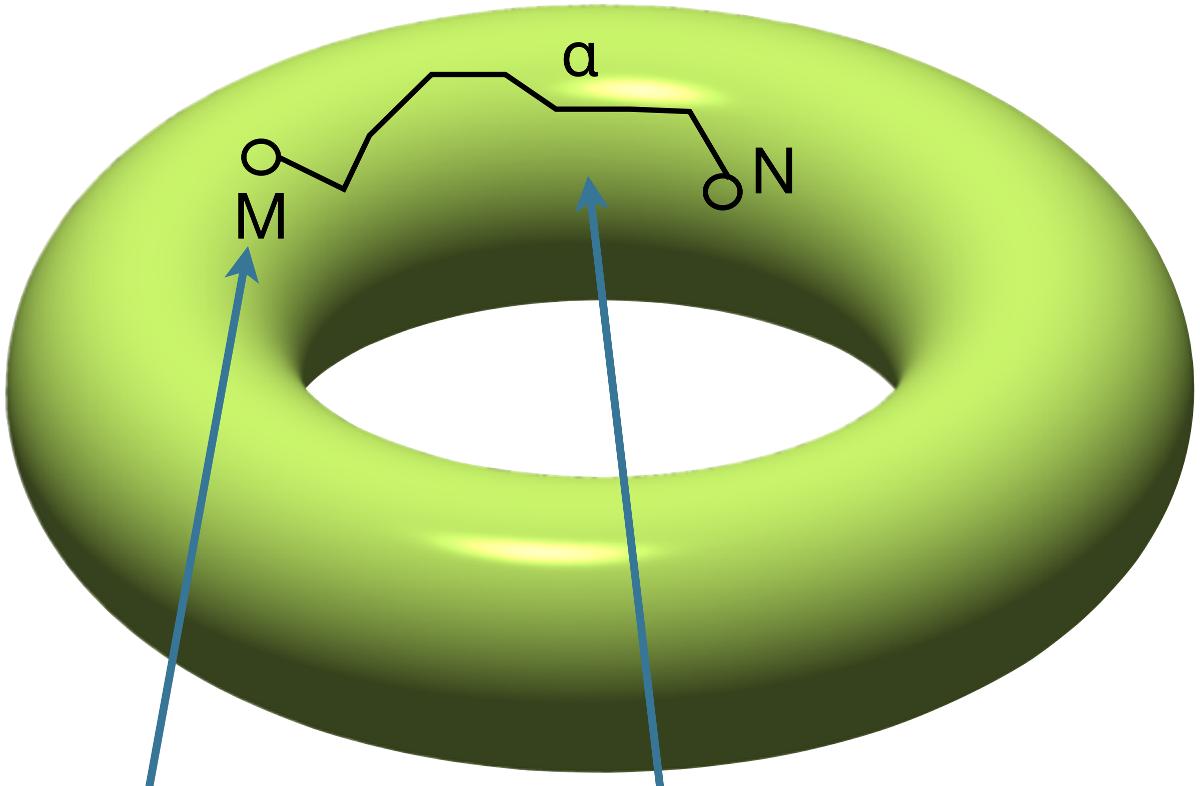


programs
$M : A$
are points

proofs of equality
$\alpha \ : \ M =_A N$
are paths

Types as spaces

type A is a space



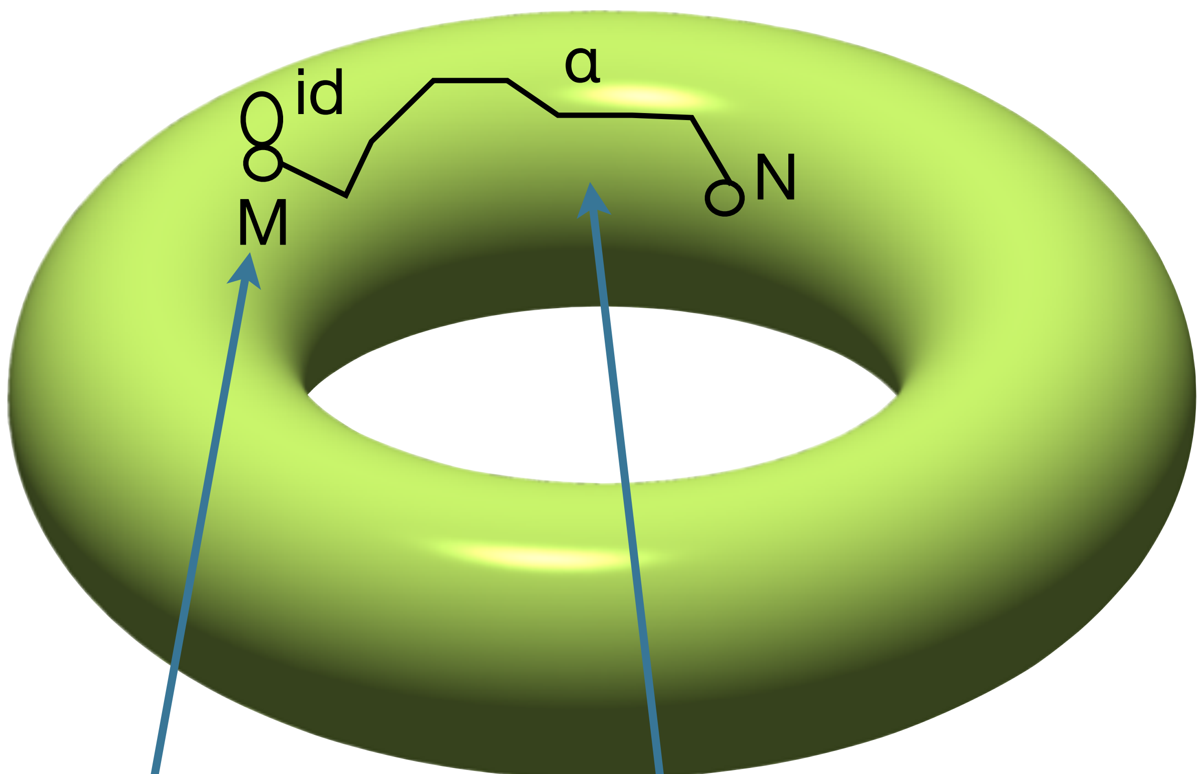
programs
M : A
are points

proofs of equality
$\alpha : M =_A N$
are paths

path operations

Types as spaces

type A is a space



programs

$M : A$

are points

proofs of equality

$\alpha \ : \ M =_A N$

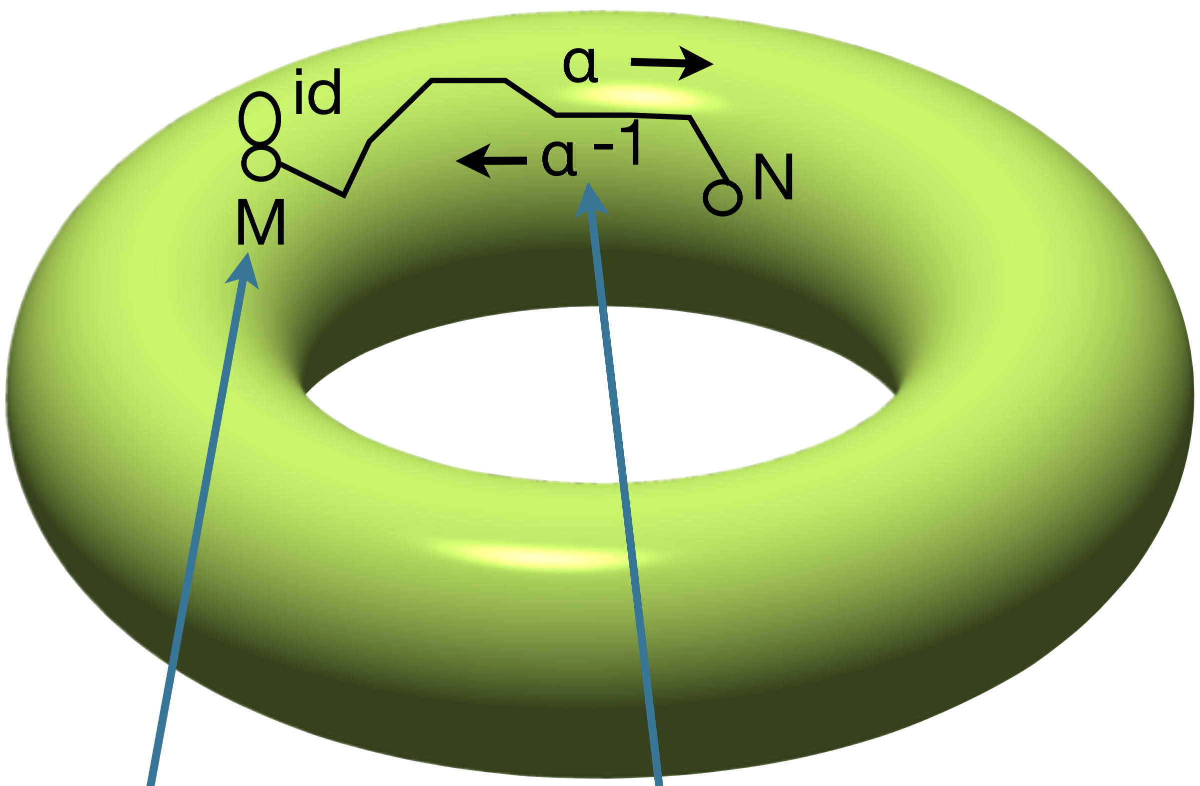
are paths

path operations

```
id      : M = M (refl)
```

Types as spaces

type A is a space



programs
M : A
are points

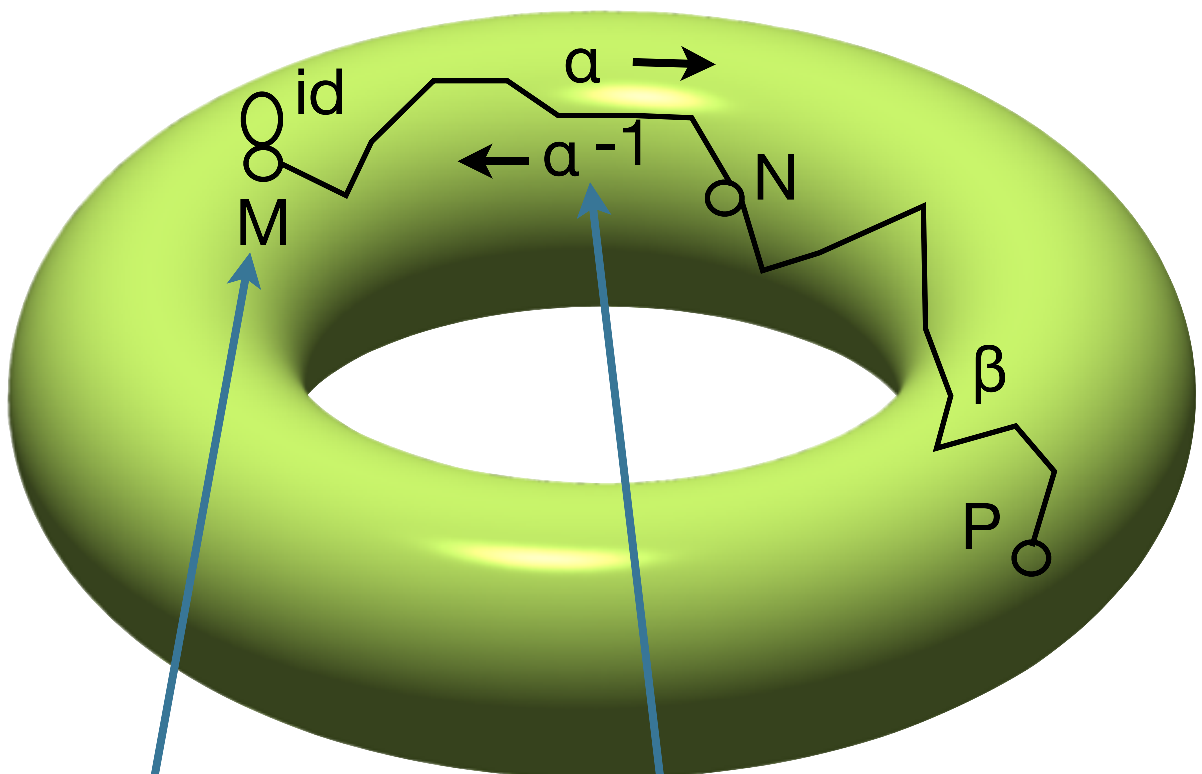
proofs of equality
$\alpha : M =_A N$
are paths

path operations

```
id      : M = M (refl)
α⁻¹     : N = M (sym)
```

Types as spaces

type A is a space



programs
M : A
are points

proofs of equality
$\alpha : M =_A N$
are paths

path operations

```
id     : M = M (refl)
α⁻¹    : N = M (sym)
β o α  : M = P (trans)
```

