Univalence

Samuel Mimram

2025

École polytechnique

Contractible types

We have seen that a type is contractible when it is equivalent to 1: for instance

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

i.e. (roughly)

$$\text{———} \quad \simeq \quad \bullet$$

Contractible types

We have seen that a type is contractible when it is equivalent to 1: for instance

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

i.e. (roughly)

$$\text{———} \quad \simeq \quad \bullet$$

However, nothing guarantees that we cannot distinguish between $\| \mathsf{Bool} \|_{-1}$ and 1.

Constructing identities

For now, we do not have a way of constructing non-trivial identities.

For instance, we have no way to show

$$\mathsf{Bool} \quad = \quad 1 \sqcup 1$$

(even though we can show that they are equivalent).

UNIVALENCE

Lemma
We have a function

$$\mathsf{idtoequiv} \quad : \quad (A = B) \to (A \simeq B)$$

Proof (non-satisfactory computationally).
We could define it by path induction by

$$\mathsf{idtoequiv}(\mathsf{refl}) \quad \hat{=} \quad \mathsf{id} \qquad \square$$

Lemma
We have a function

$$\mathsf{idtoequiv} \quad : \quad (A = B) \to (A \simeq B)$$

Proof (non-satisfactory computationally).
We could define it by path induction by

$$\mathsf{idtoequiv}(\mathsf{refl}) \quad \hat{=} \quad \mathsf{id}$$

□

This is not entirely satisfactory, because the underlying function $\mathsf{idtoequiv}(p) : A \to B$ is not something definitionally known so that we have to do the lemmas again...

For instance: $\mathsf{idtoequiv}(p \cdot q) = \mathsf{idtoequiv}(q) \circ \mathsf{idtoequiv}(p)$.

Lemma ([Uni13, Lemma 2.10.1])
We have a function

$$\mathsf{idtoequiv} \quad : \quad (A = B) \to (A \simeq B)$$

Proof.
We have a function

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

defined by $\mathsf{coe}(\mathsf{refl}) \hat{\equiv} \mathsf{id}$ or rather by $\mathsf{coe} \hat{\equiv} \mathsf{transport}\,(\lambda X.X)\,p\,x$

Lemma ([Uni13, Lemma 2.10.1])
We have a function

$$\mathsf{idtoequiv} \quad : \quad (A = B) \to (A \simeq B)$$

Proof.
We have a function

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

defined by $\mathsf{coe}(\mathsf{refl}) \hat{\equiv} \mathsf{id}$ or rather by $\mathsf{coe} \hat{\equiv} \mathsf{transport}\,(\lambda X.X)\,p\,x$ and we claim that $eq : (p : A = B) \to \mathsf{isEquiv}(\mathsf{coe}(p))$.

Lemma ([Uni13, Lemma 2.10.1])
We have a function

$$\mathsf{idtoequiv} \quad : \quad (A = B) \to (A \simeq B)$$

Proof.
We have a function

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

defined by $\mathsf{coe}(\mathsf{refl}) \hat{\equiv} \mathsf{id}$ or rather by $\mathsf{coe} \hat{\equiv} \mathsf{transport}\,(\lambda X.X)\,p\,x$ and we claim that $eq : (p : A = B) \to \mathsf{isEquiv}(\mathsf{coe}(p))$.

By path induction on p, we have to show $\mathsf{isEquiv}(\mathsf{id})$, which is easy.

Lemma ([Uni13, Lemma 2.10.1])
We have a function

$$\mathsf{idtoequiv} \quad : \quad (A = B) \to (A \simeq B)$$

Proof.
We have a function

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

defined by $\mathsf{coe}(\mathsf{refl}) \hat{\equiv} \mathsf{id}$ or rather by $\mathsf{coe} \hat{\equiv} \mathsf{transport}\,(\lambda X.X)\,p\,x$ and we claim that $eq : (p : A = B) \to \mathsf{isEquiv}(\mathsf{coe}(p))$.

