# Operations on identity types

Samuel Mimram

2025

École polytechnique

# Operations on identity types

We now see that we can define most of the expected operations involving identity types.

# Operations on identity types

We now see that we can define most of the expected operations involving identity types.

**Warning**
The naming of stuff is slightly different in the HoTT book and in the cubical Agda library.

I will use HoTT book notation in the slides and cubical Agda in the labs.

## Symmetry

Given a path $p : \mathsf{I} \to A$ there is a "symmetric path" $p^- : \mathsf{I} \to A$ defined by $p^-(t) = p(1 - t)$:

# Symmetry

Given a path $p : \mathsf{I} \to A$ there is a "symmetric path" $p^- : \mathsf{I} \to A$ defined by $p^-(t) = p(1 - t)$:


We have seen that we can define this in HoTT using $\mathsf{J}$:

$$\begin{aligned} \mathsf{sym} &: (x : A) \to (y : A) \to (x = y) \to (y = x) \\ x &\mapsto \mathsf{J}\; A\, x\, (\lambda yp.y = x)\; \mathsf{refl} \end{aligned}$$

# Symmetry

Given a path $p : \mathsf{I} \to A$ there is a "symmetric path" $p^{-} : \mathsf{I} \to A$ defined by $p^{-}(t) = p(1 - t)$:


We have seen that we can define this in HoTT using $\mathsf{J}$:

$$\mathsf{sym} : (x : A) \to (y : A) \to (x = y) \to (y = x)$$
$$x \mapsto \mathsf{J}\; A\, x\, (\lambda yp.y = x)\; \mathsf{refl}$$

More informally, in order to define $\mathsf{sym}$ on an arbitrary path $p : x = y$, by induction on $p$ it is enough to define it for $y \hat{=} x$ and $p \hat{=} \mathsf{refl}$, which we do using $\mathsf{refl}$.

# Symmetry

Given a path $p : \mathsf{I} \to A$ there is a "symmetric path" $p^- : \mathsf{I} \to A$ defined by $p^-(t) = p(1 - t)$:


We have seen that we can define this in HoTT using $\mathsf{J}$:

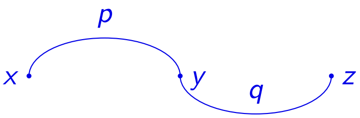
$$\begin{aligned} \mathsf{sym} &: (x : A) \to (y : A) \to (x = y) \to (y = x) \\ x &\mapsto \mathsf{J}\ A\, x\, (\lambda yp.y = x)\ \mathsf{refl} \end{aligned}$$

More informally, in order to define $\mathsf{sym}$ on an arbitrary path $p : x = y$, by induction on $p$ it is enough to define it for $y \triangleq x$ and $p \triangleq \mathsf{refl}$, which we do using $\mathsf{refl}$.

This proves that equality is symmetric.

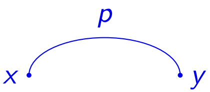
# Transitivity / concatenation

We can also show that equality is **transitive**, which amounts to concatenation of paths:



# Transitivity / concatenation

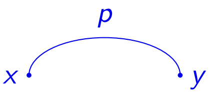
We can also show that equality is **transitive**, which amounts to concatenation of paths:




In order to define the concatenation $p \cdot q : x = z$ of paths $p : x = y$ and $q : y = z$, by induction on $q$ it is enough to define when $z \triangleq y$ and $q \triangleq \mathsf{refl} : y = y$, in which case we define it as $p$.

## Transitivity / concatenation

We can also show that equality is **transitive**, which amounts to concatenation of paths:



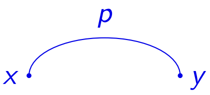

In order to define the concatenation $p \cdot q : x = z$ of paths $p : x = y$ and $q : y = z$, by induction on $q$ it is enough to define when $z \triangleq y$ and $q \triangleq \mathsf{refl} : y = y$, in which case we define it as $p$.

Geometrically, this corresponds to the concatenation of paths $p : \mathbf{I} \to A$ and $q : \mathbf{I} \to A$ defined by

$$(p \cdot q)(t) =$$

# Transitivity / concatenation

We can also show that equality is **transitive**, which amounts to concatenation of paths:




In order to define the concatenation $p \cdot q : x = z$ of paths $p : x = y$ and $q : y = z$, by induction on $q$ it is enough to define when $z \triangleq y$ and $q \triangleq \mathsf{refl} : y = y$, in which case we define it as $p$.

Geometrically, this corresponds to the concatenation of paths $p : \mathbb{I} \to A$ and $q : \mathbb{I} \to A$ defined by

$$(p \cdot q)(t) = \begin{cases} p(2t) & \text{if } 0 \leq t \leq 1/2 \\ q(2t-1) & \text{if } 1/2 \leq t \leq 2 \end{cases}$$

# Concatenation

Note that this is not the only way we could define concatenation:

- $p \cdot q \hat{=} ?$

# Concatenation

Note that this is not the only way we could define concatenation:

- $p \cdot \mathsf{refl} \triangleq p$

# Concatenation

Note that this is not the only way we could define concatenation:

- $p \cdot \mathrm{refl} \triangleq p$
- $p \cdot q \triangleq ?$

# Concatenation

Note that this is not the only way we could define concatenation:

- $p \cdot \mathsf{refl} \triangleq p$
- $\mathsf{refl} \cdot p \triangleq p$

# Concatenation

Note that this is not the only way we could define concatenation:

- $p \cdot \mathsf{refl} \triangleq p$
- $\mathsf{refl} \cdot p \triangleq p$
- $p \cdot q \triangleq ?$

# Concatenation

Note that this is not the only way we could define concatenation:

- $p \cdot \mathsf{refl} \doteq p$
- $\mathsf{refl} \cdot p \doteq p$
- $\mathsf{refl} \cdot \mathsf{refl} \doteq \mathsf{refl}$

# Concatenation

Note that this is not the only way we could define concatenation:

- $p \cdot \mathsf{refl} \triangleq p$
- $\mathsf{refl} \cdot p \triangleq p$
- $\mathsf{refl} \cdot \mathsf{refl} \triangleq \mathsf{refl}$

They are not definitionally equal, but they can be proved to be propositionally equal.

# Laws for concatenation

We have the following operation on paths: constant path (refl), concatenation ($p \cdot q$), symmetry ($p^-$).

# Laws for concatenation

We have the following operation on paths: constant path ($\mathsf{refl}$), concatenation ($p \cdot q$), symmetry ($p^{-}$).

**Proposition ([Uni13, Section 2.1])**
*The above operations satisfy the expected laws:*

# Laws for concatenation

We have the following operation on paths: constant path ($\mathsf{refl}$), concatenation ($p \cdot q$), symmetry ($p^{-}$).

**Proposition ([Uni13, Section 2.1])**
*The above operations satisfy the expected laws: for* $p : x = y$, $q : y = z$ *and* $r : z = w$,

$$\mathsf{refl} \cdot p = p \qquad (p \cdot q) \cdot r = p \cdot (q \cdot r) \qquad p \cdot p^{-} = \mathsf{refl} \qquad (p \cdot q)^{-} = q^{-} \cdot p^{-}$$

$$p \cdot \mathsf{refl} = p \qquad p^{-} \cdot p = \mathsf{refl} \qquad (p^{-})^{-} = p$$

## Laws for concatenation

We have the following operation on paths: constant path ($\mathsf{refl}$), concatenation ($p \cdot q$), symmetry ($p^-$).

**Proposition ([Uni13, Section 2.1])**
*The above operations satisfy the expected laws: for* $p : x = y$, $q : y = z$ *and* $r : z = w$,

$$\mathsf{refl} \cdot p = p \qquad (p \cdot q) \cdot r = p \cdot (q \cdot r) \qquad p \cdot p^- = \mathsf{refl} \qquad (p \cdot q)^- = q^- \cdot p^-$$

$$p \cdot \mathsf{refl} = p \qquad p^- \cdot p = \mathsf{refl} \qquad (p^-)^- = p$$

For instance, we have $p \cdot \mathsf{refl} = p$ by definition so that we can take $\mathsf{refl}$.

## Laws for concatenation

We have the following operation on paths: constant path ($\mathsf{refl}$), concatenation ($p \cdot q$), symmetry ($p^-$).