Homotopy

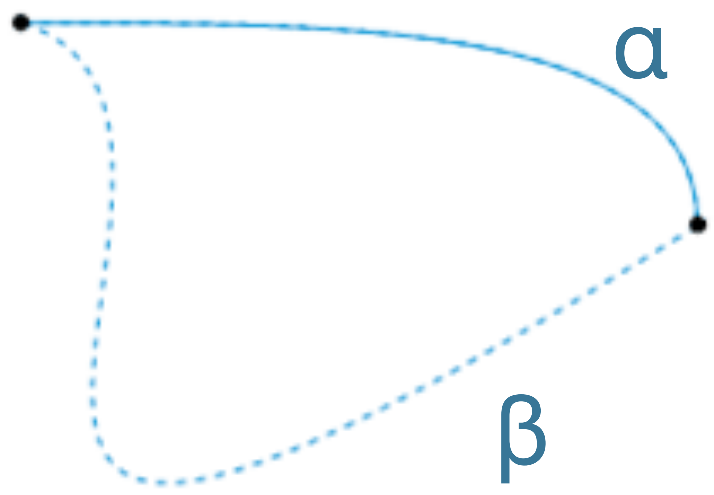
Deformation of one path into another

α

β

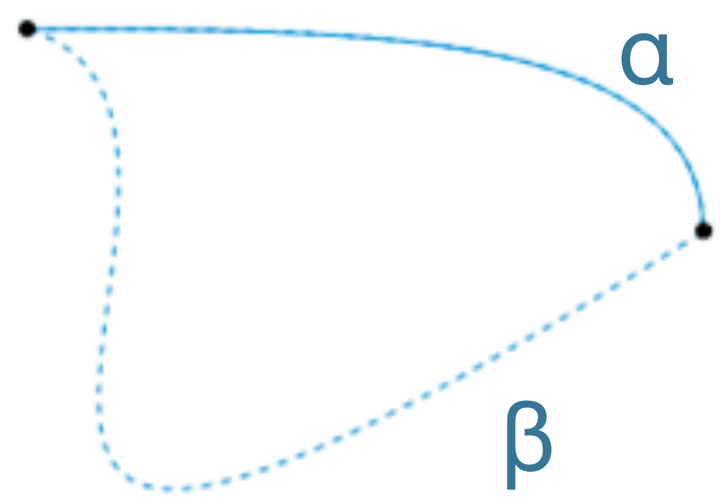
Homotopy

Deformation of one path into another



Homotopy

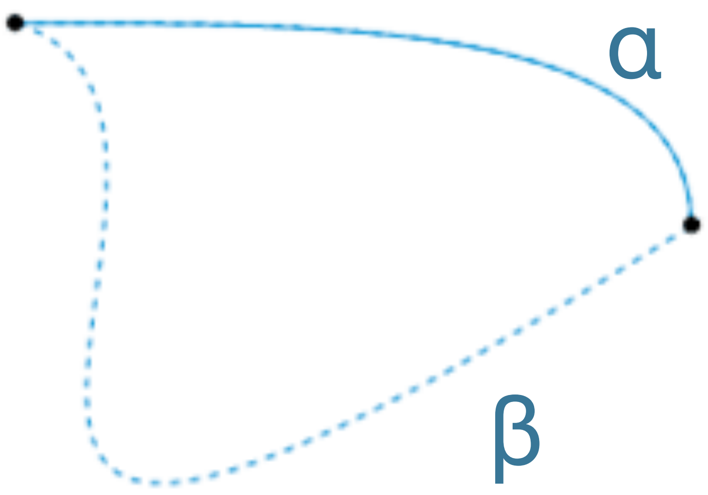
Deformation of one path into another



= 2-dimensional *path between paths*

Homotopy

Deformation of one path into another

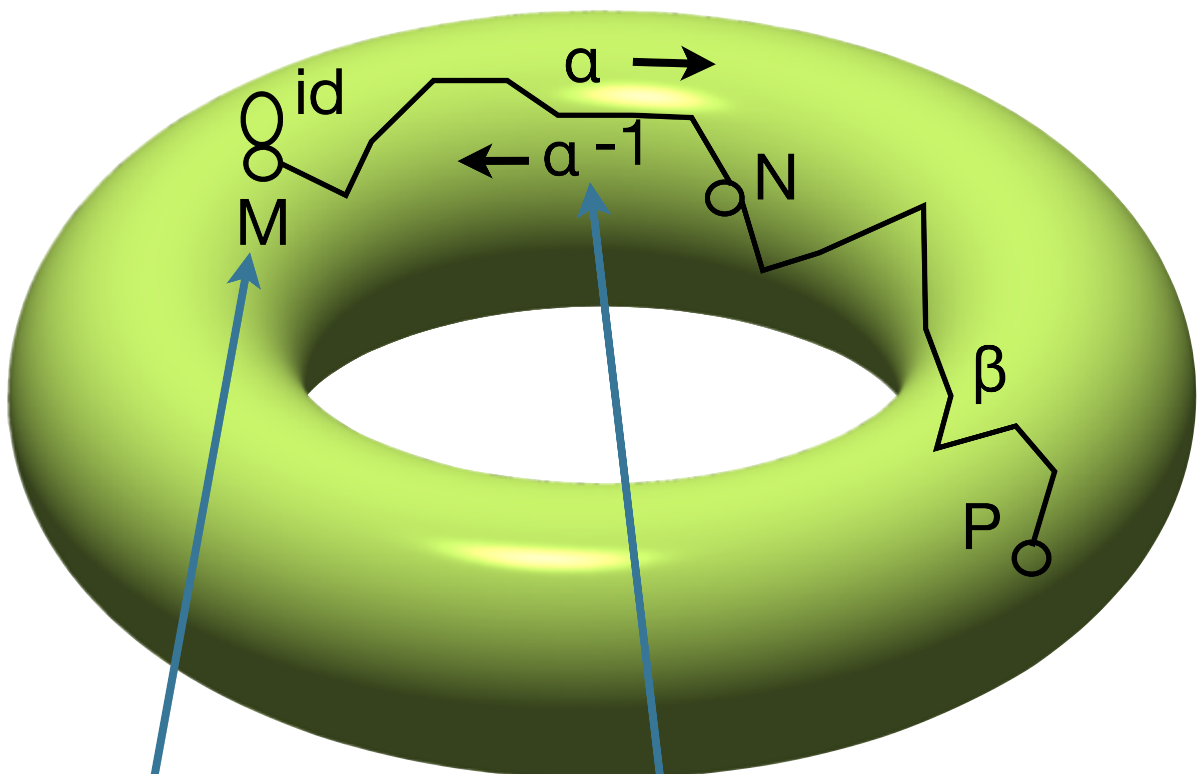


$\delta : \alpha =_{x=y} \beta$

= 2-dimensional *path between paths*

Types as spaces

type A is a space



programs

M : A

are points

proofs of equality

$\alpha : M =_A N$

are paths

path operations

id : M = M (refl)

α^{-1} : N = M (sym)

β o α : M = P (trans)

homotopies

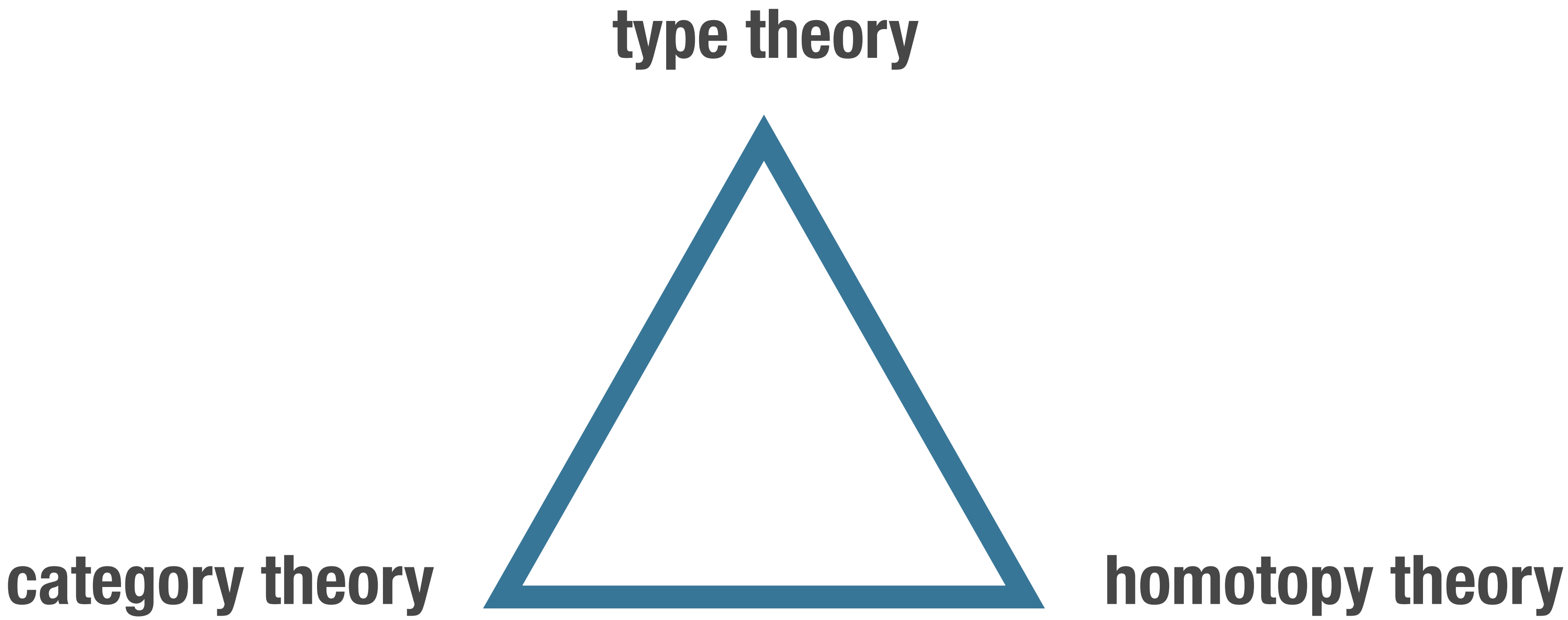
ul : id o α $=_{M=N}$ α

il : α^{-1} o α $=_{M=M}$ id

asc : γ o (β o α)

$=_{M=P}$ (γ o β) o α

Homotopy Type Theory



Types as ∞-groupoids

type A **is an ∞-groupoid**

- infinite-dimensional algebraic structure, with morphisms, morphisms between morphisms, ...
- each level has a groupoid structure, and they interact

morphisms

```
id        : M = M (refl)
α⁻¹       : N = M (sym)
β o α     : M = P (trans)
```

morphisms between morphisms

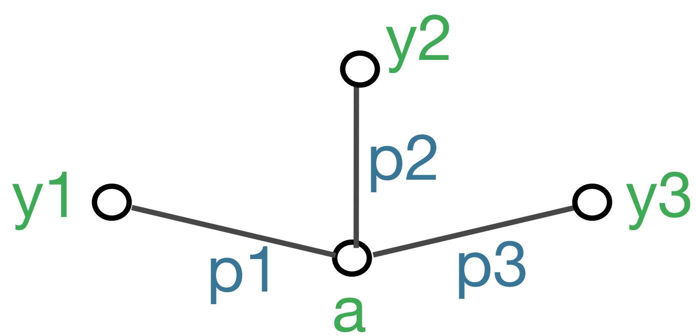
ul : id o α $=_{M=N}$ α

il : α^{-1} o α $=_{M=M}$ id

asc : γ o (β o α) $=_{M=P}$ (γ o β) o α

Path induction

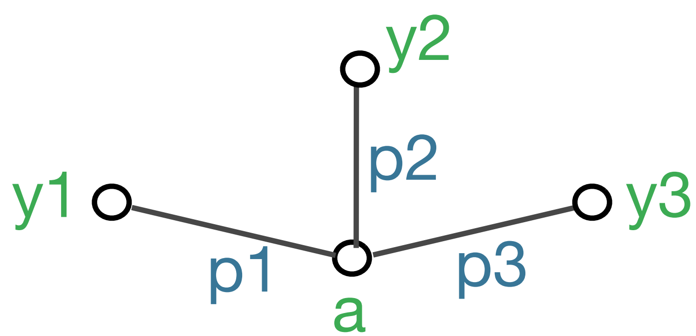
Type of paths from a to somewhere



is inductively generated by

Path induction

Type of paths from a to somewhere



is inductively generated by

Fix a type A with element a:A.

For a family of types C(y:A, p:a=y),

to give an element of

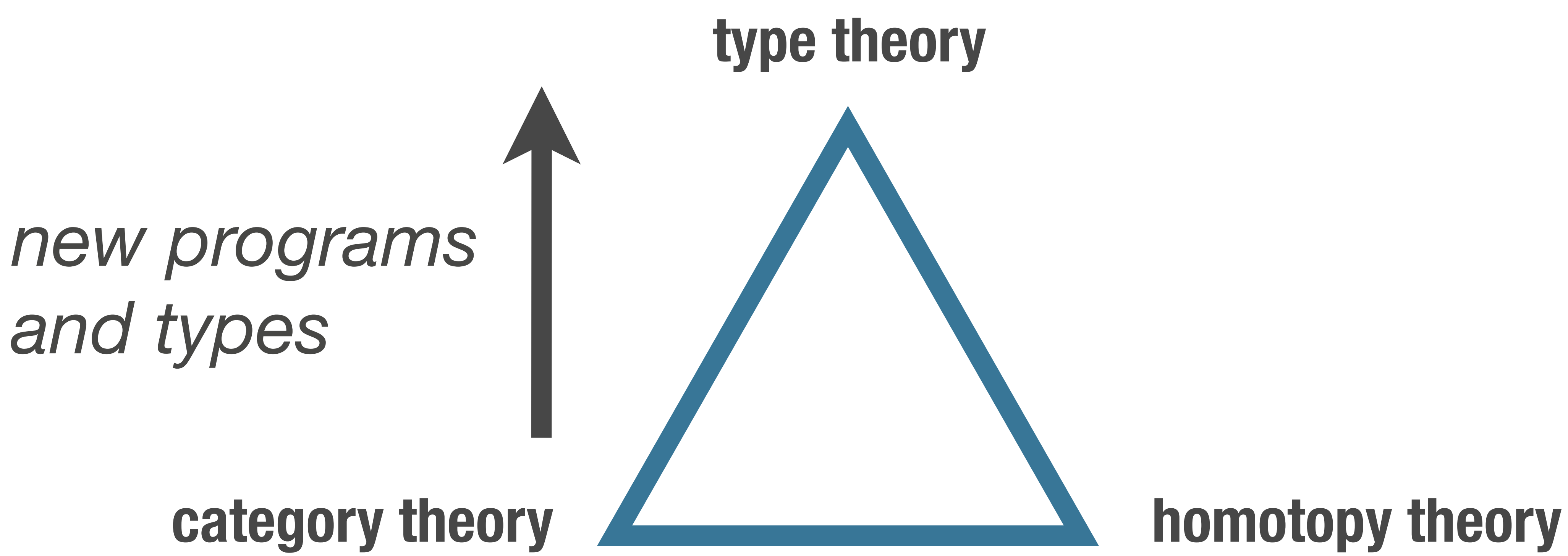
C(y,p) for all y and p:a=y,

suffices to give an element of

C(a,id)

Type theory is a synthetic theory of spaces/∞-groupoids

Homotopy Type Theory



Univalence [Voevodsky]

Univalence [Voevodsky]