By path induction on p, we have to show $\mathsf{isEquiv}(\mathsf{id})$, which is easy.

And we define

$$\mathsf{idtoequiv}(p) \quad \hat{\equiv} \quad (coe(p), eq(p))$$

□

Univalence

Axiom
The **univalence axiom** states that, for $A\,B : \mathcal{U}$, the function

$$\mathsf{idtoequiv} \quad : \quad (A = B) \to (A \simeq B)$$

is itself and equivalence.

Univalence

That idtoequiv : $(A = B) \to (A \simeq B)$ is an equivalence means that we have

- an elimination rule:

$$\text{idtoequiv} : (A = B) \to (A \simeq B)$$

Univalence

That idtoequiv : $(A = B) \to (A \simeq B)$ is an equivalence means that we have

- an elimination rule:

$$\text{idtoequiv} : (A = B) \to (A \simeq B)$$

- an introduction rule:

$$\text{ua} : (A \simeq B) \to A = B$$

Univalence

That $\mathsf{idtoequiv} : (A = B) \to (A \simeq B)$ is an equivalence means that we have

- an elimination rule:

$$\mathsf{idtoequiv} : (A = B) \to (A \simeq B)$$

- an introduction rule:

$$\mathsf{ua} : (A \simeq B) \to A = B$$

- a computation rule: for $f : A \simeq B$ and $x : A$,

$$\mathsf{coe}\,(\mathsf{ua}\ f)\,x = f\,x$$

Univalence

That $\mathsf{idtoequiv} : (A = B) \to (A \simeq B)$ is an equivalence means that we have

- an elimination rule:

$$\mathsf{idtoequiv} : (A = B) \to (A \simeq B)$$

- an introduction rule:

$$\mathsf{ua} : (A \simeq B) \to A = B$$

- a computation rule: for $f : A \simeq B$ and $x : A$,

$$\mathsf{coe}\,(\mathsf{ua}\ f)\,x = f\,x$$

(and $\mathsf{coe}\,(\mathsf{ua}\ f)^{-1}\,x = f^{-1}\,x$)

Univalence

That $\mathsf{idtoequiv} : (A = B) \to (A \simeq B)$ is an equivalence means that we have

- an elimination rule:

$$\mathsf{idtoequiv} : (A = B) \to (A \simeq B)$$

- an introduction rule:

$$\mathsf{ua} : (A \simeq B) \to A = B$$

- a computation rule: for $f : A \simeq B$ and $x : A$,

$$\mathsf{coe}\,(\mathsf{ua}\ f)\,x = f\,x$$

  (and $\mathsf{coe}\,(\mathsf{ua}\ f)^{-1}\,x = f^{-1}\,x$)
- a uniqueness rule: for $p : A = B$

$$\mathsf{ua}\,(\mathsf{coe}\ p) = p$$

A first family of applications is the construction of equalities, by ua.

A first family of applications is the construction of equalities, by ua.

For instance, we have

$$e \quad : \quad \mathsf{Bool} \quad \simeq \quad 1 \sqcup 1$$

A first family of applications is the construction of equalities, by ua.

For instance, we have

$$e \quad : \quad \mathsf{Bool} \quad \simeq \quad 1 \sqcup 1$$

and therefore

$$\mathsf{ua}\ e \quad : \quad \mathsf{Bool} \quad = \quad 1 \sqcup 1$$

A first family of applications is the construction of equalities, by ua.

For instance, we have

$$e \quad : \quad \mathsf{Bool} \quad \simeq \quad 1 \sqcup 1$$

and therefore

$$\mathsf{ua}\ e \quad : \quad \mathsf{Bool} \quad = \quad 1 \sqcup 1$$

By transport, if we have $P(\mathsf{Bool})$ then we have $P(1 \sqcup 1)$, i.e. any property satisfied by one is satisfied by the other.

