# Lambda-Calculus and Type Theory

ISR 2024
Obergurgl, Austria

Herman Geuvers & Niels van der Weide

Radboud University Nijmegen, The Netherlands

Lecture 13

Homotopy Type Theory

# Outline

# Equality of Mathematical Structures

When are two groups $G_1$ and $G_2$ the same?

# Equality of Mathematical Structures

When are two groups $G_1$ and $G_2$ the same?
Answer 1: **When they are equal**: $G_1 = G_2$.

- This is **proof irrelevant**: the proof carries no data
- If $G_1 = G_2$, then $G_1$ and $G_2$ have the same properties
- In foundations like ZFC: this is how groups are identified

# Equality of Mathematical Structures

When are two groups $G_1$ and $G_2$ the same?
Answer 1: **When they are equal**: $G_1 = G_2$.

- This is **proof irrelevant**: the proof carries no data
- If $G_1 = G_2$, then $G_1$ and $G_2$ have the same properties
- In foundations like ZFC: this is how groups are identified

Answer 2: **When they are isomorphic**: $G_1 \cong G_2$.

- This is **proof relevant**: the information is given by an isomorphism $G_1 \to G_2$
- We need to prove by hand: $G_1$ and $G_2$ have the same **group-theoretic properties**
- In practice: this is how we actually identify groups

# Equality of Mathematical Structures

When are two groups $G_1$ and $G_2$ the same?
Answer 1: **When they are equal**: $G_1 = G_2$.

- This is **proof irrelevant**: the proof carries no data
- If $G_1 = G_2$, then $G_1$ and $G_2$ have the same properties
- In foundations like ZFC: this is how groups are identified

Answer 2: **When they are isomorphic**: $G_1 \cong G_2$.

- This is **proof relevant**: the information is given by an isomorphism $G_1 \to G_2$
- We need to prove by hand: $G_1$ and $G_2$ have the same **group-theoretic properties**
- In practice: this is how we actually identify groups

Note the difference between how groups are identified in mathematical practice and in the foundations

# Why isomorphisms?

Common practice: **mathematical structures are identified up to isomorphism**

- Isomorphism is **independent of the representation**, while equality is not
- So: implementation details are hidden

However:

- Usual foundations for mathematics identifies structures up equality
- So: we have to prove by hand that properties are preserved under isomorphism
- In addition, only suitable properties are preserved under isomorphism

# Why isomorphisms?

Common practice: **mathematical structures are identified up to isomorphism**

- Isomorphism is **independent of the representation**, while equality is not
- So: implementation details are hidden

However:

- Usual foundations for mathematics identifies structures up equality
- So: we have to prove by hand that properties are preserved under isomorphism
- In addition, only suitable properties are preserved under isomorphism

**Can we have a foundation of mathematics where mathematical structures are identified up to isomorphism?**

# Homotopy type theory

- **Homotopy type theory** (HoTT) is a branch of type theory
- Key features: the **univalence axiom** and **higher inductive types** (HITs)
- Univalence axiom: allows us to identifies structures up to isomorphism
- HITs: give us access to quotients
- HoTT thinks about types in a different way: instead of viewing types as sets, we view them as **spaces**

# This lecture

This lecture gives a basic introduction to homotopy type theory
We discuss

- The identity type (the J-rule)
- Types as spaces
- The univalence axiom
- An example of a higher inductive type

# Outline

# The Identity Type

- Starting point of HoTT: the identity type
- So: what does it mean for two objects to be equal
- We shall start by discussing the rules for the identity type.

**Note: this part is not specific to HoTT**

# Rules for the Identity Type

**Formation Rule:**

$$\frac{\Gamma \vdash A : \text{Type} \qquad \Gamma \vdash x : A \qquad \Gamma \vdash y : A}{\Gamma \vdash x =_A y : \text{Type}}$$

If no confusion arises, we write $x = y$ instead of $x =_A y$

**Introduction Rule:**

$$\frac{\Gamma \vdash A : \text{Type} \qquad \Gamma \vdash x : A}{\Gamma \vdash \text{refl}_x : x = x}$$

# Elimination Rule for the Identity Type

**Elimination Rule (also known as the J-rule):**

$$\frac{\Gamma, x : A, y : A, p : x = y \vdash C : \mathrm{Type} \qquad \Gamma, x : A \vdash c : C[x := x, y := x, p := \mathrm{refl}_x] \qquad \Gamma \vdash p : x = y}{\Gamma \vdash \mathrm{J}(C, c, p) : C[x := x, y := y, p := p]}$$

We have $\mathrm{J}(C, c, \mathrm{refl}_x) \equiv c[x := x]$.
**Slogan**: to prove something for all $p : x = y$, it suffices to prove it for $\mathrm{refl}_x$ for all $x$.