**Proposition ([Uni13, Section 2.1])**
*The above operations satisfy the expected laws: for $p : x = y$, $q : y = z$ and $r : z = w$,*

$$\mathsf{refl} \cdot p = p \qquad (p \cdot q) \cdot r = p \cdot (q \cdot r) \qquad p \cdot p^- = \mathsf{refl} \qquad (p \cdot q)^- = q^- \cdot p^-$$

$$p \cdot \mathsf{refl} = p \qquad p^- \cdot p = \mathsf{refl} \qquad (p^-)^- = p$$

For instance, we have $p \cdot \mathsf{refl} = p$ by definition so that we can take $\mathsf{refl}$.

To show $\mathsf{refl} \cdot p = p$, by induction on $p$ it is enough to show $\mathsf{refl} \cdot \mathsf{refl} = \mathsf{refl}$ which holds by previous point.

# Laws for concatenation

We have the following operation on paths: constant path ($\mathsf{refl}$), concatenation ($p \cdot q$), symmetry ($p^-$).

**Proposition ([Uni13, Section 2.1])**
*The above operations satisfy the expected laws: for* $p : x = y$, $q : y = z$ *and* $r : z = w$,

$$\mathsf{refl} \cdot p = p \qquad (p \cdot q) \cdot r = p \cdot (q \cdot r) \qquad p \cdot p^- = \mathsf{refl} \qquad (p \cdot q)^- = q^- \cdot p^-$$

$$p \cdot \mathsf{refl} = p \qquad p^- \cdot p = \mathsf{refl} \qquad (p^-)^- = p$$

For instance, we have $p \cdot \mathsf{refl} = p$ by definition so that we can take $\mathsf{refl}$.

To show $\mathsf{refl} \cdot p = p$, by induction on $p$ it is enough to show $\mathsf{refl} \cdot \mathsf{refl} = \mathsf{refl}$ which holds by previous point.

Other are proved similarly by induction on paths (see the lab).

# The fundamental $\infty$-groupoid of a space

It seems that this states that to any type/space we can associate a groupoid such that

- the objects are the points $x : A$,
- the morphisms $x \to y$ are the paths $p : x = y$,
- identities are given by $\mathsf{refl}$ and composition by concatenation,
- we have seen that the axioms are satisfied:

$$\mathsf{refl} \cdot p = p \qquad\qquad p \cdot \mathsf{refl} = p \qquad\qquad (p \cdot q) \cdot r = p \cdot (q \cdot r)$$

and we have inverses.

# The fundamental $\infty$-groupoid of a space

It seems that this states that to any type/space we can associate a groupoid such that

- the objects are the points $x : A$,
- the morphisms $x \to y$ are the paths $p : x = y$,
- identities are given by refl and composition by concatenation,
- we have seen that the axioms are satisfied:

$$\mathsf{refl} \cdot p = p \qquad p \cdot \mathsf{refl} = p \qquad (p \cdot q) \cdot r = p \cdot (q \cdot r)$$

  and we have inverses.

However this is not exactly the case because ...

# The fundamental $\infty$-groupoid of a space

It seems that this states that to any type/space we can associate a groupoid such that

- the objects are the points $x : A$,
- the morphisms $x \to y$ are the paths $p : x = y$,
- identities are given by refl and composition by concatenation,
- we have seen that the **axioms** are satisfied:

$$\mathsf{refl} \cdot p = p \qquad\qquad p \cdot \mathsf{refl} = p \qquad\qquad (p \cdot q) \cdot r = p \cdot (q \cdot r)$$

  and we have inverses.

However this is not exactly the case because "axioms" are homotopies!

# The fundamental $\infty$-groupoid of a space

Consider the space

$$A \quad \hat{=} \quad x \bullet\!\!-\!\!-\!\!-\!\!-\!\!-\!\!-\bullet\, y$$

we have a path $p : x = y$, i.e. a function $p : \mathsf{I} \to A$ which can be pictured as

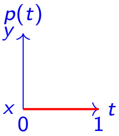

we also have $\mathsf{refl} : x = x$ pictured on the left.

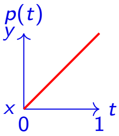
# The fundamental ∞-groupoid of a space

Consider the space

$$A \quad \triangleq \quad x \bullet\!\!\longrightarrow\!\!\bullet\, y$$

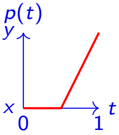
we have a path $p : x = y$, i.e. a function $p : \mathsf{I} \to A$ which can be pictured as




we also have $\mathsf{refl} : x = x$ pictured on the left.

Their concatenation is $\mathsf{refl} \cdot p$ :



Consider the space

$A \;\hat{=}$ 


we have a path $p : x = y$, i.e. a function $p : \mathsf{I} \to A$ which can be pictured as


we also have $\mathsf{refl} : x = x$ pictured on the left.

Their concatenation is $\mathsf{refl} \cdot p \sim p$:

# The fundamental $\infty$-groupoid of a space

A space induces an $\infty$-**groupoid** [Lum10, VDBG11]: we have

- 0-cells: the points of $x : A$
- 1-cells: the paths $p : x = y$ in $A$
- 2-cells: the homotopies $\alpha : p = q : x = y$ between paths in $A$
- ...

such that

- composition of $n$-cells is unital and associative *up to* $(n+1)$*-cells*
- the unitality and associativity satisfy coherence laws up to higher cells, etc.

$$\begin{array}{ccccc}
(((p \cdot q) \cdot r) \cdot s) & = & (p \cdot (q \cdot r)) \cdot s & & \\
 & & & \diagdown & \\
\| & & & & p \cdot ((q \cdot r) \cdot s) \\
 & & & & \| \\
(p \cdot q) \cdot (r \cdot s) & & = & & p \cdot (q \cdot (r \cdot s))
\end{array}$$

# Grothendieck's homotopy hypothesis

In fact, the $\infty$-groupoid is expected to contain all the relevant information of the space:

**Hypothesis**
The **Grothendieck homotopy hypothesis** [Gro83] states that spaces should be equivalent to $\infty$-groupoids.

Note: we would have to detail what we mean by "space", by "$\infty$-groupoid" and by "equivalent", which is out of the scope of this course. There are various answers for that, and the hypothesis has been proved for some of them.

# Congruence

An important property of equality is that it is a **congruence**:

**Proposition**
*Given a function* $f : A \to B$ *and an equality* $p : x = y$ *in* $A$*, we have an equality* $f(x) = f(y)$*.*

**Proof.**
By induction on $p$, it is enough to show that we have $f(x) = f(x)$, done by $\mathsf{refl}$. □

# Congruence

An important property of equality is that it is a **congruence**:

**Proposition**
*Given a function $f : A \to B$ and an equality $p : x = y$ in $A$, we have an equality $f(x) = f(y)$.*

**Proof.**
By induction on $p$, it is enough to show that we have $f(x) = f(x)$, done by $\mathsf{refl}$. □

We therefore have a function

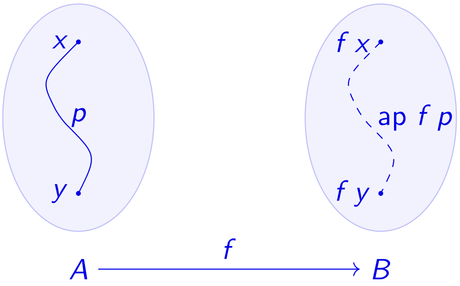
$$\mathsf{ap} : (A \to B) \to \{x\,y : A\} \to (x = y) \to (f(x) = f(y))$$

also named `cong` in Agda, which can be read as

- we can **apply** a function to a path,
- all functions induce **functors** between the corresponding $\infty$-groupoids.