- *Equivalence of types* is a generalization to spaces of bijection of sets

Univalence [Voevodsky]

- *Equivalence of types* is a generalization to spaces of bijection of sets
- Univalence axiom:
  equality of types (`A =Type B`) is (equivalent to) equivalence of types (`Equiv A B`)

Univalence [Voevodsky]

- *Equivalence of types* is a generalization to spaces of bijection of sets

- Univalence axiom:
  equality of types (`A =Type B`) is (equivalent to) equivalence of types (`Equiv A B`)

- ∴ all structures/properties respect equivalence

Univalence [Voevodsky]

- *Equivalence of types* is a generalization to spaces of bijection of sets
- Univalence axiom:
  equality of types (`A =Type B`) is (equivalent to) equivalence of types (`Equiv A B`)
- ∴ all structures/properties respect equivalence
- Not by collapsing equivalence,
  but by exploiting proof-relevant equality:
  `transport` *does real work*

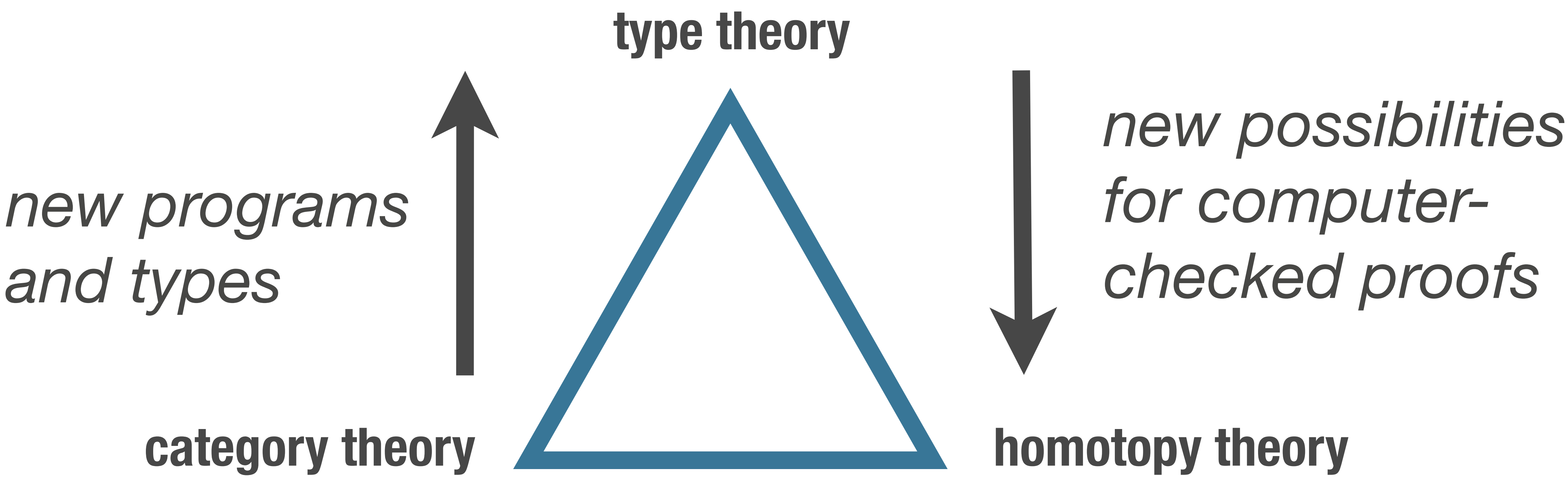
Higher inductive types

[Bauer,Lumsdaine,Shulman,Warren]

New way of forming types:

Inductive type specified by generators
not only for points (elements), but also for paths

Homotopy Type Theory



Basis for this formalization

Basis for this formalization

- Agda proof assistant [Norell, Abel, Danielsson]

Basis for this formalization

- Agda proof assistant [Norell, Abel, Danielsson]
- 10,000 line HoTT library

Basis for this formalization

- Agda proof assistant [Norell, Abel, Danielsson]
- 10,000 line HoTT library
- essentially no automation

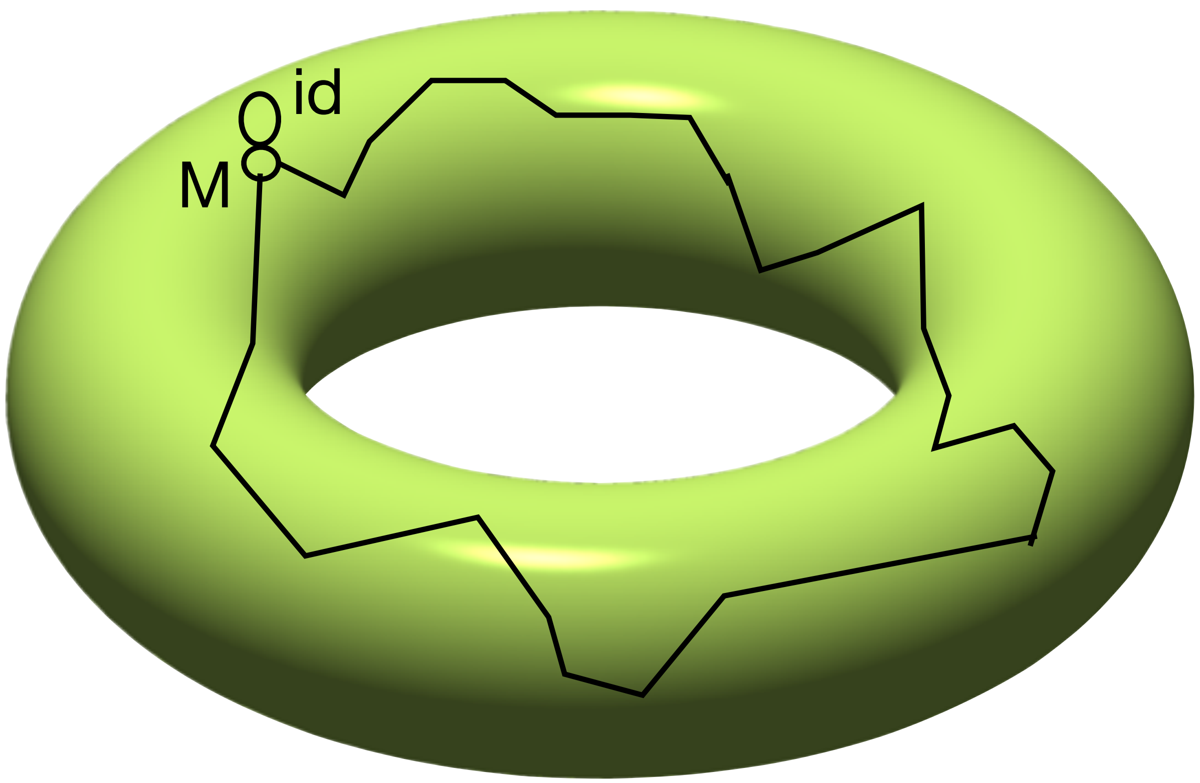
Outline

1. Eilenberg-MacLane spaces
2. K(G,1)
3. K(G,n)

Outline

1. **Eilenberg-MacLane spaces**
2. K(G,1)
3. K(G,n)

Types as spaces



loop operations

id : M = M (refl)

α^{-1} : M = M (sym)

β o α : M = M (trans)

homotopies

ul : id o α = α

il : α^{-1} o α = id

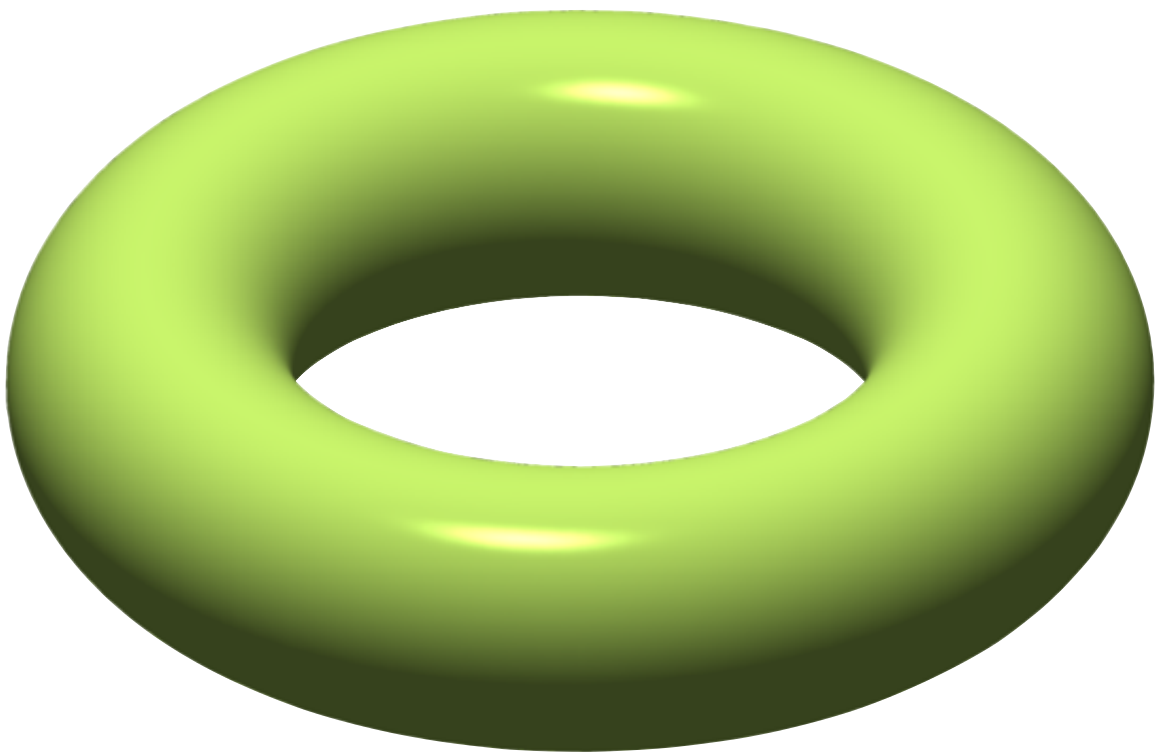
asc : γ o (β o α)
= (γ o β) o α

Homotopy Groups

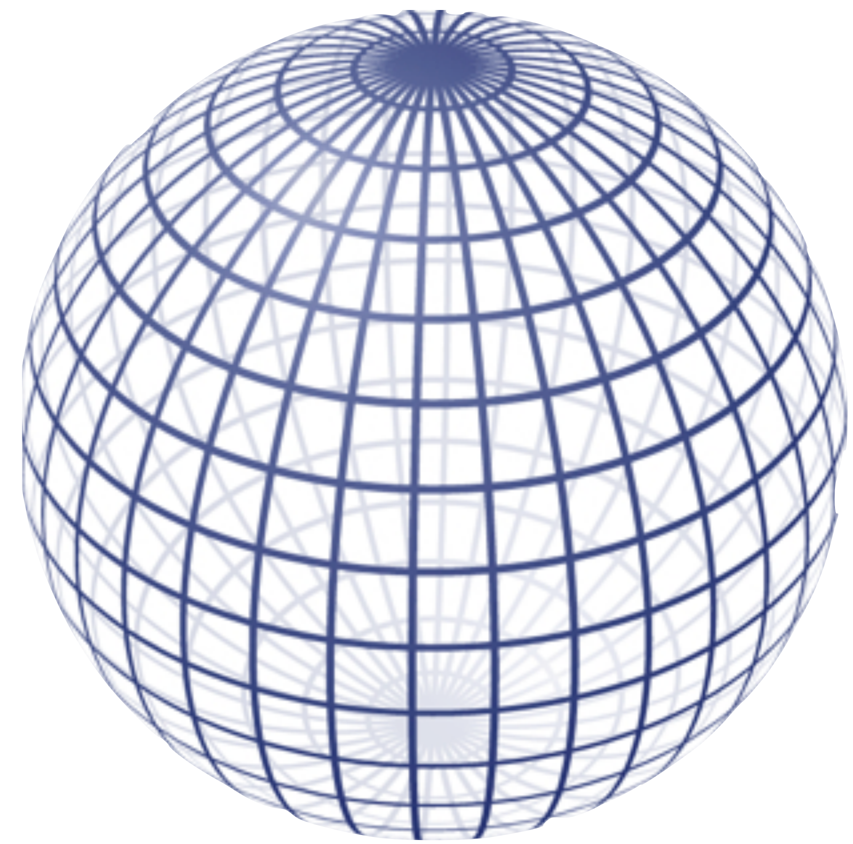
Homotopy groups of a space X:

- $\pi_1(X)$ is fundamental group (group of loops)
- $\pi_2(X)$ is group of homotopies (2-dimensional loops)
- $\pi_3(X)$ is group of 3-dimensional loops
- ...