# Elimination Rule for the Identity Type

**Elimination Rule (also known as the J-rule):**

$$\frac{\begin{array}{c}\Gamma, x : A, y : A, p : x = y \vdash C : \mathrm{Type} \\ \Gamma, x : A \vdash c : C[x := x, y := x, p := \mathrm{refl}_x] \\ \Gamma \vdash p : x = y\end{array}}{\Gamma \vdash \mathrm{J}(C, c, p) : C[x := x, y := y, p := p]}$$

We have $\mathrm{J}(C, c, \mathrm{refl}_x) \equiv c[x := x]$.
**Slogan**: to prove something for all $p : x = y$, it suffices to prove it for $\mathrm{refl}_x$ for all $x$.

# Consequences of the J-rule

With the J-rule, we can show

- **Symmetry**: given $x = y$, we have $y = x$

# Consequences of the J-rule

With the J-rule, we can show

- **Symmetry**: given $x = y$, we have $y = x$
- **Transitivity**: given $x = y$ and $y = z$, we have $x = z$

# Consequences of the J-rule

With the J-rule, we can show

- **Symmetry**: given $x = y$, we have $y = x$
- **Transitivity**: given $x = y$ and $y = z$, we have $x = z$
- **Congruence**: given $f : A \to B$ and $x =_A y$, we have $f\ x = f\ y$

# Consequences of the J-rule

With the J-rule, we can show

- **Symmetry**: given $x = y$, we have $y = x$
- **Transitivity**: given $x = y$ and $y = z$, we have $x = z$
- **Congruence**: given $f : A \to B$ and $x =_A y$, we have $f\ x = f\ y$
- **Substitution**: given a type family $B : A \to \mathrm{Type}$, $p : x =_A y$ and $\overline{x} : B(x)$, we have $p_*(\overline{x}) : B(y)$.

# Consequences of the J-rule

With the J-rule, we can show

- **Symmetry**: given $x = y$, we have $y = x$
- **Transitivity**: given $x = y$ and $y = z$, we have $x = z$
- **Congruence**: given $f : A \to B$ and $x =_A y$, we have $f\ x = f\ y$
- **Substitution**: given a type family $B : A \to \text{Type}$, $p : x =_A y$ and $\overline{x} : B(x)$, we have $p_*(\overline{x}) : B(y)$.

Each of these is proven in the same way. We only look at symmetry.

# Symmetry (formal)

**Goal**: given $p : x = y$, we have $y = x$

# Symmetry (formal)

**Goal**: given $p : x = y$, we have $y = x$
Recall the J-rule:

$$\frac{\Gamma, x : A, y : A, p : x = y \vdash C : \mathrm{Type} \qquad \Gamma, x : A \vdash c : C[x := x, y := x, p := \mathrm{refl}_x] \qquad \Gamma \vdash p : x = y}{\Gamma \vdash \mathrm{J}(C, c, p) : C[x := x, y := y, p := p]}$$

# Symmetry (formal)

**Goal**: given $p : x = y$, we have $y = x$
Recall the J-rule:

$$\frac{\Gamma, x : A, y : A, p : x = y \vdash C : \text{Type} \qquad \Gamma, x : A \vdash c : C[x := x, y := x, p := \text{refl}_x] \qquad \Gamma \vdash p : x = y}{\Gamma \vdash \text{J}(C, c, p) : C[x := x, y := y, p := p]}$$

Take

- $C \equiv y = x$
- We need $c : x = x$, for which we take $\text{refl}_x$

With this, we get

$$\text{J}(C, c, p) : y = x$$

# Symmetry (informal)

**Goal**: given $p : x = y$, we have $y = x$

- Assume that $p$ is reflexivity
- Then we must show $x = x$
- We use reflexivity