# Congruence: continuity

Geometrically, ap also means that every function is **continuous**:



**Proposition ([Uni13, Lemma 2.2.2])**
*We have that:*

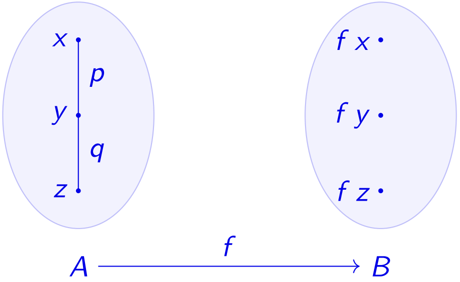
- ap *is compatible with the groupoid structure:*

# Congruence: properties

**Proposition ([Uni13, Lemma 2.2.2])**
*We have that:*

- ap *is compatible with the groupoid structure:*

$$\mathsf{ap}\ f\ (p \cdot q) = (\mathsf{ap}\ f\ p) \cdot (\mathsf{ap}\ f\ q) \qquad \mathsf{ap}\ f\ \mathsf{refl} = \mathsf{refl} \qquad \mathsf{ap}\ f\ p^{-} = (\mathsf{ap}\ f\ p)^{-}$$

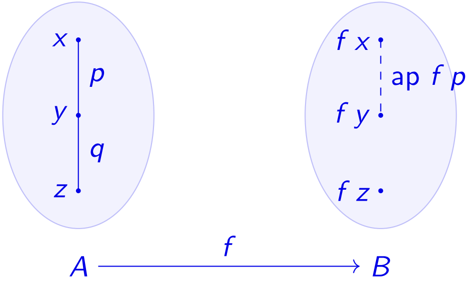


# Congruence: properties

**Proposition ([Uni13, Lemma 2.2.2])**
*We have that:*

- ap *is compatible with the groupoid structure:*

$$\mathsf{ap}\ f\ (p \cdot q) = (\mathsf{ap}\ f\ p) \cdot (\mathsf{ap}\ f\ q) \qquad \mathsf{ap}\ f\ \mathsf{refl} = \mathsf{refl} \qquad \mathsf{ap}\ f\ p^{-} = (\mathsf{ap}\ f\ p)^{-}$$

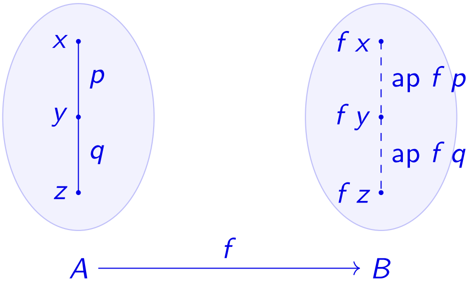


# Congruence: properties

**Proposition ([Uni13, Lemma 2.2.2])**
*We have that:*

- ap *is compatible with the groupoid structure:*

$$\text{ap}\ f\ (p \cdot q) = (\text{ap}\ f\ p) \cdot (\text{ap}\ f\ q) \qquad \text{ap}\ f\ \text{refl} = \text{refl} \qquad \text{ap}\ f\ p^{-} = (\text{ap}\ f\ p)^{-}$$

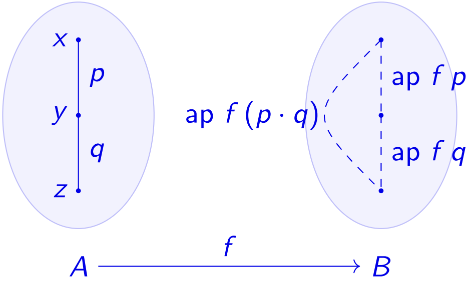


# Congruence: properties

**Proposition ([Uni13, Lemma 2.2.2])**
*We have that:*

- ap *is compatible with the groupoid structure:*

$$\mathsf{ap}\ f\,(p \cdot q) = (\mathsf{ap}\ f\ p) \cdot (\mathsf{ap}\ f\ q) \qquad \mathsf{ap}\ f\ \mathsf{refl} = \mathsf{refl} \qquad \mathsf{ap}\ f\ p^{-} = (\mathsf{ap}\ f\ p)^{-}$$

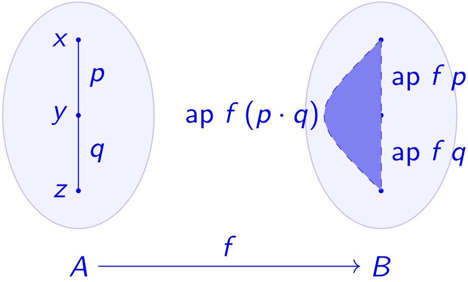


# Congruence: properties

**Proposition ([Uni13, Lemma 2.2.2])**
*We have that:*

- ap *is compatible with the groupoid structure:*

$$\text{ap } f\,(p \cdot q) = (\text{ap } f\ p) \cdot (\text{ap } f\ q) \qquad \text{ap } f\ \text{refl} = \text{refl} \qquad \text{ap } f\ p^{-} = (\text{ap } f\ p)^{-}$$



# Congruence: properties

**Proposition ([Uni13, Lemma 2.2.2])**
*We have that:*

- $\mathsf{ap}$ *is compatible with the groupoid structure:*

$$\mathsf{ap}\ f\ (p \cdot q) = (\mathsf{ap}\ f\ p) \cdot (\mathsf{ap}\ f\ q) \qquad \mathsf{ap}\ f\ \mathsf{refl} = \mathsf{refl} \qquad \mathsf{ap}\ f\ p^{-} = (\mathsf{ap}\ f\ p)^{-}$$

- $\mathsf{ap}$ *is compatible with composition:*

## Congruence: properties

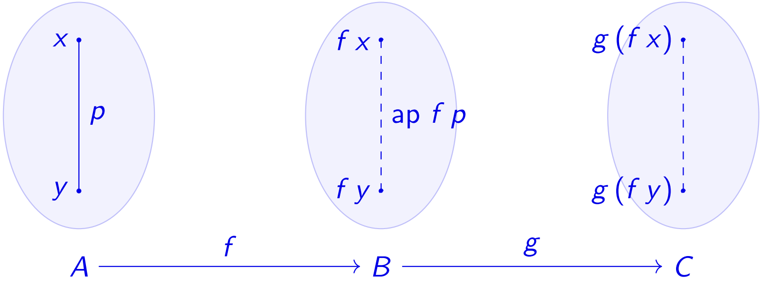
**Proposition ([Uni13, Lemma 2.2.2])**
*We have that:*

- ap *is compatible with the groupoid structure:*

$$\mathsf{ap}\ f\ (p \cdot q) = (\mathsf{ap}\ f\ p) \cdot (\mathsf{ap}\ f\ q) \qquad \mathsf{ap}\ f\ \mathsf{refl} = \mathsf{refl} \qquad \mathsf{ap}\ f\ p^{-} = (\mathsf{ap}\ f\ p)^{-}$$

- ap *is compatible with composition:*

$$\mathsf{ap}\,(g \circ f)\,p = \mathsf{ap}\ g\,(\mathsf{ap}\ f\ p) \qquad \mathsf{ap}\ \mathsf{id}\ p = p$$



# Substitutivity

An important property of equality is that it is **substitutive**:
given a property $P : A \to \mathcal{U}$, if $x$ satisfies $P$ and $x = y$ then $y$ also satisfies $P$.

# Substitutivity

An important property of equality is that it is **substitutive**:
given a property $P : A \to \mathcal{U}$, if $x$ satisfies $P$ and $x = y$ then $y$ also satisfies $P$.

Ex: in $\mathbb{Q}$, we have that $4/2$ is an integer and $4/2 = 6/3$ therefore $6/3$ is also an integer.

# Substitutivity

An important property of equality is that it is **substitutive**:
given a property $P : A \to \mathcal{U}$, if $x$ satisfies $P$ and $x = y$ then $y$ also satisfies $P$.

Ex: in $\mathbb{Q}$, we have that $4/2$ is an integer and $4/2 = 6/3$ therefore $6/3$ is also an integer.