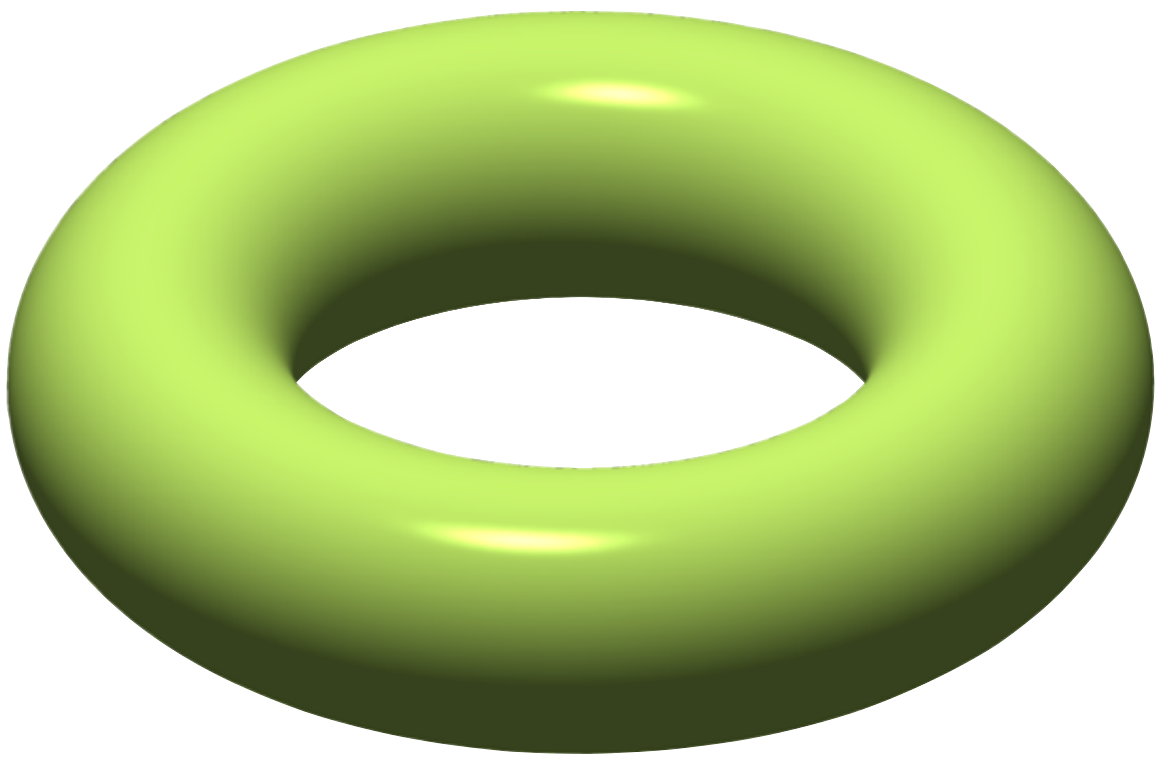
Telling spaces apart



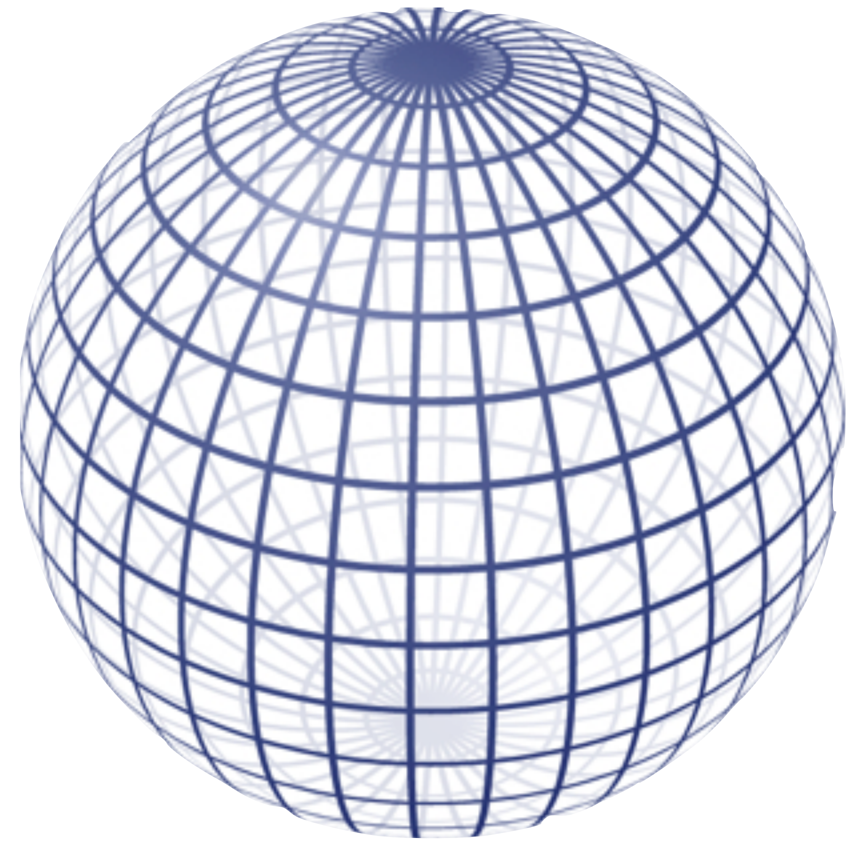
≠



Telling spaces apart



$\not\simeq$



fundamental group
is non-trivial ($\mathbb{Z} \times \mathbb{Z}$)

fundamental group
is trivial

Homotopy Groups

A

$x =_A x$ $\Omega(A,x)$

$id =_{x=x} id$ $\Omega^2(A,x)$

$id =_{\Omega 2(A,x)} id$ $\Omega^3(A,x)$

⋮

Homotopy Groups

A

$x =_A x$ $\qquad \Omega(A,x)$

$id =_{x=x} id$ $\qquad \Omega^2(A,x)$

$id =_{\Omega 2(A,x)} id$ $\qquad \Omega^3(A,x)$

$\vdots$

$$\pi_n(A,x) = ||\Omega^n(A,X)||_0$$

Homotopy Groups

A

$x =_A x$ $\Omega(A,x)$

$id =_{x=x} id$ $\Omega^2(A,x)$

$id =_{\Omega 2(A,x)} id$ $\Omega^3(A,x)$

⋮

$\pi_n(A,x) = ||\Omega^n(A,X)||_0$

0-truncation = set of connected components = all paths are equal

Eilenberg-MacLane Space

For a group G

K(G,1) is a space such that
$\pi_1(K(G,1)) = G$ and
$\pi_k(K(G,1)) = 1$ otherwise

K(G,n) is a space such that (G abelian)
$\pi_n(K(G,n)) = G$ and
$\pi_k(K(G,n)) = 1$ otherwise

Spaces with specified groups

Find X with $\pi_1(X) = G$ and $\pi_2(X) = H$

Define $X = K(G,1) \times K(H,2)$

$$\begin{aligned}\pi_1(X) &= \pi_1(K(G,1)) \times \pi_1(K(H,2)) \\ &= G \times 1 \\ &= G\end{aligned}$$

$$\begin{aligned}\pi_2(X) &= \pi_2(K(G,1)) \times \pi_2(K(H,2)) \\ &= 1 \times H \\ &= H\end{aligned}$$

Cohomology

Homotopy groups aren't the only invariant:
homology groups, cohomology groups

Define *ordinary cohomology with coefficients in G* by

$$H^n(A) = \|A \to K(G,n)\|_0$$

Cohomology

Homotopy groups aren't the only invariant:
homology groups, cohomology groups

Define *ordinary cohomology with coefficients in G* by

$$H^n(A) = \|A \to K(G,n)\|_0$$

satisfies (constructive) Eilenberg-Steenrod axioms

Eilenberg-MacLane space

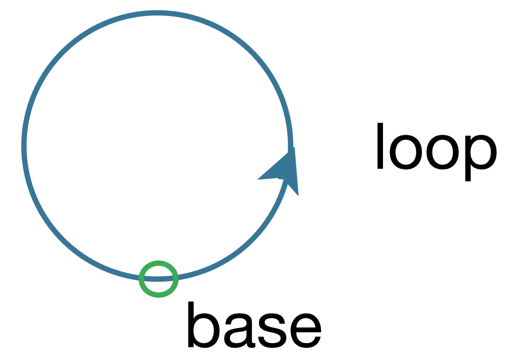
Can we build Eilenberg-MacLane spaces from higher inductive types?

Outline

1. Eilenberg-MacLane spaces
2. **K(G,1)**
3. K(G,n)
4. Proofs

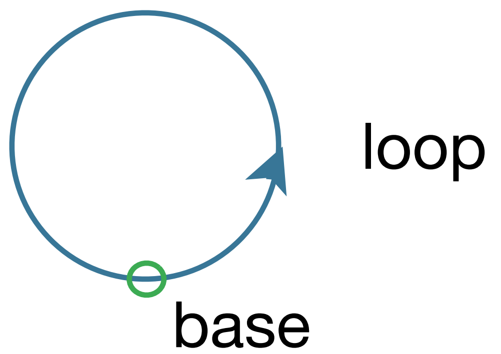
Fundamental group of circle

How many different loops are there on the circle, up to homotopy?



Fundamental group of circle

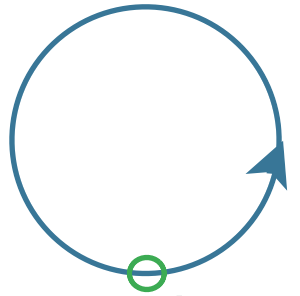
How many different loops are there on the circle, up to homotopy?




```
id
```

Fundamental group of circle

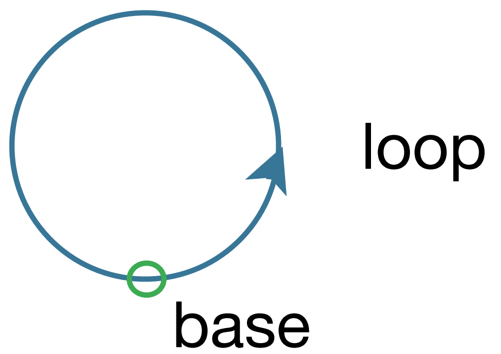
How many different loops are there on the circle, up to homotopy?




```
id
loop
```

Fundamental group of circle

How many different loops are there on the circle, up to homotopy?



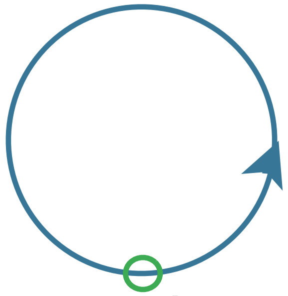
`id`

`loop`

`loop`$^{-1}$

Fundamental group of circle

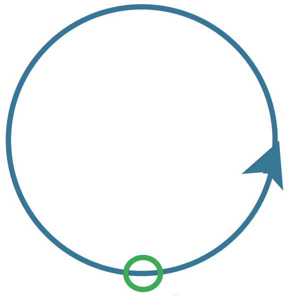
How many different loops are there on the circle, up to homotopy?




```
id
loop
loop⁻¹
loop o loop
```

Fundamental group of circle

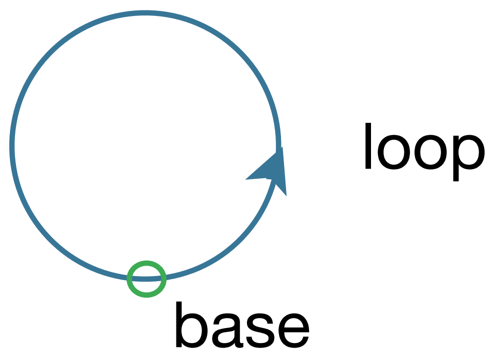
How many different loops are there on the circle, up to homotopy?




```
id
loop
loop⁻¹
loop o loop
loop⁻¹ o loop⁻¹
```

Fundamental group of circle

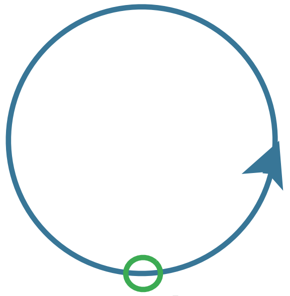
How many different loops are there on the circle, up to homotopy?




```
id
loop
loop⁻¹
loop o loop
loop⁻¹ o loop⁻¹
loop o loop⁻¹
```

Fundamental group of circle

How many different loops are there on the circle, up to homotopy?




```
id
loop
loop-1
loop o loop
loop-1 o loop-1
loop o loop-1  = id
```

$\texttt{id}$

$\texttt{loop}$

$\texttt{loop}^{-1}$

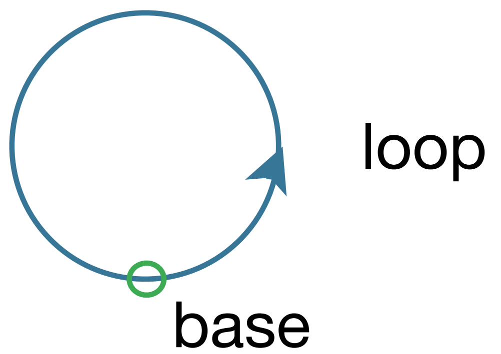
$\texttt{loop} \circ \texttt{loop}$

$\texttt{loop}^{-1} \circ \texttt{loop}^{-1}$

$\texttt{loop} \circ \texttt{loop}^{-1} = \texttt{id}$

Fundamental group of circle

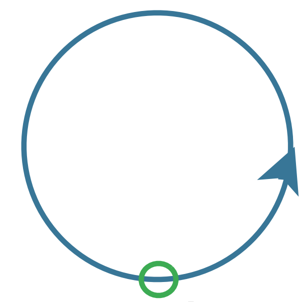
How many different loops are there on the circle, up to homotopy?




```
id                      0
loop
loop⁻¹
loop o loop
loop⁻¹ o loop⁻¹
loop o loop⁻¹   = id
```

Fundamental group of circle

How many different loops are there on the circle, up to homotopy?




```
id                      0
loop                    1
```
$\texttt{loop}^{-1}$

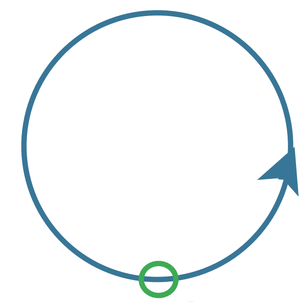
$\texttt{loop o loop}$

$\texttt{loop}^{-1}\ \texttt{o loop}^{-1}$

$\texttt{loop o loop}^{-1}$ = id

Fundamental group of circle

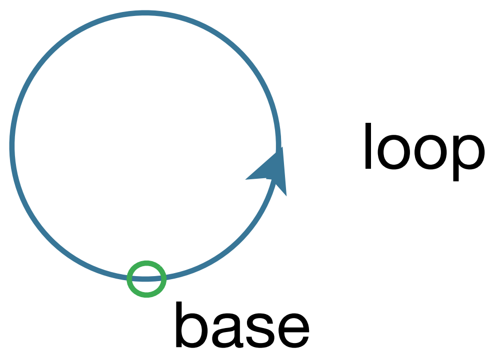
How many different loops are there on the circle, up to homotopy?