# Symmetry (in Coq)

```
Definition sym {A : Type} {x y : A} (p : x = y) : y = x.
Proof.
  induction p.
  reflexivity.
Defined.
```

# Iterating Identity Types

Note:

- The identity type is **polymorphic** in the type $A$
- So: given $p, q : x =_A y$, we also have a type $p =_{x =_A y} q$

We can iterate this as much as we want:

$$h =_{p =_{x =_A y} q} s$$

# Does the following principle hold?

- In mathematics, equality is a proposition
- We do not distinguish different proofs of equality in mathematics
- We can translate this into type theory: for all types $A$, terms $x, y : A$ and proofs $p, q : x =_A y$, we have $p =_{x=y} q$
- This principle known as **Unique of Identity Proofs (UIP)**

Does UIP hold in type theory?

# Does the following principle hold?

- In mathematics, equality is a proposition
- We do not distinguish different proofs of equality in mathematics
- We can translate this into type theory: for all types $A$, terms $x, y : A$ and proofs $p, q : x =_A y$, we have $p =_{x=y} q$
- This principle known as **Unique of Identity Proofs (UIP)**

Does UIP hold in type theory?
Well, not necessarily

# But what *is* a type?

It depends on how we **interpret types**

- If we interpret types as sets in set theory, then UIP holds
- However, there are other ways to interpret types in which UIP does not hold
- In such interpretation, other interesting principles might hold (like univalence)

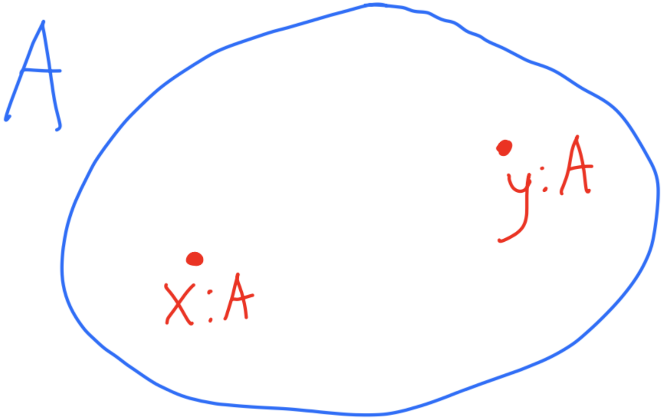
We shall look at an interpretation of types as topological spaces

# Outline

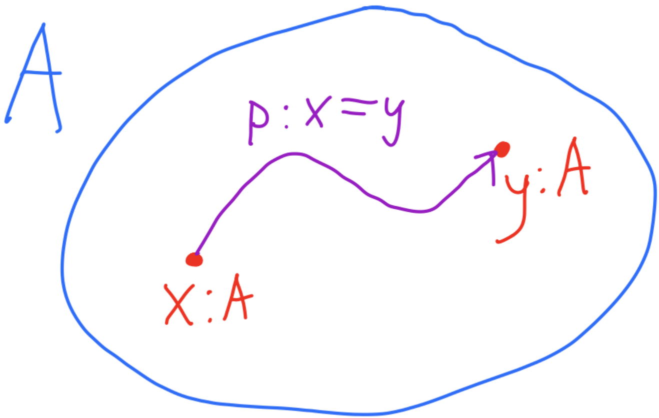
# Types and Topology

| Type Theory | Homotopy Theory |
| --- | --- |
| Types | Topological space |
| Dependent types | Fibrations |
| Terms | Points |
| Identity type | Paths |
| Identity of identities | Homotopies |