More realistically, think of two implementations of natural numbers: unary and binary.

Independence under implementation

Suppose that we have two implementations of natural numbers $\mathbb{N}_{\mathsf{unary}}$ and $\mathbb{N}_{\mathsf{binary}}$.

Independence under implementation

Suppose that we have two implementations of natural numbers $\mathbb{N}_{\mathsf{unary}}$ and $\mathbb{N}_{\mathsf{binary}}$.

We will be able to show that $e : \mathbb{N}_{\mathsf{unary}} \simeq \mathbb{N}_{\mathsf{binary}}$.

Independence under implementation

Suppose that we have two implementations of natural numbers $\mathbb{N}_{\mathsf{unary}}$ and $\mathbb{N}_{\mathsf{binary}}$.

We will be able to show that $e : \mathbb{N}_{\mathsf{unary}} \simeq \mathbb{N}_{\mathsf{binary}}$.

And thus $\mathsf{ua}\ e : \mathbb{N}_{\mathsf{unary}} = \mathbb{N}_{\mathsf{binary}}$.

Independence under implementation

Suppose that we have two implementations of natural numbers $\mathbb{N}_{\mathsf{unary}}$ and $\mathbb{N}_{\mathsf{binary}}$.

We will be able to show that $e : \mathbb{N}_{\mathsf{unary}} \simeq \mathbb{N}_{\mathsf{binary}}$.

And thus $\mathsf{ua}\ e : \mathbb{N}_{\mathsf{unary}} = \mathbb{N}_{\mathsf{binary}}$.

This means that any function working on one can be transported to a function working on the other: given

$$\mathsf{pred} \quad : \quad \mathbb{N}_{\mathsf{unary}} \to \mathbb{N}_{\mathsf{unary}}$$

we have

Independence under implementation

Suppose that we have two implementations of natural numbers $\mathbb{N}_{\mathsf{unary}}$ and $\mathbb{N}_{\mathsf{binary}}$.

We will be able to show that $e : \mathbb{N}_{\mathsf{unary}} \simeq \mathbb{N}_{\mathsf{binary}}$.

And thus $\mathsf{ua}\ e : \mathbb{N}_{\mathsf{unary}} = \mathbb{N}_{\mathsf{binary}}$.

This means that any function working on one can be transported to a function working on the other: given

$$\mathsf{pred} \quad : \quad \mathbb{N}_{\mathsf{unary}} \to \mathbb{N}_{\mathsf{unary}}$$

we have

$$\mathsf{pred}' \quad \hat{=} \quad \mathsf{transport}\,(\lambda X.X \to X)\,(\mathsf{ua}\ e)\,(\mathsf{pred}) \quad : \quad \mathbb{N}_{\mathsf{binary}} \to \mathbb{N}_{\mathsf{binary}}$$

Independence under implementation

Suppose that we have two implementations of natural numbers $\mathbb{N}_{\mathsf{unary}}$ and $\mathbb{N}_{\mathsf{binary}}$.

We will be able to show that $e : \mathbb{N}_{\mathsf{unary}} \simeq \mathbb{N}_{\mathsf{binary}}$.

And thus $\mathsf{ua}\ e : \mathbb{N}_{\mathsf{unary}} = \mathbb{N}_{\mathsf{binary}}$.

This means that any function working on one can be transported to a function working on the other: given

$$\mathsf{pred} \quad : \quad \mathbb{N}_{\mathsf{unary}} \to \mathbb{N}_{\mathsf{unary}}$$

we have

$$\mathsf{pred}' \quad \hat{=} \quad \mathsf{transport}\,(\lambda X.X \to X)\,(\mathsf{ua}\ e)\,(\mathsf{pred}) \quad : \quad \mathbb{N}_{\mathsf{binary}} \to \mathbb{N}_{\mathsf{binary}}$$

This is not magic:

$$\mathsf{pred}' \quad = \quad (\mathsf{coe}\,(\mathsf{ua}\ e)) \circ \mathsf{pred} \circ (\mathsf{coe}\,(\mathsf{ua}\ e)^{-})$$

Independence under implementation

Suppose that we have two implementations of natural numbers $\mathbb{N}_{\mathsf{unary}}$ and $\mathbb{N}_{\mathsf{binary}}$.