`id`	`0`
`loop`	`1`
$\texttt{loop}^{-1}$	`-1`
`loop o loop`	
$\texttt{loop}^{-1}\ \texttt{o}\ \texttt{loop}^{-1}$	
$\texttt{loop o loop}^{-1}\ = \texttt{id}$	

Fundamental group of circle

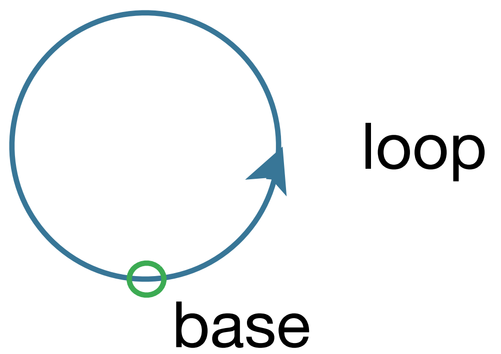
How many different loops are there on the circle, up to homotopy?



`id`	`0`
`loop`	`1`
$\texttt{loop}^{-1}$	`-1`
`loop o loop`	`2`
$\texttt{loop}^{-1}$ `o` $\texttt{loop}^{-1}$	
`loop o` $\texttt{loop}^{-1}$ `= id`	

Fundamental group of circle

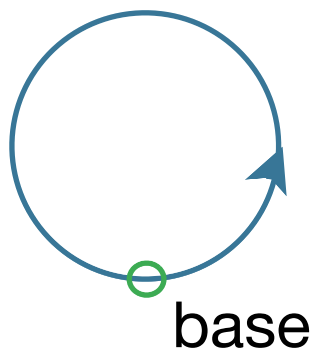
How many different loops are there on the circle, up to homotopy?



`id`	`0`
`loop`	`1`
`loop`$^{-1}$	`-1`
`loop o loop`	`2`
`loop`$^{-1}$ `o loop`$^{-1}$	`-2`
`loop o loop`$^{-1}$ `= id`	

Fundamental group of circle

How many different loops are there on the circle, up to homotopy?



`id`	`0`
`loop`	`1`
`loop`$^{-1}$	`-1`
`loop o loop`	`2`
`loop`$^{-1}$ `o loop`$^{-1}$	`-2`
`loop o loop`$^{-1}$ `= id`	`0`

Fundamental group of circle

The circle S^1 is a space such that

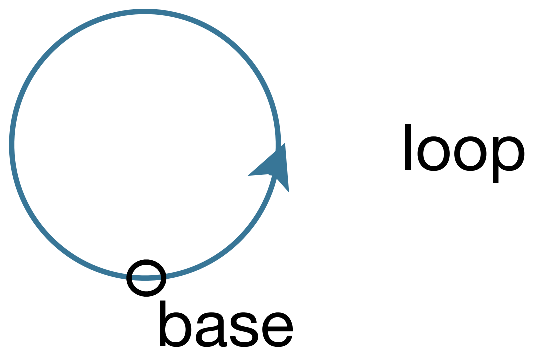
$\pi_1(S^1) = \mathbb{Z}$ and

$\pi_k(S^1) = 1$ otherwise

The circle is $K(\mathbb{Z},1)$

The Circle

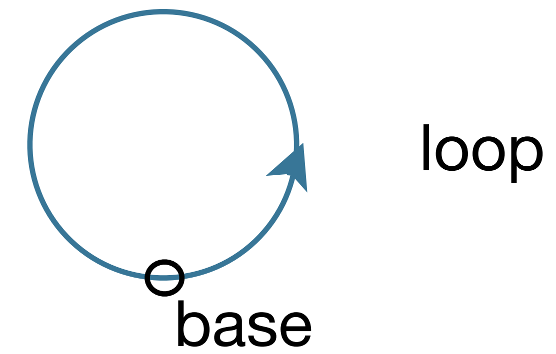
Circle S^1 is a **higher inductive type** generated by



The Circle

Circle S^1 is a **higher inductive type** generated by

```
base : S¹
loop : base = base
```

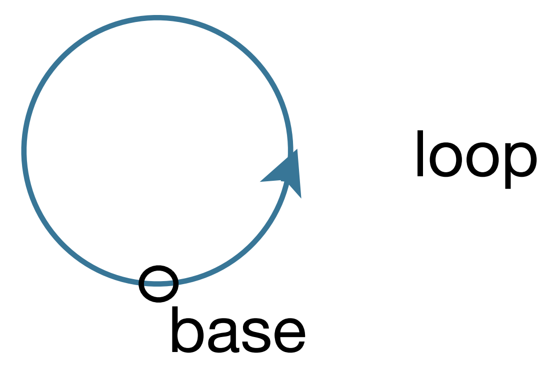


The Circle

Circle S^1 is a **higher inductive type** generated by

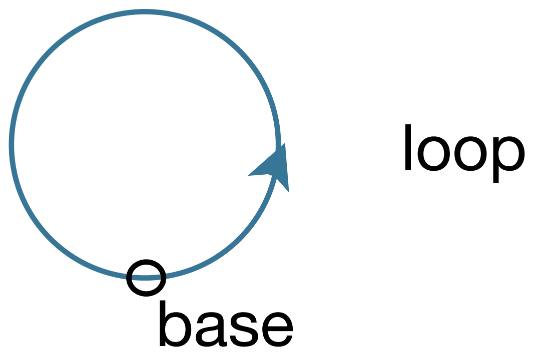
point `base : S¹`

`loop : base = base`



The Circle

Circle S^1 is a **higher inductive type** generated by

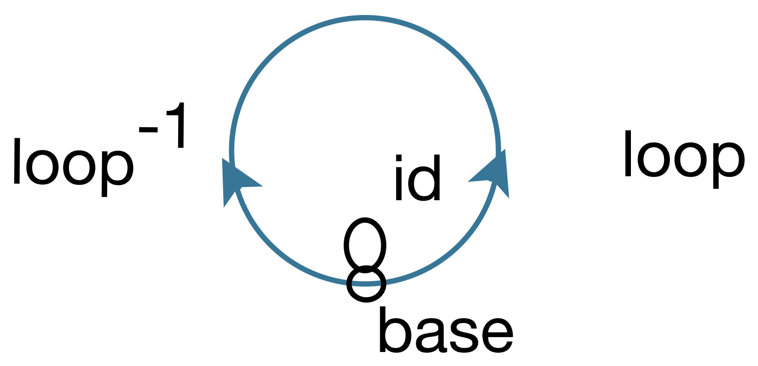


point `base : S¹`

path `loop : base = base`

The Circle

Circle S^1 is a **higher inductive type** generated by



point base : S^1

path loop : base = base

Free type: equipped with structure

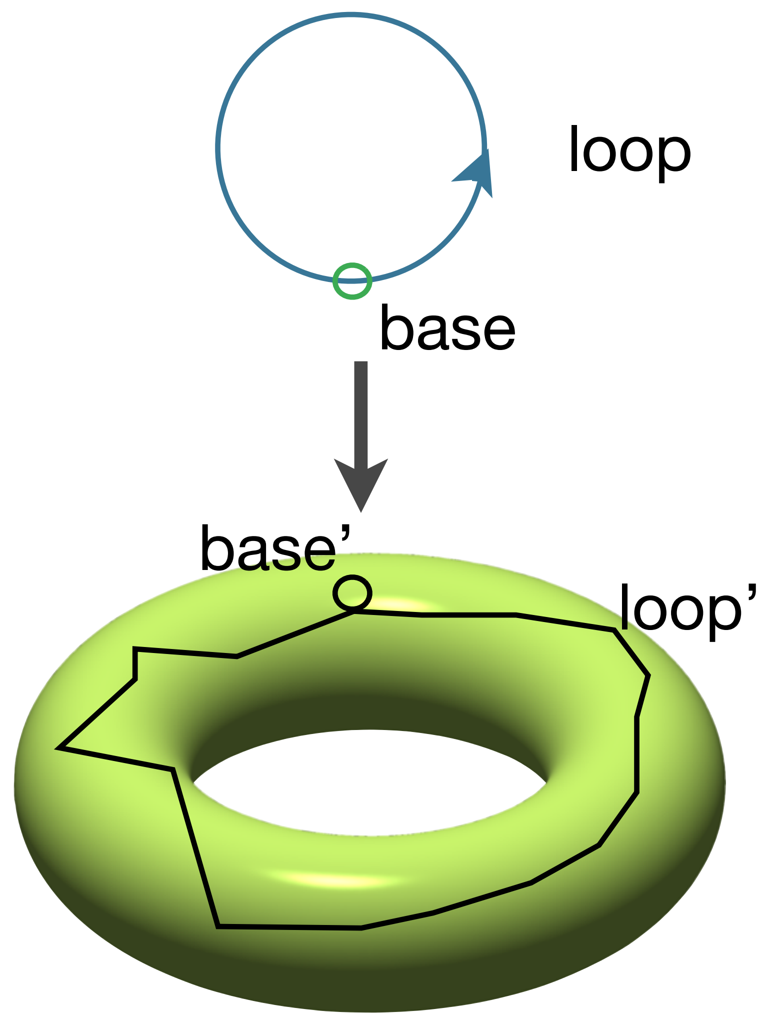
```
id                 inv : loop o loop⁻¹ = id
loop⁻¹             ...
loop o loop
```

The Circle

Circle recursion:

function $S^1 \to X$ determined by

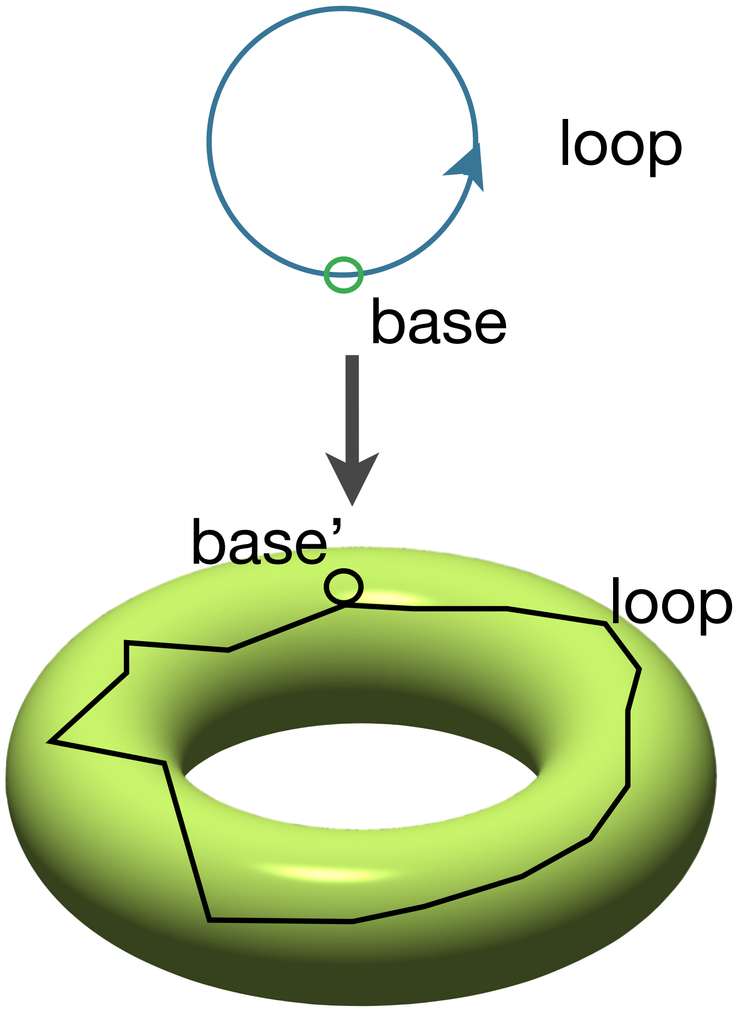
```
base' : X
loop' : base' = base'
```



The Circle

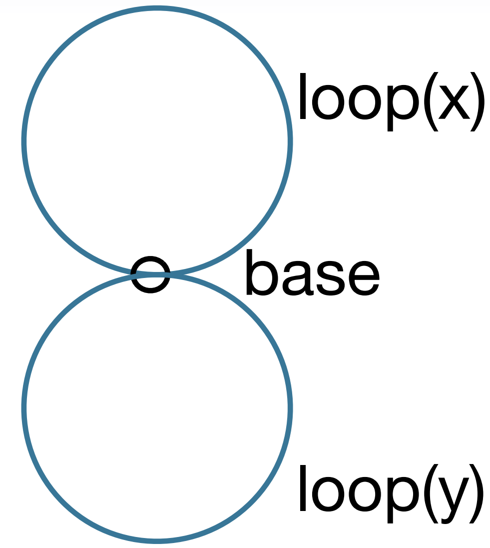
Circle recursion:
function $S^1 \rightarrow X$ determined by

```
base' : X
loop' : base' = base'
```



Circle induction: To prove a predicate P for all points on the circle, suffices to prove `P(base)`, continuously in the loop

K(G,1)

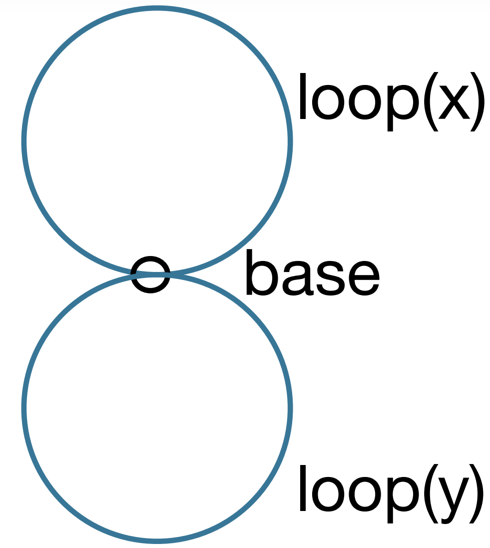


K(G,1) is a **higher inductive type** generated by

```
K(G,1) :   type

base : K(G,1)
loop : G → base=base
loop-ident : loop(1_G) = id
loop-comp : loop(x·_G y) = loop(x)·loop(y)
```

K(G,1)