# Types and Terms

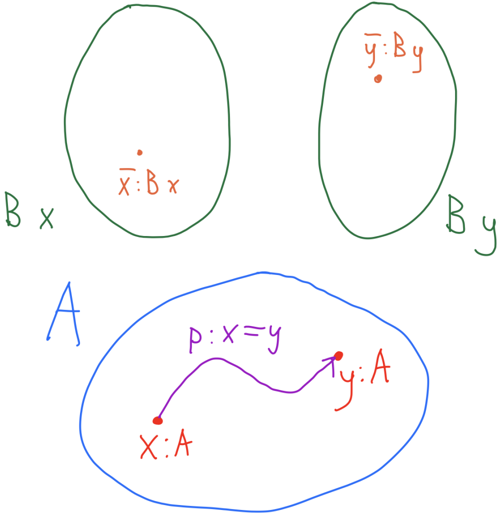


# Terms of the Identity Type

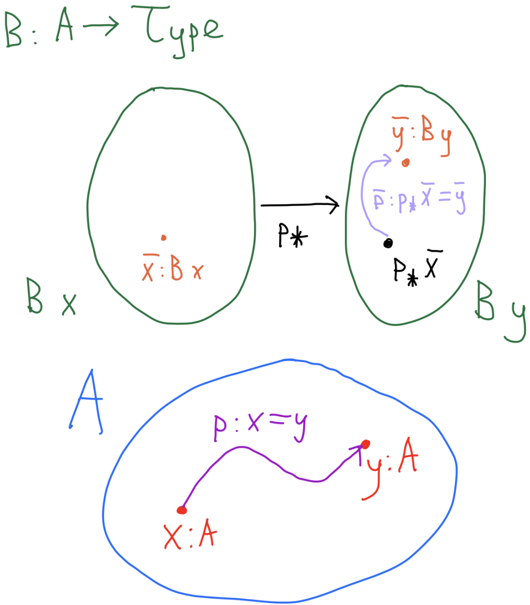


# Dependent Types

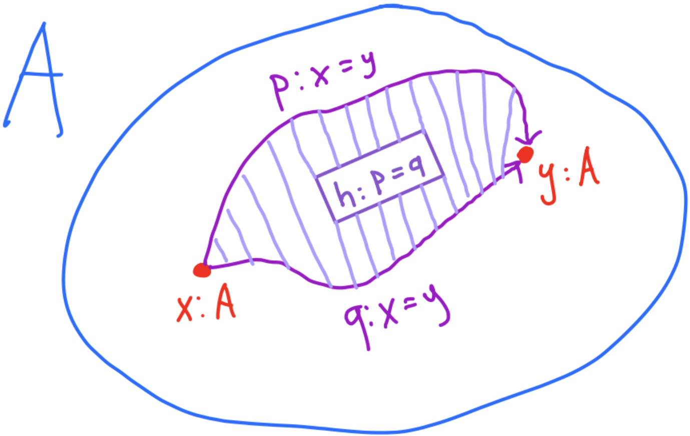
$B : A \to \mathrm{Type}$



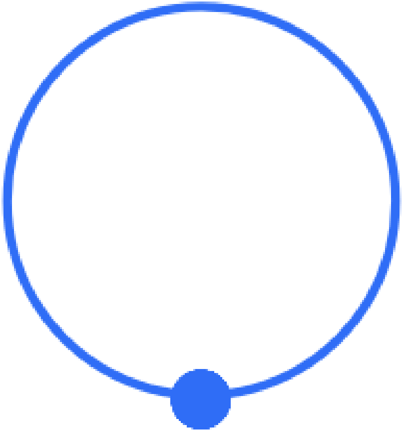
# Transport



# Homotopies



# UIP does not hold!



The circle cannot be filled.
So: **UIP does not hold**!

# Outline

# Proof Relevance of Identity

- From now on, we shall interpret types as spaces
- More specifically, we do not assume UIP
- As a consequence, statements like $p = q$ are not vacuously true for $p, q : x =_A y$

In this context, the J-rule is often referred to as **path induction**

# Computaton Rule for the Identity Type

Recall:

$$\frac{\begin{array}{c}\Gamma, x : A, y : A, p : x = y \vdash C : \mathrm{Type} \\ \Gamma, x : A \vdash c : C[x := x, y := x, p := \mathrm{refl}_x] \\ \Gamma \vdash p : x = y\end{array}}{\Gamma \vdash \mathrm{J}(C, c, p) : C[x := x, y := y, p := p]}$$

**Computation Rule:** We have $\mathrm{J}(C, c, \mathrm{refl}_x) \equiv c[x/x]$.

# Operations on the Identity Type

We have the following operations on the identity type

- **Inverse**: given $p : x = y$, we have $p^{-1} : y = x$ (*symmetry*)