We will be able to show that $e : \mathbb{N}_{\mathsf{unary}} \simeq \mathbb{N}_{\mathsf{binary}}$.

And thus $\mathsf{ua}\ e : \mathbb{N}_{\mathsf{unary}} = \mathbb{N}_{\mathsf{binary}}$.

This means that any function working on one can be transported to a function working on the other: given

$$\mathsf{pred} \quad : \quad \mathbb{N}_{\mathsf{unary}} \to \mathbb{N}_{\mathsf{unary}}$$

we have

$$\mathsf{pred}' \quad \hat{=} \quad \mathsf{transport}\,(\lambda X.X \to X)\,(\mathsf{ua}\ e)\,(\mathsf{pred}) \quad : \quad \mathbb{N}_{\mathsf{binary}} \to \mathbb{N}_{\mathsf{binary}}$$

This is not magic:

$$\mathsf{pred}' \quad = \quad e \circ \mathsf{pred} \circ e^{-1}$$

Homotopy invariance

Geometrically, this means that we cannot distinguish between two homotopy equivalent types

$$A \quad \simeq \quad B$$

because we have

$$A = B$$

Otherwise said, all constructions are *invariant under homotopy equivalence*!

Univalence: universe levels

Axiom
The **univalence axiom** states that, for $A\,B : \mathcal{U}_\ell$, the function

$$\mathsf{idtoequiv} \quad : \quad (A = B) \to (A \simeq B)$$

is itself and equivalence.

We can have $A \simeq B$ for $A : \mathcal{U}_\ell$ and $B : \mathcal{U}_{\ell'}$,
but $A = B$ only makes sense at the same level.

Contractible types

We have seen earlier that a type A is contractible iff $A \simeq 1$. By univalence, we have

Lemma ([Uni13, Lemma 3.11.3])
Given a small contractible type A, we have $A = 1$.

Contractible types

We have seen earlier that a type A is contractible iff $A \simeq 1$. By univalence, we have

Lemma ([Uni13, Lemma 3.11.3])
Given a small contractible type A, we have $A = 1$.

Note: there are also large contractible types such as

Contractible types

We have seen earlier that a type A is contractible iff $A \simeq 1$. By univalence, we have

Lemma ([Uni13, Lemma 3.11.3])
Given a small contractible type A, we have $A = 1$.

Note: there are also large contractible types such as $\|\mathcal{U}\|_{-1}$.

Contractible types

We have seen earlier that a type A is contractible iff $A \simeq 1$. By univalence, we have

Lemma ([Uni13, Lemma 3.11.3])
Given a small contractible type A, we have $A = 1$.

Note: there are also large contractible types such as $\|\mathcal{U}\|_{-1}$.

We also have

Lemma
Given a small type A such that $\neg A$, we have $A = 0$.

Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

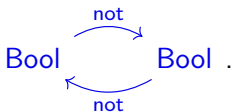
We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Therefore we have an equivalence $e : \mathsf{Bool} \simeq \mathsf{Bool}$: 

.

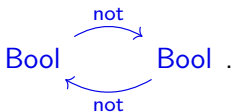
Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Therefore we have an equivalence $e : \mathsf{Bool} \simeq \mathsf{Bool}$:



.

By univalence, we have a path $p : \mathsf{Bool} = \mathsf{Bool}$.

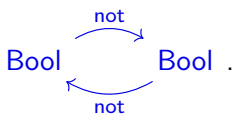
Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Therefore we have an equivalence $e : \mathsf{Bool} \simeq \mathsf{Bool}$:



.