**Proposition**
*We have a function called* ***transport*** *or* `subst` *in Agda:*

$$\text{transport} : \{A : \mathcal{U}\} \to (P : A \to \mathcal{U}) \to \{x\, y : A\} \to (x = y) \to P\, x \to P\, y$$

**Proof.**
By induction, it is enough to provide a function $P\, x \to P\, x$ and we take $\text{id}$. □

# Difference between booleans

**Proposition**
*In* Bool*, we have* $\neg(\mathit{false} = \mathit{true})$*.*

# Difference between booleans

**Proposition**
*In* Bool*, we have* $\neg(\mathit{false} = \mathit{true})$*.*

**Proof.**
Suppose given $p : \text{false} = \text{true}$.

# Difference between booleans

**Proposition**
*In* Bool*, we have* $\neg(\mathit{false} = \mathit{true})$*.*

**Proof.**
Suppose given $p : \mathsf{false} = \mathsf{true}$. Consider the function

$$\begin{aligned} F : \mathsf{Bool} &\to \mathcal{U} \\ \mathsf{false} &\mapsto 1 \\ \mathsf{true} &\mapsto \bot \end{aligned}$$

## Difference between booleans

**Proposition**
*In* Bool*, we have* $\neg(\mathit{false} = \mathit{true})$*.*

**Proof.**
Suppose given $p : \mathsf{false} = \mathsf{true}$. Consider the function

$$\begin{aligned} F : \mathsf{Bool} &\to \mathcal{U} \\ \mathsf{false} &\mapsto 1 \\ \mathsf{true} &\mapsto \bot \end{aligned}$$

By transport, we have

$$\mathsf{transport}\ F\ p \ : F\,\mathsf{false} \to F\,\mathsf{true}$$

## Difference between booleans

**Proposition**
*In* Bool*, we have* $\neg(\mathit{false} = \mathit{true})$*.*

**Proof.**
Suppose given $p : \mathsf{false} = \mathsf{true}$. Consider the function

$$\begin{aligned} F : \mathsf{Bool} &\to \mathcal{U} \\ \mathsf{false} &\mapsto 1 \\ \mathsf{true} &\mapsto \bot \end{aligned}$$

By transport, we have

$$\mathsf{transport}\ F\ p\ : 1 \to \bot$$

# Difference between booleans

**Proposition**
*In Bool, we have $\neg(\mathit{false} = \mathit{true})$.*

**Proof.**
Suppose given $p : \mathsf{false} = \mathsf{true}$. Consider the function

$$\begin{aligned} F : \mathsf{Bool} &\to \mathcal{U} \\ \mathsf{false} &\mapsto 1 \\ \mathsf{true} &\mapsto \bot \end{aligned}$$

By transport, we have

$$\mathsf{transport}\ F\ p : 1 \to \bot$$

We thus have $\bot$ by applying it to $\star : 1$. □

# Leibniz equality

This property of **indiscernability of identicals** can be taken as the definition of **Leibniz equality** [Lei86]: on $A$, we define

$$(x \overset{\mathsf{L}}{=} y) \quad \hat{=} \quad ((P : A \to \mathcal{U}) \to P\,x \to P\,y)$$

# Leibniz equality

This property of **indiscernability of identicals** can be taken as the definition of **Leibniz equality** [Lei86]: on $A$, we define

$$(x \overset{\mathsf{L}}{=} y) \quad \hat{=} \quad ((P : A \to \mathcal{U}) \to P\,x \to P\,y)$$

**Lemma**
*This is a symmetric relation.*

**Proof.**
Suppose $x \overset{\mathsf{L}}{=} y$. Given $P : A \to \mathcal{U}$ such that $P\,y$, we have to show $P\,x$.

# Leibniz equality

This property of **indiscernability of identicals** can be taken as the definition of **Leibniz equality** [Lei86]: on $A$, we define

$$(x \stackrel{\mathsf{L}}{=} y) \quad \hat{=} \quad ((P : A \to \mathcal{U}) \to P\,x \to P\,y)$$

**Lemma**
*This is a symmetric relation.*

**Proof.**
Suppose $x \stackrel{\mathsf{L}}{=} y$. Given $P : A \to \mathcal{U}$ such that $P\,y$, we have to show $P\,x$.

Consider the property $Q(y) \mathrel{\hat{=}} P\,y \to P\,x$.

# Leibniz equality

This property of **indiscernability of identicals** can be taken as the definition of **Leibniz equality** [Lei86]: on $A$, we define

$$(x \stackrel{\mathsf{L}}{=} y) \quad \hat{=} \quad ((P : A \to \mathcal{U}) \to P\,x \to P\,y)$$

**Lemma**
*This is a symmetric relation.*

**Proof.**
Suppose $x \stackrel{\mathsf{L}}{=} y$. Given $P : A \to \mathcal{U}$ such that $P\,y$, we have to show $P\,x$.

Consider the property $Q(y) \mathrel{\hat{=}} P\,y \to P\,x$. We have $Q(x) \mathrel{\hat{=}} P\,x \to P\,y$ by $\mathsf{id}$,

# Leibniz equality

This property of **indiscernability of identicals** can be taken as the definition of **Leibniz equality** [Lei86]: on $A$, we define

$$(x \overset{\mathsf{L}}{=} y) \quad \hat{=} \quad ((P : A \to \mathcal{U}) \to P\,x \to P\,y)$$

**Lemma**
*This is a symmetric relation.*

**Proof.**
Suppose $x \overset{\mathsf{L}}{=} y$. Given $P : A \to \mathcal{U}$ such that $P\,y$, we have to show $P\,x$.

Consider the property $Q(y) \hat{=} P\,y \to P\,x$. We have $Q(x) \hat{=} P\,x \to P\,y$ by $\mathsf{id}$, therefore $Q(y) \hat{=} P\,y \to P\,x$ because $x \overset{\mathsf{L}}{=} y$,

# Leibniz equality

This property of **indiscernability of identicals** can be taken as the definition of **Leibniz equality** [Lei86]: on $A$, we define

$$(x \overset{\mathsf{L}}{=} y) \quad \hat{=} \quad ((P : A \to \mathcal{U}) \to P\,x \to P\,y)$$

**Lemma**

*This is a symmetric relation.*

**Proof.**

Suppose $x \overset{\mathsf{L}}{=} y$. Given $P : A \to \mathcal{U}$ such that $P\,y$, we have to show $P\,x$.

Consider the property $Q(y) \mathrel{\hat{=}} P\,y \to P\,x$. We have $Q(x) \mathrel{\hat{=}} P\,x \to P\,y$ by $\mathsf{id}$, therefore $Q(y) \mathrel{\hat{=}} P\,y \to P\,x$ because $x \overset{\mathsf{L}}{=} y$, and we deduce $P\,x$ since we have $P\,y$. □

# Leibniz equality

This property of **indiscernability of identicals** can be taken as the definition of **Leibniz equality** [Lei86]: on $A$, we define

$$(x \overset{\mathsf{L}}{=} y) \quad \hat{=} \quad ((P : A \to \mathcal{U}) \to P\,x \to P\,y)$$

**Lemma**
*This is a symmetric relation.*

**Proof.**
Suppose $x \overset{\mathsf{L}}{=} y$. Given $P : A \to \mathcal{U}$ such that $P\,y$, we have to show $P\,x$.

Consider the property $Q(y) \mathrel{\hat{=}} P\,y \to P\,x$. We have $Q(x) \mathrel{\hat{=}} P\,x \to P\,y$ by $\mathsf{id}$, therefore $Q(y) \mathrel{\hat{=}} P\,y \to P\,x$ because $x \overset{\mathsf{L}}{=} y$, and we deduce $P\,x$ since we have $P\,y$. □

In fact, Leibniz equality is logically equivalent to identity [ACD+20]:

$$(x \overset{\mathsf{L}}{=} y) \leftrightarrow (x = y)$$

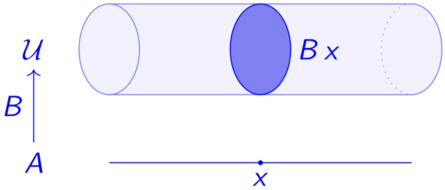
Let's provide a geometric interpretation for transport.

# Type families

A **type family** is a function

$$B : A \to \mathcal{U}$$

which can be thought of as a family of spaces $B\,x$ continuously indexed by $x : A$

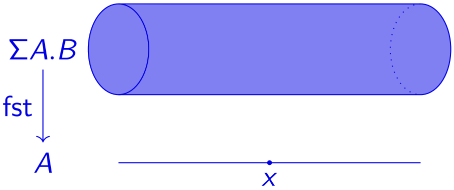


# Type families

A **type family** is a function

$$B : A \to \mathcal{U}$$

which can be thought of as a family of spaces $B\,x$ continuously indexed by $x : A$



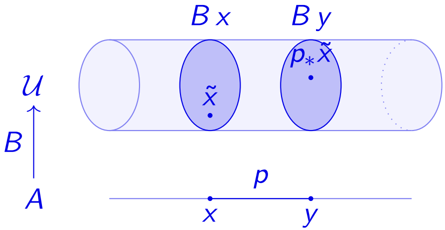

and $\Sigma(x : A).B\,x$ is the **total space**.