K(G,1) is a **higher inductive type** generated by

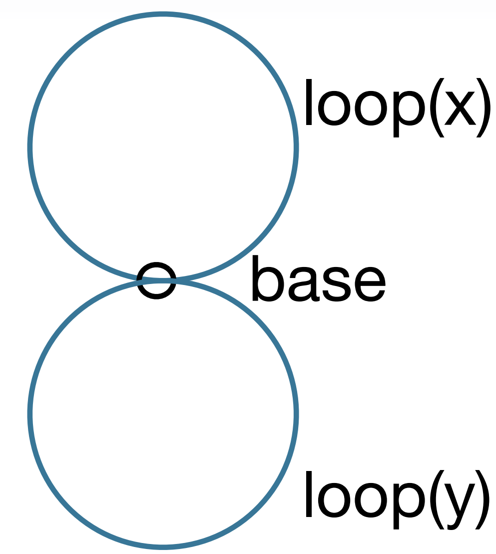



```
K(G,1) :   type

base : K(G,1)
loop : G → base=base
loop-ident : loop(1_G) = id
loop-comp : loop(x·_G y) = loop(x)·loop(y)
```

group homomorphism from G to Ω(K(G,1))

K(G,1)

K(G,1) is a **higher inductive type** generated by

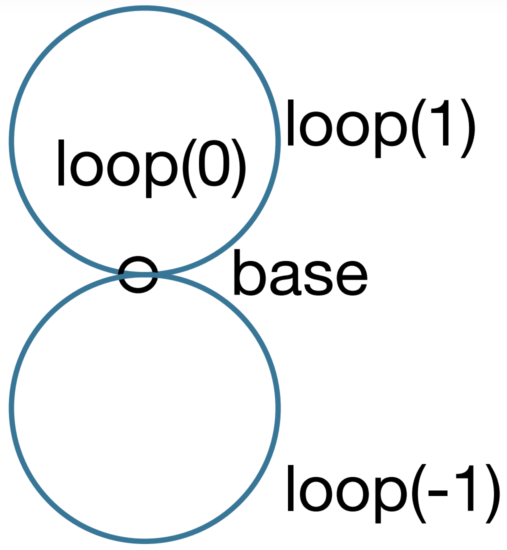



```
K(G,1) : 1-type

base : K(G,1)
loop : G → base=base
loop-ident : loop(1_G) = id
loop-comp : loop(x·_G y) = loop(x)·loop(y)
```

group homomorphism from G to Ω(K(G,1))

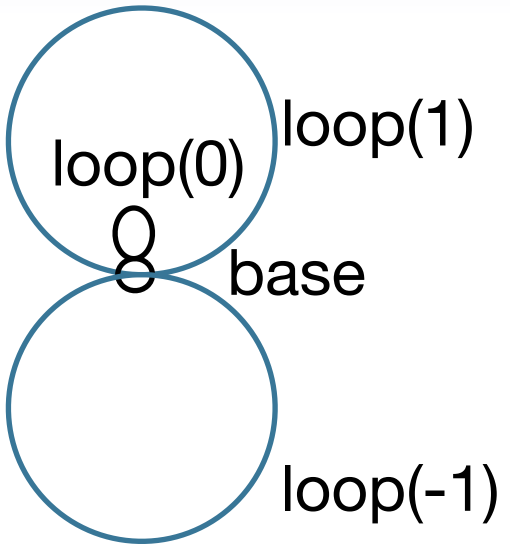
K($\mathbb{Z}$,1) revisited

K($\mathbb{Z}$,1) is *equivalent* to previous S^1



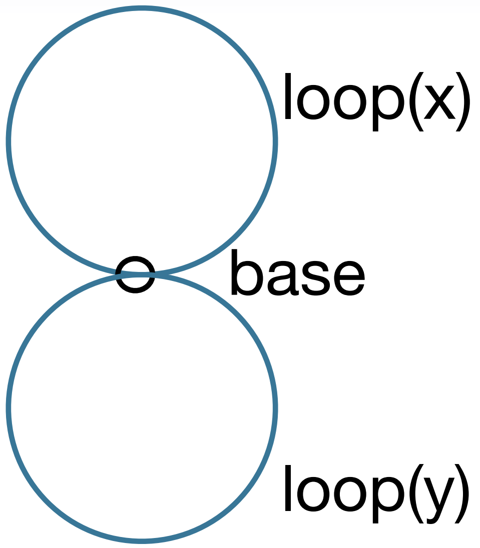
K($\mathbb{Z}$,1) revisited

K($\mathbb{Z}$,1) is *equivalent* to previous S^1

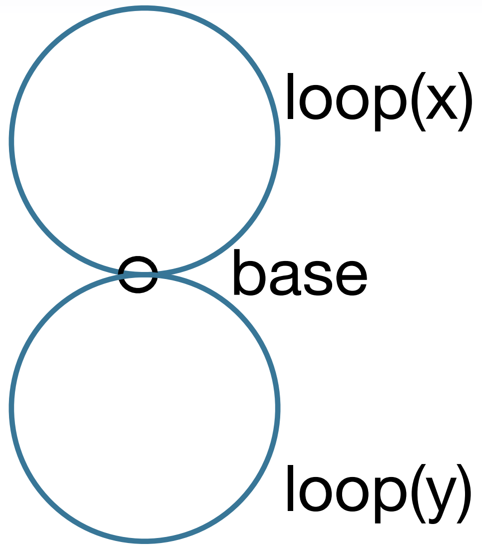



```
loop(0) = id

loop(1)

loop(-1) = !loop(1)

loop(2) = loop(1) · loop(1)
```

K(G,1) recursion



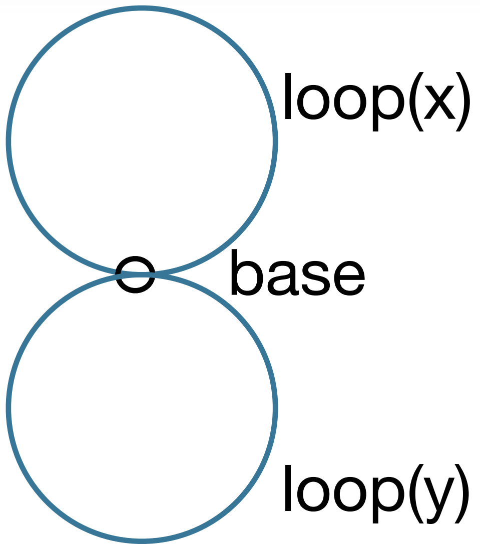
K(G,1) recursion



To define f : K(G,1) → C

- show C is a 1-type
- give f(base) : C
- f(loop) : group homomorphism from G to Ω(C,f(base))

$\pi_1(K(G,1)) = G$

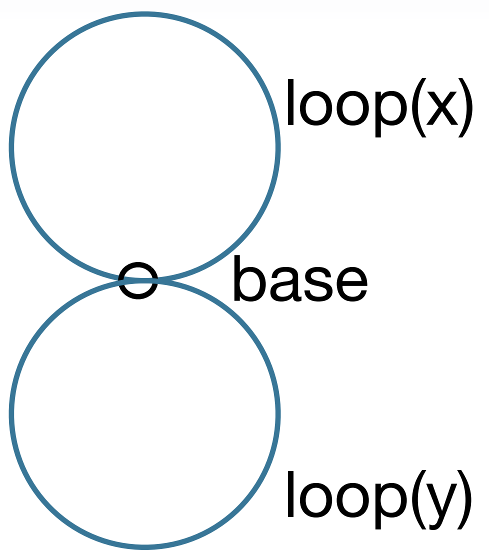


$\pi_1(K(G,1)) = G$

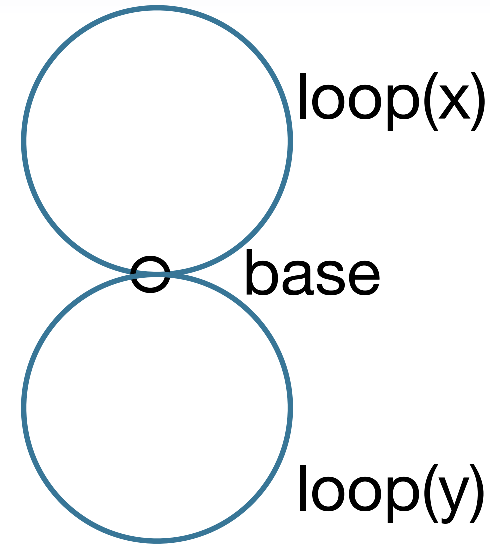
1. `Codes : K(G,1) → 1-Type`

   `Codes(base) = G`

   `Codes(loop(x)) =`
   “multiplication by x”



$\pi_1(K(G,1)) = G$



1. `Codes : K(G,1) → 1-Type`

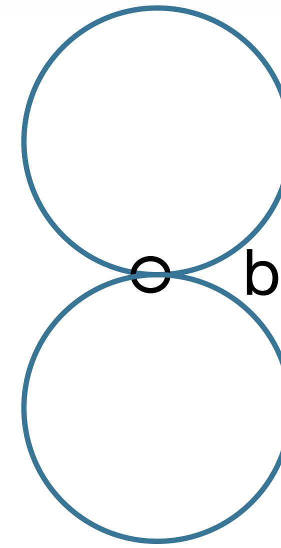
   `Codes(base) = G`

   `Codes(loop(x)) =`
   "multiplication by x"

2. `encode : Ω(K(G,1)) → G`

   `encode(p) =` $\texttt{transport}_{\texttt{Codes}}(\texttt{p}, 1_{\texttt{G}})$

$\pi_1(K(G,1)) = G$



1. `Codes : K(G,1) → 1-Type`

   `Codes(base) = G`

   `Codes(loop(x)) =`
   "multiplication by x"

2. `encode : Ω(K(G,1)) → G`

   `encode(p) =` $\texttt{transport}_{\texttt{Codes}}(\texttt{p}, \texttt{1}_{\texttt{G}})$

3. Calculate that `encode` and `loop` are mutually inverse using encode-decode method

```
comp-equiv : ∀ g -> Equiv El El
comp-equiv a = (improve (hequiv (\ x -> comp x a)
                                (\ x -> comp x (inv a))
                                (λ x → (unitr x ∘ ap (λ y → comp x y) (invr a)) ∘ assoc x a (inv a))
                                (λ x → (unitr x ∘ ap (λ y → comp x y) (invl a)) ∘ assoc x (inv a) a)))

decode' : El → Path{KG1} KG1.base KG1.base
decode' = KG1.loop

module Codes where

  f : ∀ g → (El , El-level) ≃ (El , El-level)
  f =  λ g → coe (Path-NTypes (tl 0)) (ua (comp-equiv g))

  abstract
      pri : f ident ≃ id
      pri = coe (! (Path2-NTypes (tl 0) _ _))
                (type≃-ext (ua (comp-equiv ident)) id
                           (λ x → unitr x ∘ ap≃ (type≃β (comp-equiv ident)) {x})
                 ∘ Path-NTypesβ (tl 0) (ua (comp-equiv ident)))

      prc : ∀ g1 g2 -> f (comp g1 g2) ≃ f g2 ∘ f g1
      prc g1 g2 = coe (! (Path2-NTypes (tl 0) _ _))
                      (! (ap-∘ fst (f g2) (f g1)) ∘
                       ! (ap (λ x → x ∘ ap fst (f g1)) (Path-NTypesβ (tl 0) (ua (comp-equiv g2)))) ∘
                       ! (ap (λ x → ua (comp-equiv g2) ∘ x) (Path-NTypesβ (tl 0) (ua (comp-equiv g1)))) ∘
                       type≃-ext (ua (comp-equiv (comp g1 g2))) (ua (comp-equiv g2) ∘ ua (comp-equiv g1))
                                 (λ g → ! (ap≃ (transport-∘ (λ x → x) (ua (comp-equiv g2)) (ua (comp-equiv g1)))) ∘
                                        (! (ap≃ (type≃β (comp-equiv g2))) ∘ ! (ap (λ x → fst (comp-equiv g2) x) (ap≃ (type≃β (comp-equiv g1)))) ∘
                                        ! (assoc g g1 g2)) ∘
                                        ap≃ (type≃β (comp-equiv (comp g1 g2))))
                       ∘ Path-NTypesβ (tl 0) (ua (comp-equiv (comp g1 g2))))

Codes : KG1 → NTypes (tl 0)
Codes = KG1-rec (NTypes-level (tl 0))
                (El , El-level)
                (record { f = Codes.f;
                          pres-ident = Codes.pri ;
                          pres-comp = Codes.prc })


abstract
  transport-Codes-loop : ∀ g g' -> (transport (fst o Codes) (KG1.loop g) g') ≃ comp g' g
  transport-Codes-loop g g' = transport (fst o Codes) (KG1.loop g) g' ≃⟨ ap≃ (transport-ap-assoc' fst Codes (KG1.loop g)) ⟩
                              transport fst (ap Codes (KG1.loop g)) g' ≃⟨ ap (λ x → transport fst x g') (KG1.KG1-rec/βloop{_}{NTypes-level (tl 0)}
                                                                         (record {f = Codes.f; pres-ident = Codes.pri; pres-comp = Codes.prc })) ⟩
                              transport fst (coe (Path-NTypes (tl 0)) (ua (comp-equiv g))) g' ≃⟨ ap≃ (transport-ap-assoc fst (coe (Path-NTypes (tl 0))
                                                                                                   (ua (comp-equiv g)))) ⟩
                              coe (fst≃ (coe (Path-NTypes (tl 0)) (ua (comp-equiv g)))) g' ≃⟨ ap (λ x → coe x g') (Path-NTypesβ (tl 0) (ua (comp-equiv g))) ⟩
                              coe (ua (comp-equiv g)) g' ≃⟨ ap≃ (type≃β (comp-equiv g)) ⟩
                              comp g' g ∎

encode : {x : KG1} -> Path KG1.base x -> fst (Codes x)
encode α = transport (fst o Codes) α ident

abstract
  encode-decode' : ∀ x -> encode (decode' x) ≃ x
  encode-decode' x = encode (decode' x) ≃⟨ id ⟩
                     encode (KG1.loop x) ≃⟨ id ⟩
                     transport (fst o Codes) (KG1.loop x) ident ≃⟨ transport-Codes-loop x ident ⟩
                     comp ident x ≃⟨ unitl x ⟩
                     x ∎

decode : {x : _} -> fst (Codes x) -> Path KG1.base x
decode {x} = KG1-elim (λ x' → (fst (Codes x') → Path KG1.base x') , Πlevel (λ _ → path-preserves-level KG1.level))
                      decode'
                      loop'
                      (λ _ → HSet-UIP (Πlevel (λ _ → use-level KG1.level _ _)) _ _ _ _)
                      (λ _ _ → HSet-UIP (Πlevel (λ _ → use-level KG1.level _ _)) _ _ _ _)
                      x where
  abstract
    loop' : ∀ g -> transport (\x -> fst (Codes x) -> Path KG1.base x) (KG1.loop g) decode' ≃ decode'
    loop' = (λ g → transport-→-from-square (fst o Codes) (Path KG1.base) (KG1.loop g) decode' decode'
                        (λ≃ (\ g' ->
                          (transport (Path KG1.base) (KG1.loop g) (decode' g') ≃⟨ id ⟩
                           transport (Path KG1.base) (KG1.loop g) (KG1.loop g') ≃⟨ transport-Path-right (KG1.loop g) (KG1.loop g') ⟩
                           (KG1.loop g) ∘ (KG1.loop g') ≃⟨ ! (KG1.loop-comp g' g) ⟩
                           KG1.loop (comp g' g) ≃⟨ ap KG1.loop (! (transport-Codes-loop g g')) ⟩
                           KG1.loop (transport (fst o Codes) (KG1.loop g) g') ≃⟨ id ⟩
                           decode' (transport (fst o Codes) (KG1.loop g) g') ∎))))


decode-encode : ∀ {x} (α : Path KG1.base x) -> decode (encode α) ≃ α
decode-encode id = KG1.loop-ident

Ω1[KG1]-Equiv-G : Equiv (Path{KG1} KG1.base KG1.base) El
Ω1[KG1]-Equiv-G = improve (hequiv encode decode decode-encode encode-decode')
```