# Operations on the Identity Type

We have the following operations on the identity type

- **Inverse**: given $p : x = y$, we have $p^{-1} : y = x$ (*symmetry*)
- **Concatenation**: given $p : x = y$ and $q : y = z$, we have $p \cdot q : x = z$ (*transitivity*)

# Operations on the Identity Type

We have the following operations on the identity type

- **Inverse**: given $p : x = y$, we have $p^{-1} : y = x$ (*symmetry*)
- **Concatenation**: given $p : x = y$ and $q : y = z$, we have $p \cdot q : x = z$ (*transitivity*)
- **Application**: given $f : A \to B$ and $p : x =_A y$, we have $\mathrm{ap}_f(p) : f\,x = f\,y$ (*congruence*)

# Operations on the Identity Type

We have the following operations on the identity type

- **Inverse**: given $p : x = y$, we have $p^{-1} : y = x$ (*symmetry*)
- **Concatenation**: given $p : x = y$ and $q : y = z$, we have $p \cdot q : x = z$ (*transitivity*)
- **Application**: given $f : A \to B$ and $p : x =_A y$, we have $\mathrm{ap}_f(p) : f\,x = f\,y$ (*congruence*)
- **Transport**: given a type family $B : A \to \mathrm{Type}$, $p : x =_A y$ and $\overline{x} : B(x)$, we have $p_*(\overline{x}) : B(y)$ (*substitution*).

# Operations on the Identity Type

We have the following operations on the identity type

- **Inverse**: given $p : x = y$, we have $p^{-1} : y = x$ (*symmetry*)
- **Concatenation**: given $p : x = y$ and $q : y = z$, we have $p \cdot q : x = z$ (*transitivity*)
- **Application**: given $f : A \to B$ and $p : x =_A y$, we have $\mathsf{ap}_f(p) : f\ x = f\ y$ (*congruence*)
- **Transport**: given a type family $B : A \to \mathsf{Type}$, $p : x =_A y$ and $\overline{x} : B(x)$, we have $p_*(\overline{x}) : B(y)$ (*substitution*).

**Reduction rules**

- $\mathsf{refl}_x^{-1} \equiv \mathsf{refl}_x$
- $\mathsf{refl}_x \cdot q \equiv q$
- $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f\ x}$
- $(\mathsf{refl}_x)_*(\overline{x}) \equiv \overline{x}$

# Laws for Operations on the Identity Type

We have the following equalities:

- $p \cdot \mathrm{refl}_y = p$
- $p \cdot (q \cdot r) = (p \cdot q) \cdot r$
- $p \cdot p^{-1} = \mathrm{refl}_x$
- $p^{-1} \cdot p = \mathrm{refl}_y$
- $\mathrm{ap}_f(p \cdot q) = \mathrm{ap}_f(p) \cdot \mathrm{ap}_f(q)$
- $(p \cdot q)_*(\overline{x}) = q_*(p_*(\overline{x}))$

Here $p : x = y$, $q : y = z$, and $r : z = a$.
These follow by the J-rule.

# Laws for Operations on the Identity Type

We have the following equalities:

- $p \cdot \mathrm{refl}_y = p$
- $p \cdot (q \cdot r) = (p \cdot q) \cdot r$
- $p \cdot p^{-1} = \mathrm{refl}_x$
- $p^{-1} \cdot p = \mathrm{refl}_y$
- $\mathrm{ap}_f(p \cdot q) = \mathrm{ap}_f(p) \cdot \mathrm{ap}_f(q)$
- $(p \cdot q)_*(\overline{x}) = q_*(p_*(\overline{x}))$

Here $p : x = y$, $q : y = z$, and $r : z = a$.
These follow by the J-rule.
We demonstrate this for $p \cdot \mathrm{refl}_y = p$ (right unitality).