# Transport

Given a type family $B : A \to \mathcal{U}$ the **transport** (or `subst`) operation associates to a path $p : x = y$ in $A$ a function

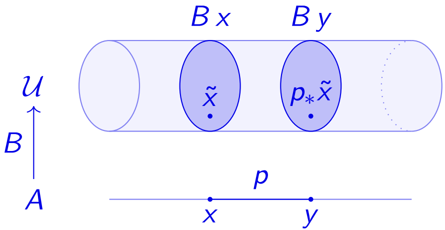
$$p_* : B\,x \to B\,y$$

which can be pictured as



# Transport

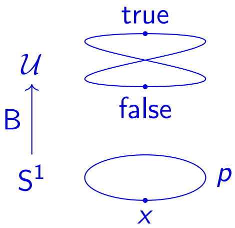
By ap all the fibers have to be equal when $A$ is connected, but the transport can still be non-trivial!



# Transport

By ap all the fibers have to be equal when $A$ is connected, but the transport can still be non-trivial!

For instance, consider the non-trivial fibration $B : \mathsf{S}^1 \to \mathcal{U}$ with $B\,x \triangleq \mathsf{Bool}$.




We have

$$p_*\,\mathsf{false} = \mathsf{true}$$

# Transport: properties

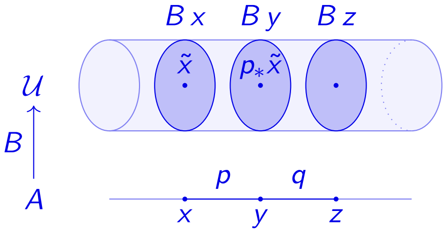
Transport satisfies the expected properties:

# Transport: properties

Transport satisfies the expected properties:

**Proposition ([Uni13, Lemma 2.3.9])**
*For $p : x = y$ and $q : y = z$ in $A$, and $\tilde{x} : B\,x$, we have*

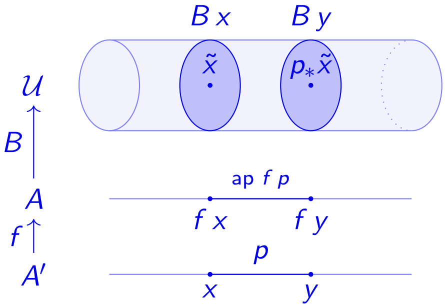
$$(p \cdot q)_* \, \tilde{x} = q_*(p_* \, \tilde{x})$$



# Transport: properties

**Proposition ([Uni13, Lemma 2.3.10])**
*Given* $f : A' \to A$, $B : A \to \mathcal{U}$, $p : x = y$ *in* $A'$ *and* $\tilde{x} : B\,x$,

$$\text{transport}\,(B \circ f)\,p\,\tilde{x} = \text{transport}\ B\,(\text{ap}\ f\ p)\,\tilde{x}$$



# Transport: a variant

We have the following variant of transport

$$\mathsf{transport} : (B : A \to \mathcal{U}) \to \{x\, y : A\} \to (x = y) \to B\, x \to B\, y$$

sometimes called $\mathsf{coe}$ for **coercion** and noted `transport` in Agda:

$$\mathsf{coe} : A = B \to A \to B$$

# Transport: a variant

We have the following variant of transport

$$\mathsf{transport} : (B : A \to \mathcal{U}) \to \{x\,y : A\} \to (x = y) \to B\,x \to B\,y$$

sometimes called coe for **coercion** and noted `transport` in Agda:

$$\mathsf{coe} : A = B \to A \to B$$

**Proposition**
*The functions* transport *and* coe *are logically equivalent.*

**Proof.**
We have

# Transport: a variant

We have the following variant of transport

$$\text{transport} : (B : A \to \mathcal{U}) \to \{x\, y : A\} \to (x = y) \to B\, x \to B\, y$$

sometimes called coe for **coercion** and noted `transport` in Agda:

$$\text{coe} : A = B \to A \to B$$

**Proposition**
*The functions* transport *and* coe *are logically equivalent.*

**Proof.**
We have

$$\text{coe } p\, x = \text{transport}\, (\lambda X.X)\, p\, x$$

$$\text{transport } b\, p\, \tilde{x} = \text{coe}\, (\text{ap } B\, p)\, \tilde{x}$$

□

# Transport: a variant

We have the following variant of transport

$$\text{transport} : (B : A \to \mathcal{U}) \to \{x\,y : A\} \to (x = y) \to B\,x \to B\,y$$

sometimes called coe for **coercion** and noted `transport` in Agda:

$$\text{coe} : A = B \to A \to B$$

**Proposition**
*The functions* transport *and* coe *are logically equivalent.*

**Proof.**
We have

$$\text{coe}\ p\,x = \text{transport}\,(\lambda X.X)\,p\,x$$

$$\text{transport}\ b\,p\,\tilde{x} = \text{coe}\,(\text{ap}\ B\ p)\,\tilde{x}$$

□

# Type families are fibrations

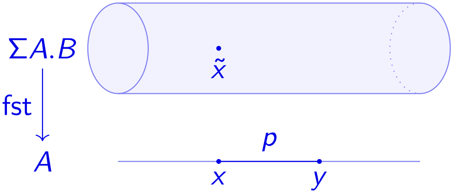
**Proposition ([Uni13, Lemma 2.3.2])**
*Consider a type family* $B : A \to \mathcal{U}$*, the map*

$$\mathsf{fst} : \Sigma(x : A).B\,x \to A$$

*is a* ***fibration****: given a path* $p : x = y$ *and* $\tilde{x} : B\,x$*, there is a path*

$$\tilde{p} : (x, \tilde{x}) = (y, p_*\,\tilde{x})$$

*such that* $\mathsf{ap}\ \mathsf{fst}\ \tilde{p} = p$.



# Type families are fibrations

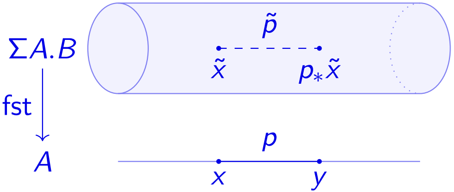
**Proposition ([Uni13, Lemma 2.3.2])**
*Consider a type family* $B : A \to \mathcal{U}$*, the map*

$$\mathsf{fst} : \Sigma(x : A).B\,x \to A$$

*is a* ***fibration****: given a path* $p : x = y$ *and* $\tilde{x} : B\,x$*, there is a path*

$$\tilde{p} : (x, \tilde{x}) = (y, p_*\,\tilde{x})$$

*such that* $\mathsf{ap}\ \mathsf{fst}\ \tilde{p} = p$*.*



# Type families are fibrations

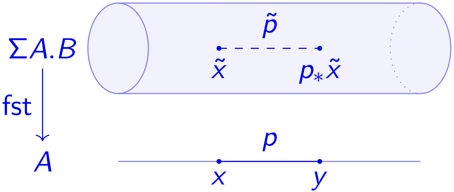
**Proposition ([Uni13, Lemma 2.3.2])**
*Consider a type family $B : A \to \mathcal{U}$, the map*

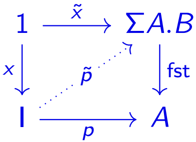
$$\mathsf{fst} : \Sigma(x : A).B\,x \to A$$

*is a **fibration**: given a path $p : x = y$ and $\tilde{x} : B\,x$, there is a path*

$$\tilde{p} : (x, \tilde{x}) = (y, p_*\,\tilde{x})$$

*such that $\mathsf{ap}\ \mathsf{fst}\ \tilde{p} = p$.*



# Dependent application

We have the congruence/application function

$$\mathsf{ap} : \{A\, B : \mathcal{U}\} \to (f : A \to B) \to \{x\, y : A\} \to (p : x = y) \to f\, x = f\, y$$

# Dependent application

We have the congruence/application function

$$\mathsf{ap} : \{A\,B : \mathcal{U}\} \to (f : A \to B) \to \{x\,y : A\} \to (p : x = y) \to f\,x = f\,y$$