Eilenberg-MacLane space

So far: For a group G, can define a space K(G,1) such that

$\pi_1(K(G,1)) = G$ and

$\pi_k(K(G,1)) = 1$ otherwise

Outline

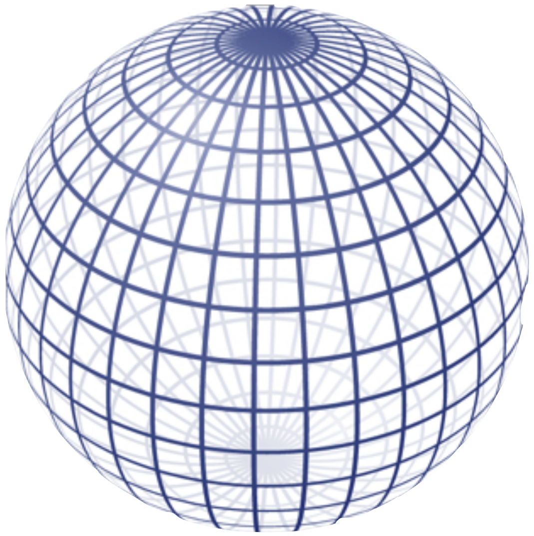
1. Eilenberg-MacLane spaces

2. $K(G,1)$

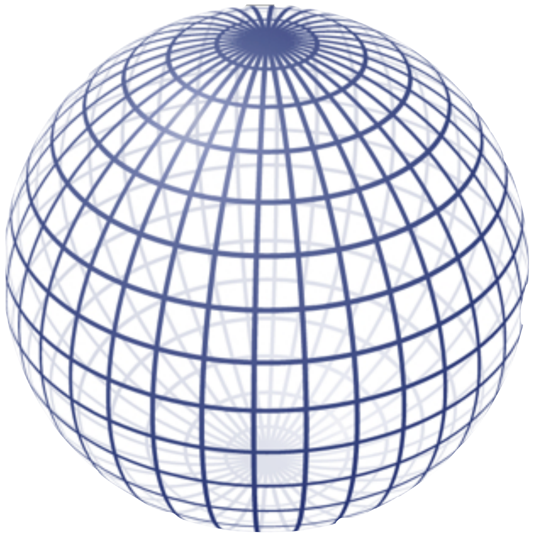
3. **$K(G,n)$**

Sphere S^2

Sphere S^2



Sphere S^2



- π_1 is trivial: inside of any loop can be filled
- π_2 is $\mathbb{Z}$: 2-paths on sphere = paths on circle
- $\pi_{k>2}$ is …

Homotopy Groups of S^2

k[th] homotopy group

n-dimensional sphere

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}	π_{14}	π_{15}
S^0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S^1	$\mathbf{Z}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S^2	0	$\mathbf{Z}$	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{12}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$
S^3	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{12}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$
S^4	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}\times\mathbf{Z}_{12}$	$\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{24}\times\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^3$	$\mathbf{Z}_{120}\times\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^5$
S^5	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{30}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^3$	$\mathbf{Z}_{72}\times\mathbf{Z}_2$
S^6	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_{60}$	$\mathbf{Z}_{24}\times\mathbf{Z}_2$	$\mathbf{Z}_2^3$
S^7	0	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	0	$\mathbf{Z}_2$	$\mathbf{Z}_{120}$	$\mathbf{Z}_2^3$
S^8	0	0	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	0	$\mathbf{Z}_2$	$\mathbf{Z}\times\mathbf{Z}_{120}$

[image from wikipedia]

Homotopy Groups of S^2

k[th] homotopy group

n-dimensional sphere

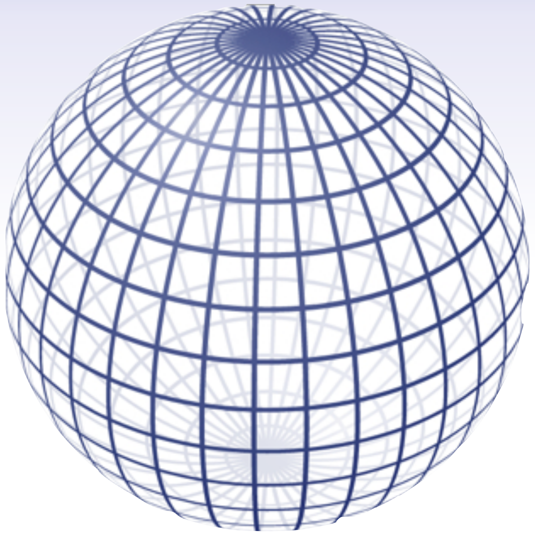
	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}	π_{14}	π_{15}
S^0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S^1	Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S^2	0	Z	Z	Z_2	Z_2	Z_{12}	Z_2	Z_2	Z_3	Z_{15}	Z_2	Z_2^2	$Z_{12}\times Z_2$	$Z_{84}\times Z_2^2$	Z_2^2
S^3	0	0	Z	Z_2	Z_2	Z_{12}	Z_2	Z_2	Z_3	Z_{15}	Z_2	Z_2^2	$Z_{12}\times Z_2$	$Z_{84}\times Z_2^2$	Z_2^2
S^4	0	0	0	Z	Z_2	Z_2	$Z\times Z_{12}$	Z_2^2	Z_2^2	$Z_{24}\times Z_3$	Z_{15}	Z_2	Z_2^3	$Z_{120}\times Z_{12}\times Z_2$	$Z_{84}\times Z_2^5$
S^5	0	0	0	0	Z	Z_2	Z_2	Z_{24}	Z_2	Z_2	Z_2	Z_{30}	Z_2	Z_2^3	$Z_{72}\times Z_2$
S^6	0	0	0	0	0	Z	Z_2	Z_2	Z_{24}	0	Z	Z_2	Z_{60}	$Z_{24}\times Z_2$	Z_2^3
S^7	0	0	0	0	0	0	Z	Z_2	Z_2	Z_{24}	0	0	Z_2	Z_{120}	Z_2^3
S^8	0	0	0	0	0	0	0	Z	Z_2	Z_2	Z_{24}	0	0	Z_2	$Z\times Z_{120}$

[image from wikipedia]

K($\mathbb{Z}$,2)

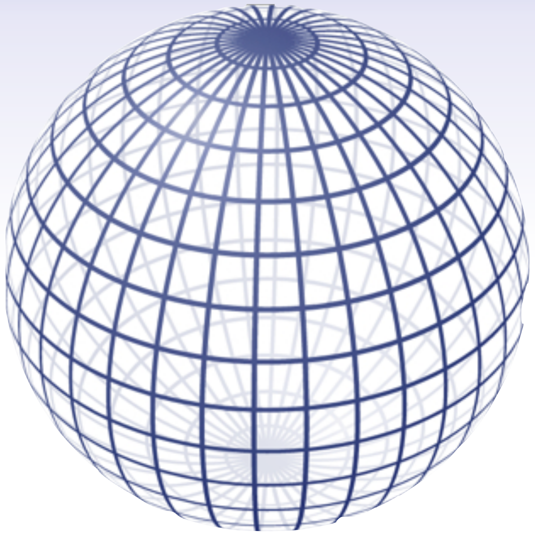
K($\mathbb{Z}$,2)

$K(\mathbb{Z},2)$



Define $K(\mathbb{Z},2) = ||S^2||_2$

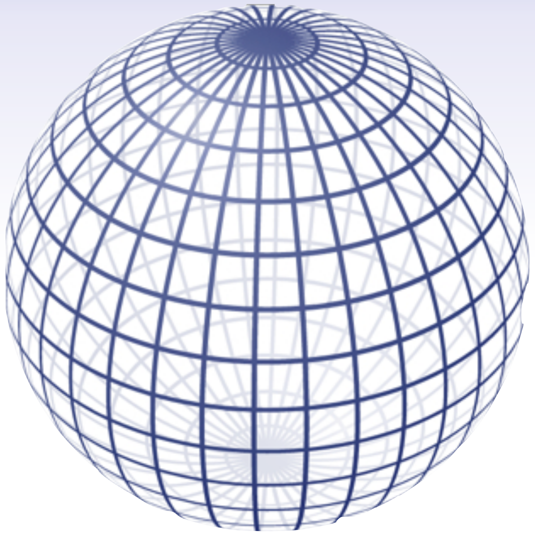
K($\mathbb{Z}$,2)



Define K($\mathbb{Z}$,2) = $\|S^2\|_2$

2-truncation = kill all paths at level higher than 2

$K(\mathbb{Z},2)$



Define $K(\mathbb{Z},2) = ||S^2||_2$

2-truncation = kill all paths at level higher than 2

- π_1 is trivial: same as S^2
- π_2 is $\mathbb{Z}$: same as S^2
- $\pi_{k>2}$ is trivial

$\pi_n(S^n)$

k^th homotopy group

n-dimensional sphere

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}	π_{14}	π_{15}
S^0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S^1	$\mathbf{Z}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S^2	0	$\mathbf{Z}$	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{12}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$
S^3	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{12}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$
S^4	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}\times\mathbf{Z}_{12}$	$\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{24}\times\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^3$	$\mathbf{Z}_{120}\times\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^5$
S^5	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{30}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^3$	$\mathbf{Z}_{72}\times\mathbf{Z}_2$
S^6	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_{60}$	$\mathbf{Z}_{24}\times\mathbf{Z}_2$	$\mathbf{Z}_2^3$
S^7	0	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	0	$\mathbf{Z}_2$	$\mathbf{Z}_{120}$	$\mathbf{Z}_2^3$
S^8	0	0	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	0	$\mathbf{Z}_2$	$\mathbf{Z}\times\mathbf{Z}_{120}$

[image from wikipedia]

K($\mathbb{Z}$,n)

Define K($\mathbb{Z}$,n) = $||S^n||_n$

- $\pi_{k<n}$ is trivial
- π_n is $\mathbb{Z}$
- $\pi_{k>n}$ is trivial

$K(\mathbb{Z},n)$

Define $K(\mathbb{Z},n) = ||S^n||_n$

- $\pi_{k<n}$ is trivial
- π_n is $\mathbb{Z}$
- $\pi_{k>n}$ is trivial

[HoTT proofs: L.,Brunerie,Lumsdaine]

K($\mathbb{Z}$,n)

Define K($\mathbb{Z}$,n) = $\|S^n\|_n$

- $\pi_{k<n}$ is trivial
- π_n is $\mathbb{Z}$
- $\pi_{k>n}$ is trivial

Generalize to other groups G?

[HoTT proofs: L.,Brunerie,Lumsdaine]

Suspension

ΣA is a higher inductive type generated by

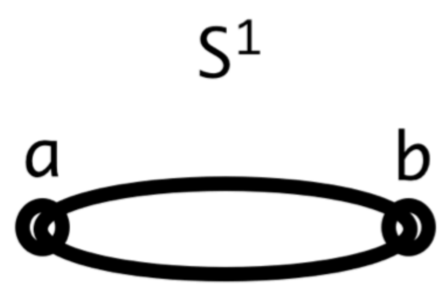
Suspension

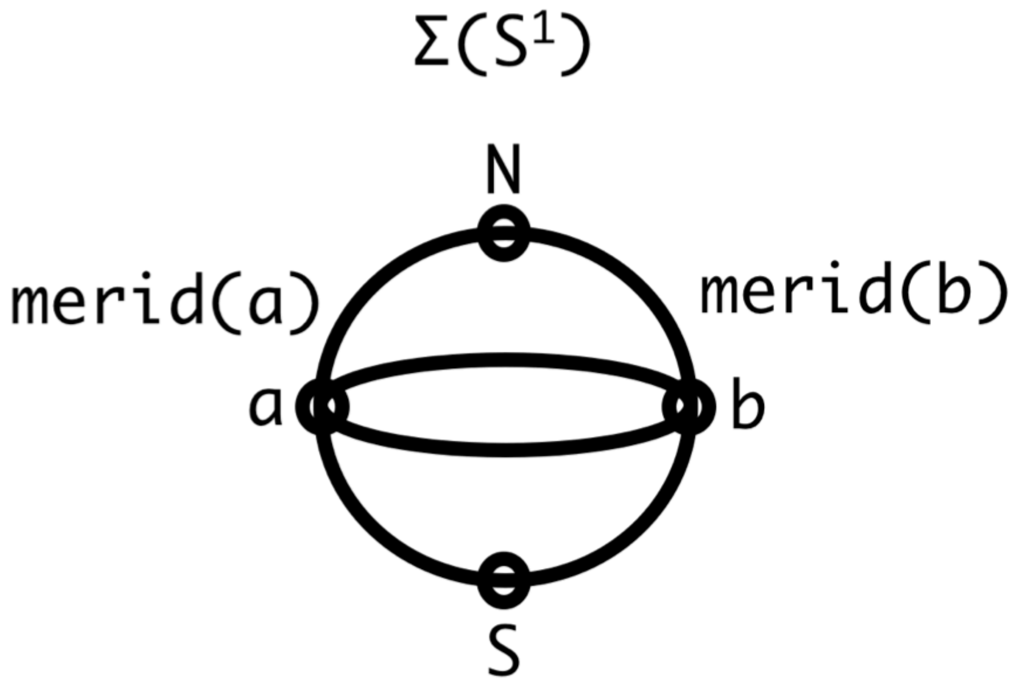
Σ A is a higher inductive type generated by

```
N : Σ A
S : Σ A
merid : A → Σ A
```