# Right Unitality (formal)

**Goal**: given $p : x = y$, we have $p \cdot \mathsf{refl}_y = p$

# Right Unitality (formal)

**Goal**: given $p : x = y$, we have $p \cdot \mathrm{refl}_y = p$
Recall the J-rule:

$$\frac{\begin{array}{c}\Gamma, x : A, y : A, p : x = y \vdash C : \mathrm{Type} \\ \Gamma, x : A \vdash c : C[x := x, y := x, p := \mathrm{refl}_x] \\ \Gamma \vdash p : x = y\end{array}}{\Gamma \vdash \mathrm{J}(C, c, p) : C[x := x, y := y, p := p]}$$

# Right Unitality (formal)

**Goal**: given $p : x = y$, we have $p \cdot \mathsf{refl}_y = p$
Recall the J-rule:

$$\frac{\Gamma, x : A, y : A, p : x = y \vdash C : \mathsf{Type} \qquad \Gamma, x : A \vdash c : C[x := x, y := x, p := \mathsf{refl}_x] \qquad \Gamma \vdash p : x = y}{\Gamma \vdash \mathsf{J}(C, c, p) : C[x := x, y := y, p := p]}$$

Take

- $C \equiv p \cdot \mathsf{refl}_y = p$
- For all $x$, we need an inhabitant of $\mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x$
- Note: $\mathsf{refl}_x \cdot \mathsf{refl}_x$ reduces to $\mathsf{refl}_x$, so it holds by reflexivity

With this, we get

$$\mathsf{J}(C, c, p) : p \cdot \mathsf{refl}_y = p$$

# Right Unitality (informal)

**Goal**: given $p : x = y$, we have $p \cdot \mathsf{refl}_y = p$

- Assume that $p$ is reflexivity
- Then we must show $\mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x$
- Since $\mathsf{refl}_x \cdot \mathsf{refl}_x$ reduces to $\mathsf{refl}_x$, we can use reflexivity

# Right Unitality (informal)

**Goal**: given $p : x = y$, we have $p \cdot \text{refl}_y = p$

- Assume that $p$ is reflexivity
- Then we must show $\text{refl}_x \cdot \text{refl}_x = \text{refl}_x$
- Since $\text{refl}_x \cdot \text{refl}_x$ reduces to $\text{refl}_x$, we can use reflexivity

In Coq: again a matter of using the induction tactic.

# Outline

# Key Features Homotopy Type Theory

In **Homotopy Type Theory (HoTT)**, we view

- types as **spaces**
- terms as **points**
- identities as **paths**
- identities of identities as **homotopies**

HoTT also offers 2 new features to type theory:

- The **univalence axiom**
- **Higher inductive types**

# The Univalence Axiom

- Key feature of HoTT: **the univalence axiom**
- Intuition: two types are the same if they are isomorphism
- This is some kind of representation independence
- If you can prove two representations are equivalent, then they can be replaced by each other

**Note**: in HoTT, we say **equivalence** instead of isomorphism

# Equivalences

## Definition

Let $f : A \to B$ be a function.

- The **fiber** $\mathrm{fib}_f(y)$ of $f$ along $y : B$ is the type

$$\sum_{x:A} f(x) = y$$

- So: an inhabitant of $\mathrm{fib}_f(y)$ is a pair $x : A$ together with a path $f(x) = y$

# Equivalences

## Definition

Let $f : A \to B$ be a function.

- The **fiber** $\mathrm{fib}_f(y)$ of $f$ along $y : B$ is the type

$$\sum_{x:A} f(x) = y$$

- So: an inhabitant of $\mathrm{fib}_f(y)$ is a pair $x : A$ together with a path $f(x) = y$

## Definition

We say that $f$ is an **equivalence** if

- for all $y : B$ the type $\mathrm{fib}_f(y)$ is inhabited (*surjective*)
- all $x, y : \mathrm{fib}_f(y)$ are equal (*injective*)

The type $A \simeq B$ consists of maps $f : A \to B$ together with a proof that $f$ is an equivalence.

# The Univalence Axiom

### Proposition

*The identity map, which sends every $x$ to $x$, is an equivalence.*

### Proposition

*For all types* $A, B : \mathrm{Type}$*, we have a map* $\mathrm{idtoequiv} : A = B \to A \simeq B$*.*

### Axiom (Univalence Axiom)

*The map* $\mathrm{idtoequiv} : A = B \to A \simeq B$ *is an equivalence.*

Intuitively: $(A = B) = (A \simeq B)$

# UIP versus Univalence

Assuming univalence:

- There are two equivalences Bool $\simeq$ Bool
- So: there are two paths Bool $=$ Bool
- This contradicts UIP!