We would now like to generalize it to the dependent case

$$\mathsf{apd} : \{A : \mathcal{U}\}\,\{B : A \to \mathcal{U}\} \to (f : (x : A) \to B\,x) \to \{x\,y : A\} \to (p : x = y) \to f\,x = f\,y$$

# Dependent application

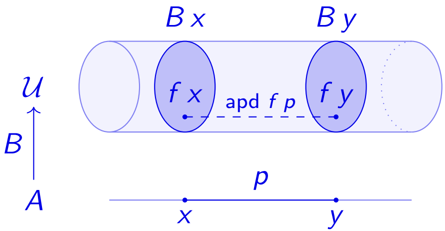
We have the congruence/application function

$$\mathsf{ap} : \{A\, B : \mathcal{U}\} \to (f : A \to B) \to \{x\, y : A\} \to (p : x = y) \to f\, x = f\, y$$

We would now like to generalize it to the dependent case

$$\mathsf{apd} : \{A : \mathcal{U}\}\, \{B : A \to \mathcal{U}\} \to (f : (x : A) \to B\, x) \to \{x\, y : A\} \to (p : x = y) \to f\, x = f\, y$$

We want to have a path between elements of $B\, x$ and $B\, y$ which is not allowed, but intuitively fine because we have a path $p : x = y$.



# Dependent application

One way out is to define from

$$f : (x : A) \to B\,x$$

the application to the total space

$$\begin{aligned} F &: A \to \Sigma A.B \\ x &\mapsto (x, f\,x) \end{aligned}$$

which is a section of $\mathsf{fst} : \Sigma A.B \to A$, i.e. $\mathsf{fst} \circ F(x) \triangleq x$,

# Dependent application

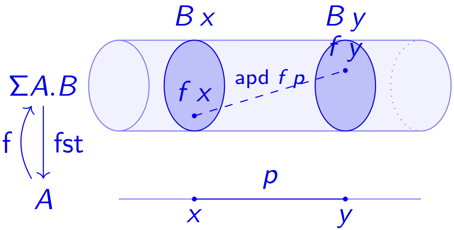
One way out is to define from

$$f : (x : A) \to B\,x$$

the application to the total space

$$F : A \to \Sigma A.B$$
$$x \mapsto (x, f\,x)$$

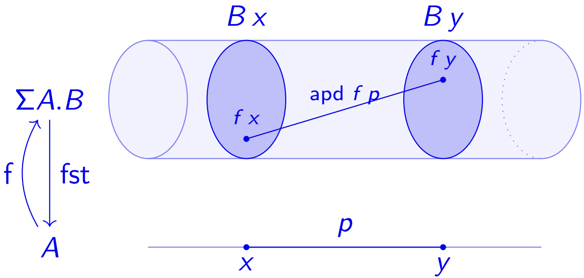
which is a section of $\mathsf{fst} : \Sigma A.B \to A$, i.e. $\mathsf{fst} \circ F(x) \triangleq x$, and use $\mathsf{ap}$ on $\tilde{f}$.




But we loose the fact that we are over $p$!

# Dependent application [Uni13, Lemma 2.3.4]

A better idea is to encode the path $\mathsf{apd}\ f\ p$ as a path in $B\ y$.

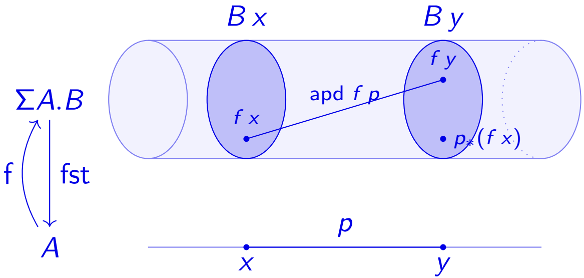



and we define

$$\mathsf{apd} : (f : (x : A) \to B\,x) \to \{x\,y : A\} \to (p : x = y) \to f\,x = f\,y$$

# Dependent application [Uni13, Lemma 2.3.4]

A better idea is to encode the path $\mathsf{apd}\ f\ p$ as a path in $B\ y$.

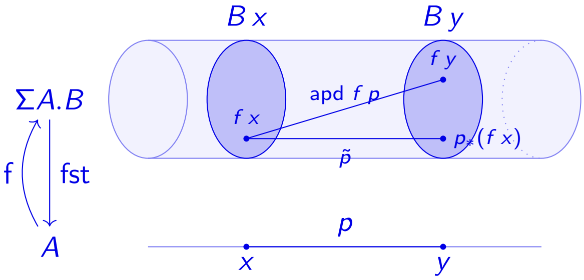



and we define

$$\mathsf{apd} : (f : (x : A) \to B\,x) \to \{x\,y : A\} \to (p : x = y) \to f\,x = f\,y$$

# Dependent application [Uni13, Lemma 2.3.4]

A better idea is to encode the path $\mathsf{apd}\ f\ p$ as a path in $B\ y$.

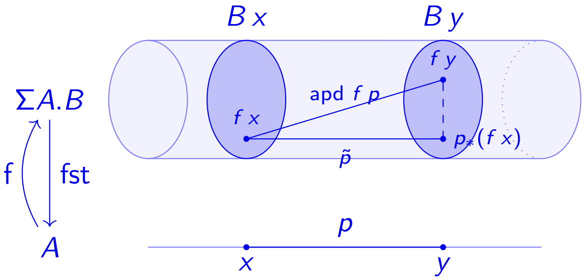



and we define

$$\mathsf{apd} : (f : (x : A) \to B\,x) \to \{x\,y : A\} \to (p : x = y) \to f\,x = f\,y$$

# Dependent application [Uni13, Lemma 2.3.4]

A better idea is to encode the path $\mathsf{apd}\ f\ p$ as a path in $B\,y$.

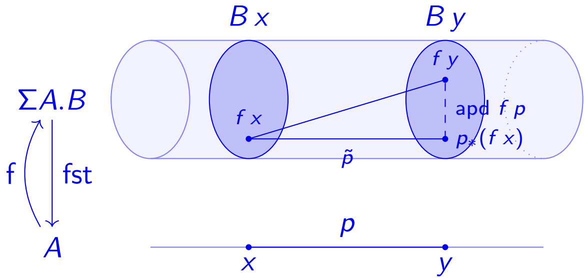



and we define

$$\mathsf{apd} : (f : (x : A) \to B\,x) \to \{x\,y : A\} \to (p : x = y) \to f\,x = f\,y$$

# Dependent application [Uni13, Lemma 2.3.4]

A better idea is to encode the path apd $f\ p$ as a path in $B\ y$.

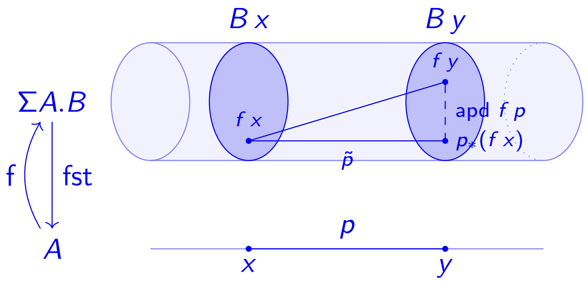



and we define

$$\text{apd} : (f : (x : A) \to B\ x) \to \{x\ y : A\} \to (p : x = y) \to f\ x = f\ y$$

# Dependent application [Uni13, Lemma 2.3.4]

A better idea is to encode the path $\mathsf{apd}\ f\ p$ as a path in $B\ y$.