By univalence, we have a path $p : \mathsf{Bool} = \mathsf{Bool}$.

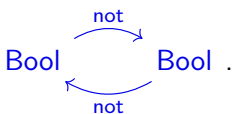
This path satisfies: $\mathsf{coe}\ p\ \mathsf{false} =$ (by univalence computation).

Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Therefore we have an equivalence $e : \mathsf{Bool} \simeq \mathsf{Bool}$: 

.

By univalence, we have a path $p : \mathsf{Bool} = \mathsf{Bool}$.

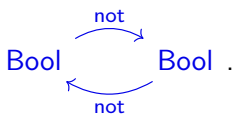
This path satisfies: $\mathsf{coe}\ p\ \mathsf{false} = \mathsf{not}\ \mathsf{false}$ (by univalence computation).

Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Therefore we have an equivalence $e : \mathsf{Bool} \simeq \mathsf{Bool}$: 

.

By univalence, we have a path $p : \mathsf{Bool} = \mathsf{Bool}$.

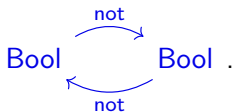
This path satisfies: $\mathsf{coe}\ p\ \mathsf{false} = \mathsf{not}\ \mathsf{false} = \mathsf{true}$ (by univalence computation).

Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Therefore we have an equivalence $e : \mathsf{Bool} \simeq \mathsf{Bool}$: 

.

By univalence, we have a path $p : \mathsf{Bool} = \mathsf{Bool}$.

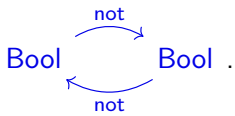
This path satisfies: $\mathsf{coe}\ p\ \mathsf{false} = \mathsf{not}\ \mathsf{false} = \mathsf{true}$ (by univalence computation).

Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

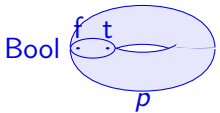
We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Therefore we have an equivalence $e : \mathsf{Bool} \simeq \mathsf{Bool}$: 

.

By univalence, we have a path $p : \mathsf{Bool} = \mathsf{Bool}$.

This path satisfies: $\mathsf{coe}\ p\ \mathsf{false} = \mathsf{not}\ \mathsf{false} = \mathsf{true}$ (by univalence computation).

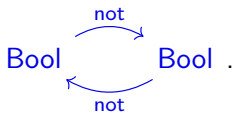


Incompatibility with the set-theoretic interpretation

We can also now start to construct some interesting non-trivial paths!

Consider the boolean negation function $\mathsf{not} : \mathsf{Bool} \to \mathsf{Bool}$.

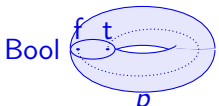
We have $\mathsf{not} \circ \mathsf{not} = \mathsf{id}_{\mathsf{Bool}}$.

Therefore we have an equivalence $e : \mathsf{Bool} \simeq \mathsf{Bool}$: 

.

By univalence, we have a path $p : \mathsf{Bool} = \mathsf{Bool}$.

This path satisfies: $\mathsf{coe}\ p\ \mathsf{false} = \mathsf{not}\ \mathsf{false} = \mathsf{true}$ (by univalence computation).



Incompatibility with the set-theoretic interpretation

It might be secretly the case that all the types we can constructs are sets?

Incompatibility with the set-theoretic interpretation

It might be secretly the case that all the types we can constructs are sets?

Proposition ([Uni13, Example 3.1.9])
The universe is not a set.

Proof.
Suppose that $\mathcal{U}$ is a set.

Incompatibility with the set-theoretic interpretation

It might be secretly the case that all the types we can constructs are sets?

Proposition ([Uni13, Example 3.1.9])
The universe is not a set.

Proof.
Suppose that $\mathcal{U}$ is a set. This implies that two paths p and refl of type Bool $=$ Bool are equal.