Univalence and UIP provide different perspectives on type theory

# What are higher inductive types?

**Higher inductive types** are an extension of **inductive types** where we can have constructors for **points**, **paths**, **homotopies**, and so on.

We can use higher inductive types to define:

- Topological spaces, like the circle or the interval
- Quotient types
- Free algebraic structures (free group, polynomial ring)

We shall only look at a simple example: the **interval**

# The Interval

```
Inductive interval : Type :=
| 0 : interval
| 1 : interval
| seg : 0 = 1.
```

**Note that Coq does not natively support higher inductive types**

# The Interval

```
Inductive interval : Type :=
| 0 : interval
| 1 : interval
| seg : 0 = 1.
```

**Note that Coq does not natively support higher inductive types**

What are the rules for the interval?

# The Interval

# The Introduction Rules

**Introduction Rules:**

$$\Gamma \vdash 0 : \text{interval}$$

$$\Gamma \vdash 1 : \text{interval}$$

$$\Gamma \vdash \text{seg} : 0 = 1$$

# The Recursion Rule

Before we do induction, let's do recursion

$$\frac{\Gamma \vdash A : \text{Type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p : a = b}{\Gamma \vdash \text{intRec}_{A,a,b,p} : \text{interval} \to A}$$

# The Recursion Rule

Before we do induction, let's do recursion

$$\frac{\Gamma \vdash A : \mathrm{Type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p : a = b}{\Gamma \vdash \mathrm{intRec}_{A,a,b,p} : \mathrm{interval} \to A}$$

**Computation rules**:

- $\mathrm{intRec}_{A,a,b,p}(0) = a$
- $\mathrm{intRec}_{A,a,b,p}(1) = b$
- $\mathrm{ap}_{\mathrm{intRec}_{A,a,b,p}}(\mathrm{seg}) = p$

# The Induction Principle

One might guess that the induction principle might be:

$$\frac{\begin{array}{c}\Gamma \vdash A : \text{interval} \to \text{Type} \\ \Gamma \vdash a : A(0) \\ \Gamma \vdash b : A(1) \\ \Gamma \vdash p : a = b\end{array}}{\Gamma \vdash \text{intRec}_{A,a,b,p} : \prod(x : \text{interval}), A(x)}$$

# The Induction Principle

One might guess that the induction principle might be:

$$\frac{\Gamma \vdash A : \text{interval} \to \text{Type} \quad \Gamma \vdash a : A(0) \quad \Gamma \vdash b : A(1) \quad \Gamma \vdash p : a = b}{\Gamma \vdash \text{intRec}_{A,a,b,p} : \prod(x : \text{interval}), A(x)}$$

However, **this does not type check!**, because *a* and *b* have a different type

# The Induction Principle

One might guess that the induction principle might be:

$$\frac{\Gamma \vdash A : \text{interval} \to \text{Type} \quad \Gamma \vdash a : A(0) \quad \Gamma \vdash b : A(1) \quad \Gamma \vdash p : a = b}{\Gamma \vdash \text{intRec}_{A,a,b,p} : \prod(x : \text{interval}), A(x)}$$

However, **this does not type check!**, because *a* and *b* have a different type
Solution: transport

# The Induction Principle

The induction principle for the interval:

$$\frac{\begin{array}{c}\Gamma \vdash A : \text{interval} \to \text{Type} \\ \Gamma \vdash a : A(0) \\ \Gamma \vdash b : A(1) \\ \Gamma \vdash p : \text{seg}_*(a) = b\end{array}}{\Gamma \vdash \text{intRec}_{A,a,b,p} : \prod(x : \text{interval}), A(x)}$$

# Outline

# More on HoTT

There are many interesting topics in homotopy type theory:

- Cubical type theory: how can we compute with univalence?
- Synthetic homotopy theory: develop algebraic topology in HoTT using that types represent spaces
- Univalent category theory: develop category theory from a univalent perspective
- Univalence and representation independence
- HITs allow us to define more data types, such as finite sets and finite multisets

# Summary

Main points of this lecture:

- The **identity type** and the **J-rule**
- Using the J-rule to define operations and proving laws for the identity types
- **Types as spaces**: this connects type theory and topology
- The **univalence axiom**: equality is equivalence
- **Higher inductive types**: defining data types with extra equalities