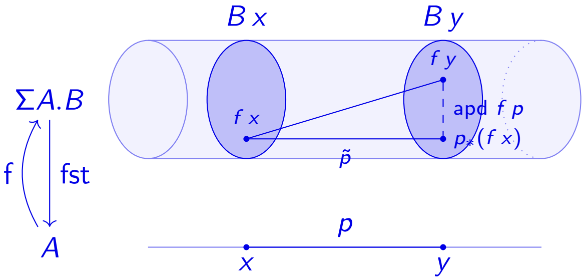



and we define

$$\mathsf{apd} : (f : (x : A) \to B\,x) \to \{x\,y : A\} \to (p : x = y) \to p_*(f\,x) = f\,y$$

# Dependent application [Uni13, Lemma 2.3.4]

A better idea is to encode the path $\mathsf{apd}\ f\ p$ as a path in $B\ y$.




and we define

$$\mathsf{apd} : (f : (x : A) \to B\ x) \to \{x\ y : A\} \to (p : x = y) \to p_*(f\ x) = f\ y$$

by induction by

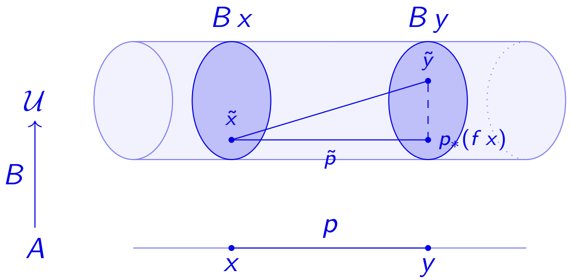
$$\mathsf{apd}\ f\ \mathsf{refl} \triangleq \mathsf{refl}_{f\ x}$$

# Paths over

More generally, given $x = y$ in $A$, $B : A \to \mathcal{U}$, $\tilde{x} : B\,x$ and $\tilde{y} : B\,y$, the type of **paths** in $B$ **over** $p$ between $\tilde{x}$ and $\tilde{y}$ is

$$\tilde{x} =^{B}_{p} \tilde{y} \quad \hat{=} \quad p_{*}\,\tilde{x} = \tilde{y}$$

which can be pictured as



# Paths in product types [Uni13, Section 2.6]

Suppose given $x\, x' : A$ and $y\, y' : B$.

A path $p : (x, y) = (x', y')$ induces paths

$$: x = x' \qquad\qquad : y = y'$$

# Paths in product types [Uni13, Section 2.6]

Suppose given $x\,x' : A$ and $y\,y' : B$.

A path $p : (x, y) = (x', y')$ induces paths

$$\mathsf{ap}\ \mathsf{fst}\ p : x = x' \qquad\qquad \mathsf{ap}\ \mathsf{snd}\ p : y = y'$$

# Paths in product types [Uni13, Section 2.6]

Suppose given $x\,x' : A$ and $y\,y' : B$.

A path $p : (x, y) = (x', y')$ induces paths

$$\mathsf{ap}\ \mathsf{fst}\ p : x = x' \qquad\qquad \mathsf{ap}\ \mathsf{snd}\ p : y = y'$$

Conversely, we have a function

$$\mathsf{pair}^{=} : (x = x') \to (y = y') \to (x, y) = (x', y')$$

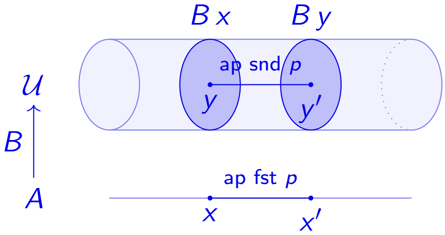
which is defined by path induction and is useful to construct paths in products.

# Path in dependent sum types [Uni13, Section 2.7]

Suppose given $x\, x' : A$, $y : B\, x$ and $y' : B\, x'$.

A path $p : (x, y) = (x', y')$ induces paths

$\text{ap fst } p : x = x'$ $\qquad$ $\text{ap snd } p :$

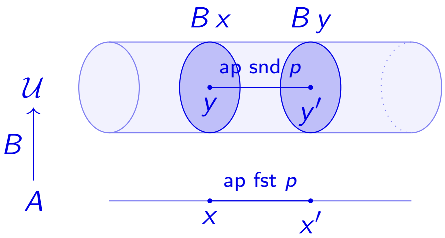


# Path in dependent sum types [Uni13, Section 2.7]

Suppose given $x\,x' : A$, $y : B\,x$ and $y' : B\,x'$.

A path $p : (x, y) = (x', y')$ induces paths

$$\mathsf{ap}\ \mathsf{fst}\ p : x = x' \qquad\qquad \mathsf{ap}\ \mathsf{snd}\ p : y =^{B}_{p} y'$$



# Path in dependent sum types [Uni13, Section 2.7]

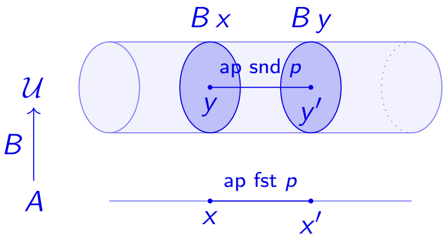
Suppose given $x\,x' : A$, $y : B\,x$ and $y' : B\,x'$.

A path $p : (x, y) = (x', y')$ induces paths

$$\mathsf{ap}\ \mathsf{fst}\ p : x = x' \qquad\qquad \mathsf{ap}\ \mathsf{snd}\ p : y =^B_p y'$$

Conversely, we have a function

$$\mathsf{pair}^= :$$



# Path in dependent sum types [Uni13, Section 2.7]

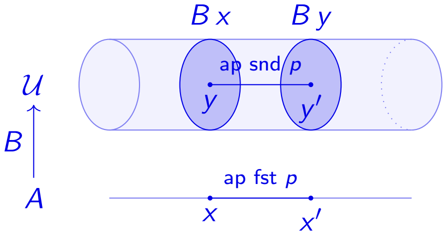
Suppose given $x\,x' : A$, $y : B\,x$ and $y' : B\,x'$.

A path $p : (x, y) = (x', y')$ induces paths

$$\mathsf{ap}\ \mathsf{fst}\ p : x = x' \qquad\qquad \mathsf{ap}\ \mathsf{snd}\ p : y =^B_p y'$$

Conversely, we have a function

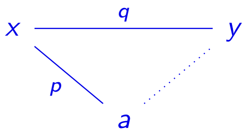
$$\mathsf{pair}^= : (p : x = x') \to (y =^B_p y') \to (x, y) = (x', y')$$



# Transporting paths

**Proposition ([Uni13, Lemma 2.11.2])**
*Given paths* $p : a = x$ *and* $q : x = y$*, we have*

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



**Proposition ([Uni13, Lemma 2.11.2])**
*Given paths* $p : a = x$ *and* $q : x = y$*, we have*

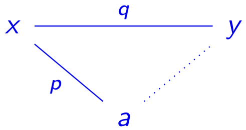
$$\mathsf{transport}\,(\lambda x.a = x)\; q\; p = p \cdot q$$

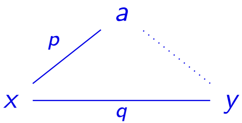
*Similarly, given paths* $p : x = a$ *and* $q : x = y$*, we have*

$$\mathsf{transport}\,(\lambda x.a = x)\; q\; p = q^{-} \cdot p$$

*Similarly, given paths* $p : x = x$ *and* $q : x = y$*, we have*

$$\mathsf{transport}\,(\lambda x.x = x)\; q\; p = q^{-} \cdot p \cdot q$$



**Proposition ([Uni13, Lemma 2.11.2])**
*Given paths $p : a = x$ and $q : x = y$, we have*

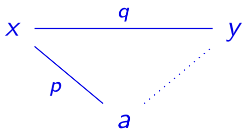
$$\mathsf{transport}\,(\lambda x.a = x)\; q\; p = p \cdot q$$

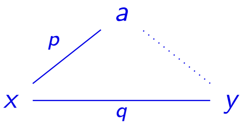
*Similarly, given paths $p : x = a$ and $q : x = y$, we have*

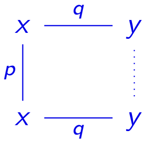
$$\mathsf{transport}\,(\lambda x.a = x)\; q\; p = q^{-} \cdot p$$

*Similarly, given paths $p : x = x$ and $q : x = y$, we have*

$$\mathsf{transport}\,(\lambda x.x = x)\; q\; p = q^{-} \cdot p \cdot q$$




**Proof.**
By path induction on $p$. □

# Transporting paths

More generally,

**Proposition ([Uni13, Lemma 2.11.3])**
*Given* $f\, g : A \to B$, $p : f\, x = g\, x$ *in* $B$ *and* $q : x = y$ *in* $A$,

$$\mathsf{transport}\,(\lambda x. f\, x = g\, x)\, q\, p \quad = \quad (\mathsf{ap}\ f\ q)^{-} \cdot p \cdot \mathsf{ap}\ g\, q$$

*in* $f\, y = g\, y$.