Incompatibility with the set-theoretic interpretation

It might be secretly the case that all the types we can constructs are sets?

Proposition ([Uni13, Example 3.1.9])
The universe is not a set.

Proof.
Suppose that $\mathcal{U}$ is a set. This implies that two paths p and refl of type Bool $=$ Bool are equal. Therefore we have

$$\text{coe } p \text{ false} = \text{coe refl false}$$

Incompatibility with the set-theoretic interpretation

It might be secretly the case that all the types we can constructs are sets?

Proposition ([Uni13, Example 3.1.9])
The universe is not a set.

Proof.
Suppose that $\mathcal{U}$ is a set. This implies that two paths p and refl of type Bool $=$ Bool are equal. Therefore we have

$$\text{true} = \text{coe}\ p\ \text{false} = \text{coe refl false}$$

Incompatibility with the set-theoretic interpretation

It might be secretly the case that all the types we can constructs are sets?

Proposition ([Uni13, Example 3.1.9])
The universe is not a set.

Proof.
Suppose that $\mathcal{U}$ is a set. This implies that two paths p and refl of type Bool $=$ Bool are equal. Therefore we have

$$\text{true} = \text{coe } p \text{ false} = \text{coe refl false} = \text{false}$$

Incompatibility with the set-theoretic interpretation

It might be secretly the case that all the types we can constructs are sets?

Proposition ([Uni13, Example 3.1.9])
The universe is not a set.

Proof.
Suppose that $\mathcal{U}$ is a set. This implies that two paths p and refl of type Bool $=$ Bool are equal. Therefore we have

$$\text{true} = \text{coe } p \text{ false} = \text{coe refl false} = \text{false}$$

Contradiction. □

Incompatibility with classical logic

It might be secretly the case that the logic is classical?

Incompatibility with classical logic

It might be secretly the case that the logic is classical?

Proposition ([Uni13, Corollary 3.2.7])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$A \sqcup \neg A$$

Incompatibility with classical logic

It might be secretly the case that the logic is classical?

Proposition ([Uni13, Corollary 3.2.7])
It is not the case that for every $A : \mathcal{U}$, we have

$$A \sqcup \neg A$$

Proof.
It is well known that

$$(A : \mathcal{U}) \to A \sqcup \neg A$$

implies

$$(A : \mathcal{U}) \to \neg\neg A \to A$$

(it is in fact logically equivalent) and we do not have this (see next slide). □

Incompatibility with classical logic

It might be secretly the case that the logic is classical?

Proposition ([Uni13, Corollary 3.2.7])
It is not the case that for every $A : \mathcal{U}$, we have

$$A \sqcup \neg A$$

Proof.
It is well known that

$$(A : \mathcal{U}) \to A \sqcup \neg A$$

implies

$$(A : \mathcal{U}) \to \neg\neg A \to A$$

(it is in fact logically equivalent) and we do not have this (see next slide). □

Note that we are parametric in A!

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$,

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$, we have

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have $f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have
$f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.
With $p : \mathsf{Bool} = \mathsf{Bool}$ the non-trivial path, we define $g : \mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f$.

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have

$f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.

With $p : \mathsf{Bool} = \mathsf{Bool}$ the non-trivial path, we define $g : \mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f$.

We have $g = f$ by $\mathsf{apd}\,(\lambda X.\neg\neg X \to X)\,F\,p$,

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have

$f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.

With $p : \mathsf{Bool} = \mathsf{Bool}$ the non-trivial path, we define $g : \mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f$.

We have $g = f$ by $\mathsf{apd}\,(\lambda X.\neg\neg X \to X)\,F\,p$, i.e. for $x : \neg\neg\,\mathsf{Bool}$

$$\mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f\,x = f\,x$$

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have
$f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.
With $p : \mathsf{Bool} = \mathsf{Bool}$ the non-trivial path, we define $g : \mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f$.