Spheres

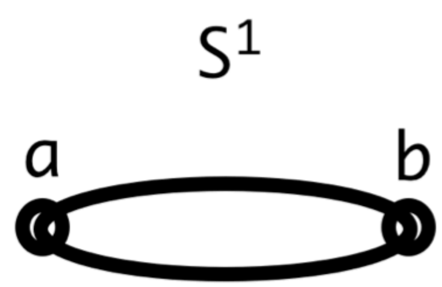
$S^2 = \Sigma\, S^1$

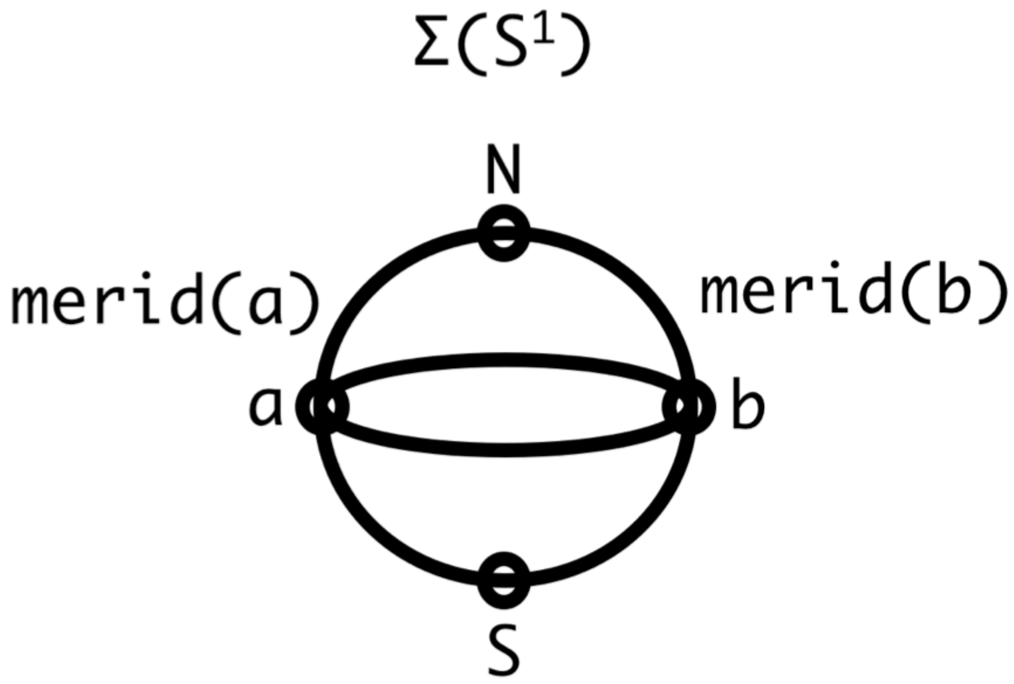




Spheres

$S^2 = \Sigma\ S^1$





$S^n = \Sigma^{n-1}\ S^1$

K(G,n)

Define $K(G,n) = ||\Sigma^{n-1} K(G,1)||_n$

Theorem:

- $\pi_{k<n}$ is trivial
- π_n is G
- $\pi_{k>n}$ is trivial

$\pi_n(K(G,n)) = G$

Define $K(G,n) = \|\Sigma^{n-1} K(G,1)\|_n$

$\pi_n(K(G,n)) = G$

Define $K(G,n) = ||\Sigma^{n-1} K(G,1)||_n$

$\pi_{n+1}(K(G,n+1)) = \pi_{n+1} ||\Sigma^n K(G,1)||_{n+1}$

$\pi_n(K(G,n)) = G$

Define $K(G,n) = \|\Sigma^{n-1} K(G,1)\|_n$

$$\pi_{n+1}(K(G,n+1)) = \pi_{n+1} \|\Sigma^n K(G,1)\|_{n+1}$$
$$= \pi_{n+1} \; \Sigma^n K(G,1)$$

$\pi_n(K(G,n)) = G$

Define $K(G,n) = ||\Sigma^{n-1} K(G,1)||_n$

$\pi_{n+1}(K(G,n+1)) = \pi_{n+1} ||\Sigma^n K(G,1)||_{n+1}$

$= \pi_{n+1} \ \Sigma^n K(G,1)$

$= \pi_n \ \Sigma^{n-1} K(G,1)$

$\pi_n(K(G,n)) = G$

Define $K(G,n) = ||\Sigma^{n-1} K(G,1)||_n$

$$\begin{aligned}\pi_{n+1}(K(G,n+1)) &= \pi_{n+1} ||\Sigma^n K(G,1)||_{n+1} \\ &= \pi_{n+1} \; \Sigma^n K(G,1) \\ &= \pi_n \; \Sigma^{n-1} K(G,1)\end{aligned}$$

Freudenthal suspension theorem; use HoTT proof by Lumsdaine

$\pi_n(K(G,n)) = G$

Define $K(G,n) = \|\Sigma^{n-1} K(G,1)\|_n$

$$\begin{aligned}
\pi_{n+1}(K(G,n+1)) &= \pi_{n+1} \|\Sigma^n K(G,1)\|_{n+1} \\
&= \pi_{n+1}\ \Sigma^n K(G,1) \\
&= \pi_n\ \Sigma^{n-1} K(G,1) \\
&= \pi_n\ \|\ \Sigma^{n-1} K(G,1)\ \|_n
\end{aligned}$$

Freudenthal suspension theorem; use HoTT proof by Lumsdaine

$\pi_n(K(G,n)) = G$

Define $K(G,n) = \|\Sigma^{n-1} K(G,1)\|_n$

$$\begin{aligned}
\pi_{n+1}(K(G,n+1)) &= \pi_{n+1} \|\Sigma^n K(G,1)\|_{n+1} \\
&= \pi_{n+1} \; \Sigma^n K(G,1) \\
&= \pi_n \; \Sigma^{n-1} K(G,1) \\
&= \pi_n \; \| \Sigma^{n-1} K(G,1) \|_n \\
&= \pi_n \; K(G,n)
\end{aligned}$$

Freudenthal suspension theorem; use HoTT proof by Lumsdaine

```
stable2 : π (k +1) (KG (n +1)) (base^ (n +1)) ≃ π k (KG n) (base^ n)
stable2 = π (k +1) (KG (n +1)) (base^ (n +1))              ≃⟨ (π<=Trunc (k +1) (n +1) (<=SCong lte) (FS.base'^ (n +1))) ⟩
          π (k +1) (Susp^ (S n -1pn) KG1) (FS.base'^ (n +1))  ≃⟨ ! (FS.Stable.stable k n (k<=n->k<=2n-2 k n indexing)) ⟩
          π k (Susp^ (n -1pn) KG1) (FS.base'^ n)              ≃⟨ ! (π<=Trunc k n lte (FS.base'^ n)) ⟩
          π k (KG n) (base^ n) ∎
```

Define $K(G,n) = \|\Sigma^{n-1} K(G,1)\|_n$

$$
\begin{aligned}
\pi_{n+1}(K(G,n+1)) &= \pi_{n+1} \|\Sigma^n K(G,1)\|_{n+1} \\
&= \pi_{n+1} \; \Sigma^n K(G,1) \\
&= \pi_n \; \Sigma^{n-1} K(G,1) \\
&= \pi_n \; \| \Sigma^{n-1} K(G,1) \|_n \\
&= \pi_n \; K(G,n)
\end{aligned}
$$

Freudenthal suspension theorem; use HoTT proof by Lumsdaine

```
stable2 : π (k +1) (KG (n +1)) (base^ (n +1)) ≃ π k (KG n) (base^ n)
stable2 = π (k +1) (KG (n +1)) (base^ (n +1))              ≃⟨ (π<=Trunc (k +1) (n +1) (<=SCong lte) (FS.base'^ (n +1))) ⟩
          π (k +1) (Susp^ (S n -1pn) KG1) (FS.base'^ (n +1))  ≃⟨ ! (FS.Stable.stable k n (k<=n->k<=2n-2 k n indexing)) ⟩
          π k (Susp^ (n -1pn) KG1) (FS.base'^ n)              ≃⟨ ! (π<=Trunc k n lte (FS.base'^ n)) ⟩
          π k (KG n) (base^ n) ∎
```

Define $K(G,n) = \|\Sigma^{n-1} K(G,1)\|_n$

$$\begin{aligned}
\pi_{n+1}(K(G,n+1)) &= \pi_{n+1} \|\Sigma^n K(G,1)\|_{n+1} \\
&= \pi_{n+1} \; \Sigma^n K(G,1) \\
&= \pi_n \; \Sigma^{n-1} K(G,1) \\
&= \pi_n \; \| \Sigma^{n-1} K(G,1) \|_n \\
&= \pi_n \; K(G,n)
\end{aligned}$$

merid : A → Ω (Σ A) is an equivalence?

Freudenthal suspension theorem; use HoTT proof by Lumsdaine

Freudenthal Suspension Thm

k^th homotopy group

n-dimensional sphere

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}	π_{14}	π_{15}
S^0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S^1	$\mathbf{Z}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S^2	0	$\mathbf{Z}$	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{12}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$
S^3	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{12}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$
S^4	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}\times\mathbf{Z}_{12}$	$\mathbf{Z}_2^2$	$\mathbf{Z}_2^2$	$\mathbf{Z}_{24}\times\mathbf{Z}_3$	$\mathbf{Z}_{15}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^3$	$\mathbf{Z}_{120}\times\mathbf{Z}_{12}\times\mathbf{Z}_2$	$\mathbf{Z}_{84}\times\mathbf{Z}_2^5$
S^5	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{30}$	$\mathbf{Z}_2$	$\mathbf{Z}_2^3$	$\mathbf{Z}_{72}\times\mathbf{Z}_2$
S^6	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_{60}$	$\mathbf{Z}_{24}\times\mathbf{Z}_2$	$\mathbf{Z}_2^3$
S^7	0	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	0	$\mathbf{Z}_2$	$\mathbf{Z}_{120}$	$\mathbf{Z}_2^3$
S^8	0	0	0	0	0	0	0	$\mathbf{Z}$	$\mathbf{Z}_2$	$\mathbf{Z}_2$	$\mathbf{Z}_{24}$	0	0	$\mathbf{Z}_2$	$\mathbf{Z}\times\mathbf{Z}_{120}$

[image from wikipedia]

Freudenthal

Theorem: If A is n-connected (trivial up to level n), then $||A||_{2n} = ||\Omega\ (\Sigma\ A)||_{2n}$

HoTT Proof [Lumsdaine]: generalization of encode-decode method

Freudenthal

Theorem: If A is n-connected (trivial up to level n), then $||A||_{2n} = ||\Omega\ (\Sigma\ A)||_{2n}$

HoTT Proof [Lumsdaine]: generalization of encode-decode method

Corollary:

$$\ldots = \pi_4(K(G,4)) = \pi_3(K(G,3)) = \pi_2(K(G,2))$$

$\pi_n(K(G,n)) = G$

By Freudenthal:

$\ldots = \pi_4(K(G,4)) = \pi_3(K(G,3) = \pi_2(K(G,2))$

$\pi_n(K(G,n)) = G$

By Freudenthal:

$\ldots = \pi_4(K(G,4)) = \pi_3(K(G,3) = \pi_2(K(G,2))$

By above:

$\pi_1(K(G,1)) = G$

$\pi_n(K(G,n)) = G$

By Freudenthal:

$\ldots = \pi_4(K(G,4)) = \pi_3(K(G,3) = \pi_2(K(G,2))$

By another encode-decode proof:

$\pi_2(K(G,2)) = \pi_1(K(G,1))$

By above:

$\pi_1(K(G,1)) = G$

Formalization

Formalization

* 10,000 line HoTT library

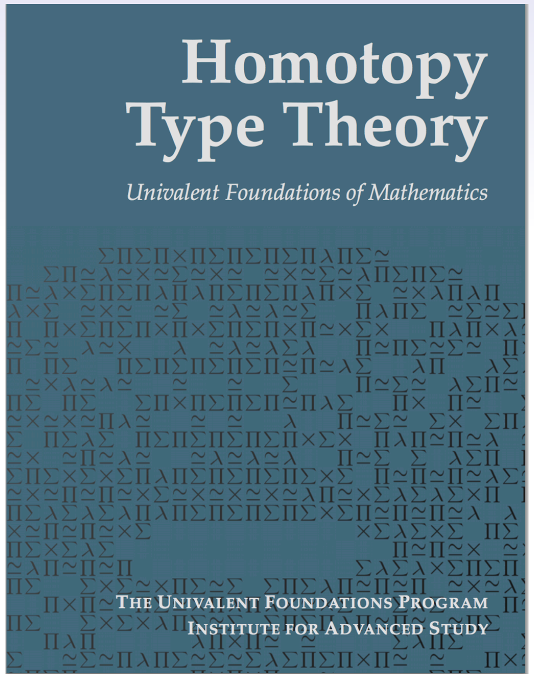
Formalization

- 10,000 line HoTT library
- + 250 lines for Freudenthal Suspension Theorem

Formalization

- 10,000 line HoTT library
- + 250 lines for Freudenthal Suspension Theorem
- + 750 lines for K(G,n)

Reading list



1. The HoTT Book

2. Homotopy theory in Agda:
   Fundamental group of the circle [LICS'13]
   $\pi_n(S^n) = \mathbb{Z}$ [CPP'13]
   K(G,n) [LICS'14]
   `github.com/dlicata335/`
   `github.com/hott/hott-agda`

3. Blog: `homotopytypetheory.org`