# Transporting functions

Writing $\mathbb{N}$ for the unary natural number and $\mathbb{B}$ for the binary ones, we have

$$p : \mathbb{N} = \mathbb{B}$$

# Transporting functions

Writing $\mathbb{N}$ for the unary natural number and $\mathbb{B}$ for the binary ones, we have

$$p : \mathbb{N} = \mathbb{B}$$

By transport, we obtain

$$\mathsf{coe}\ p : \mathbb{N} \to \mathbb{B} \qquad\qquad \mathsf{coe}\ p^{-} : \mathbb{B} \to \mathbb{N}$$

# Transporting functions

Writing $\mathbb{N}$ for the unary natural number and $\mathbb{B}$ for the binary ones, we have

$$p : \mathbb{N} = \mathbb{B}$$

By transport, we obtain

$$\mathsf{coe}\ p : \mathbb{N} \to \mathbb{B} \qquad\qquad \mathsf{coe}\ p^{-} : \mathbb{B} \to \mathbb{N}$$

Writing $\mathsf{suc} : \mathbb{N} \to \mathbb{N}$ for the successor function, by transport we have

$$\mathsf{transport}\,(\lambda X.X \to X)\,p\ \mathsf{suc} : \mathbb{B} \to \mathbb{B}$$

# Transporting functions

Writing $\mathbb{N}$ for the unary natural number and $\mathbb{B}$ for the binary ones, we have

$$p : \mathbb{N} = \mathbb{B}$$

By transport, we obtain

$$\mathsf{coe}\ p : \mathbb{N} \to \mathbb{B} \qquad\qquad \mathsf{coe}\ p^{-} : \mathbb{B} \to \mathbb{N}$$

Writing $\mathsf{suc} : \mathbb{N} \to \mathbb{N}$ for the successor function, by transport we have

$$\mathsf{transport}\,(\lambda X.X \to X)\,p\ \mathsf{suc} : \mathbb{B} \to \mathbb{B}$$

This function is

# Transporting functions

Writing $\mathbb{N}$ for the unary natural number and $\mathbb{B}$ for the binary ones, we have

$$p : \mathbb{N} = \mathbb{B}$$

By transport, we obtain

$$\mathsf{coe}\ p : \mathbb{N} \to \mathbb{B} \qquad\qquad \mathsf{coe}\ p^{-} : \mathbb{B} \to \mathbb{N}$$

Writing $\mathsf{suc} : \mathbb{N} \to \mathbb{N}$ for the successor function, by transport we have

$$\mathsf{transport}\,(\lambda X.X \to X)\ p\ \mathsf{suc} : \mathbb{B} \to \mathbb{B}$$

This function is

$$\mathsf{transport}\,(\lambda X.X \to X)\ p\ \mathsf{suc} = (\mathsf{coe}\ p) \circ \mathsf{suc} \circ (\mathsf{coe}\ p^{-})$$

# Transporting functions

**Proposition**
*Given a function* $f : A \to B$ *and paths* $p : A = A'$ *and* $q : B = B'$*, we have*

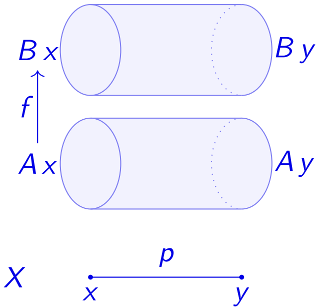
$$\text{transport}\,(\lambda X.A \to X)\,q\,f = \text{coe}\ q \circ f$$

$$\text{transport}\,(\lambda X.X \to B)\,p\,f = f \circ \text{coe}\ p^{-}$$

# Transporting functions

**Proposition ([Uni13, (2.9.4)])**
*Given type families $A : X \to \mathcal{U}$ and $B : X \to \mathcal{U}$, a path $p : x = y$ in $X$ and a function $f : A\,x \to B\,x$, we have*

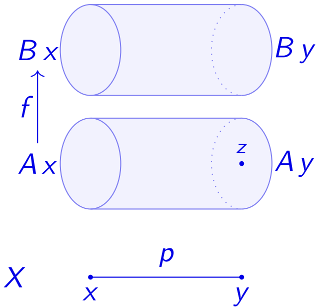
$$\text{transport}\,(\lambda x.A\,x \to B\,x)\,p\,f = \textit{transport}\,B\,p \circ f \circ \text{transport}\;A\,p^{-}$$



# Transporting functions

**Proposition ([Uni13, (2.9.4)])**
*Given type families $A : X \to \mathcal{U}$ and $B : X \to \mathcal{U}$, a path $p : x = y$ in $X$ and a function $f : A\,x \to B\,x$, we have*

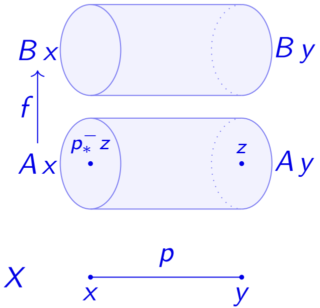
$$\mathsf{transport}\,(\lambda x.A\,x \to B\,x)\,p\,f = \mathit{transport}\,B\,p \circ f \circ \mathsf{transport}\;A\,p^{-}$$



# Transporting functions

**Proposition ([Uni13, (2.9.4)])**
*Given type families $A : X \to \mathcal{U}$ and $B : X \to \mathcal{U}$, a path $p : x = y$ in $X$ and a function $f : A\,x \to B\,x$, we have*

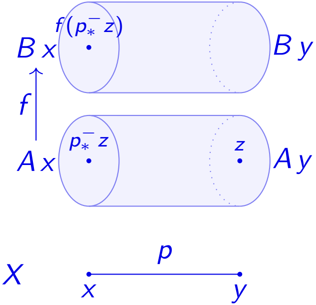
$$\mathsf{transport}\,(\lambda x. A\,x \to B\,x)\,p\,f = transport\;B\,p \circ f \circ \mathsf{transport}\;A\,p^{-}$$



# Transporting functions

**Proposition ([Uni13, (2.9.4)])**
*Given type families $A : X \to \mathcal{U}$ and $B : X \to \mathcal{U}$, a path $p : x = y$ in $X$ and a function $f : A\,x \to B\,x$, we have*

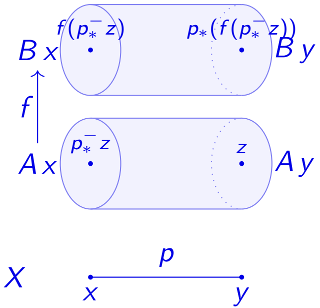
$$\mathsf{transport}\,(\lambda x.A\,x \to B\,x)\,p\,f = \mathit{transport}\;B\;p \circ f \circ \mathsf{transport}\;A\,p^{-}$$



# Transporting functions

**Proposition ([Uni13, (2.9.4)])**
*Given type families $A : X \to \mathcal{U}$ and $B : X \to \mathcal{U}$, a path $p : x = y$ in $X$ and a function $f : A\,x \to B\,x$, we have*

$$\text{transport}\,(\lambda x. A\,x \to B\,x)\,p\,f = \textit{transport}\;B\,p \circ f \circ \text{transport}\;A\,p^{-}$$



[ACD+20] Andreas Abel, Jesper Cockx, Dominique Devriese, Amin Timany, and Philip Wadler.
**Leibniz equality is isomorphic to Martin-Löf identity, parametrically.**
*Journal of Functional Programming*, 30, 2020.
doi:10.1017/S0956796820000155.

[Gro83] Alexander Grothendieck.
**Pursuing stacks, 1983.**
Letter to Daniel Quillen.
arXiv:2111.01000.

[Lei86] Gottfried Wilhelm Leibniz.
***Discours de métaphysique*.**
1686.

[Lum10] Peter LeFanu Lumsdaine.
**Weak omega-categories from intensional type theory.**
*Logical Methods in Computer Science*, 6, 2010.
`arXiv:0812.0409`, `doi:10.2168/LMCS-6(3:24)2010`.

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

[VDBG11] Benno Van Den Berg and Richard Garner.
**Types are weak $\omega$-groupoids.**
*Proceedings of the london mathematical society*, 102(2):370–394, 2011.
arXiv:0812.0298, doi:10.1112/plms/pdq026.