We have $g = f$ by $\mathsf{apd}\,(\lambda X.\neg\neg X \to X)\,F\,p$, i.e. for $x : \neg\neg\,\mathsf{Bool}$

$$\mathsf{transport}\,(\lambda X.X)\,p\,(f(\mathsf{transport}\,(\lambda X.\neg\neg X)\,p\,x)) = f\,x$$

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$, we have

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have
$f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.
With $p : \mathsf{Bool} = \mathsf{Bool}$ the non-trivial path, we define $g : \mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f$.

We have $g = f$ by $\mathsf{apd}\,(\lambda X.\neg\neg X \to X)\,F\,p$, i.e. for $x : \neg\neg\,\mathsf{Bool}$

$$\mathsf{transport}\,(\lambda X.X)\,p\,(f\,x) = f\,x$$

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have

$f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.
With $p : \mathsf{Bool} = \mathsf{Bool}$ the non-trivial path, we define $g : \mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f$.

We have $g = f$ by $\mathsf{apd}\,(\lambda X.\neg\neg X \to X)\,F\,p$, i.e. for $x : \neg\neg\,\mathsf{Bool}$

$$\mathsf{coe}\ p\,(f\,x) = f\,x$$

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have

$f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.

With $p : \mathsf{Bool} = \mathsf{Bool}$ the non-trivial path, we define $g : \mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f$.

We have $g = f$ by $\mathsf{apd}\,(\lambda X.\neg\neg X \to X)\,F\,p$, i.e. for $x : \neg\neg\,\mathsf{Bool}$

$$\mathsf{not}\,(f\,x) = f\,x$$

Incompatibility with classical logic

Proposition ([Uni13, Theorem 3.2.2])
It is not the case that for every $A : \mathcal{U}$*, we have*

$$\neg\neg A \to A$$

Proof.
Suppose that we have $F : (A : \mathcal{U}) \to \neg\neg A \to A$, in particular we have
$f \triangleq F\ \mathsf{Bool} : \neg\neg\,\mathsf{Bool} \to \mathsf{Bool}$.
With $p : \mathsf{Bool} = \mathsf{Bool}$ the non-trivial path, we define $g : \mathsf{transport}\,(\lambda X.\neg\neg X \to X)\,p\,f$.

We have $g = f$ by $\mathsf{apd}\,(\lambda X.\neg\neg X \to X)\,F\,p$, i.e. for $x : \neg\neg\,\mathsf{Bool}$

$$\mathsf{not}\,(f\,x) = f\,x$$

which is impossible. □

Incompatibility with classical logic

We have see that we cannot assume the principle

$$(A : \mathcal{U}) \to A \sqcup \neg A$$

Incompatibility with classical logic

We have see that we cannot assume the principle

$$(A : \mathcal{U}) \to A \sqcup \neg A$$

however, we can assume

$$(A : \mathsf{HProp}) \to A \sqcup \neg A$$

which is the right way of formulating excluded middle.

Decidable propositions

In fact, assuming that the logic is classical amounts to assuming that all propositions are booleans.

Decidable propositions

In fact, assuming that the logic is classical amounts to assuming that all propositions are booleans.

Recall that a proposition A is **decidable** when $A \sqcup \neg A$ holds.

Decidable propositions

In fact, assuming that the logic is classical amounts to assuming that all propositions are booleans.

Recall that a proposition A is **decidable** when $A \sqcup \neg A$ holds.

Proposition
Decidable propositions are booleans.

Proof (sketch).
We want to show $\Sigma(A : \mathcal{U}).(\mathsf{isProp}\, A \times (A \sqcup \neg A)) \simeq \mathsf{Bool}$.

Decidable propositions

In fact, assuming that the logic is classical amounts to assuming that all propositions are booleans.

Recall that a proposition A is **decidable** when $A \sqcup \neg A$ holds.

Proposition
Decidable propositions are booleans.

Proof (sketch).
We want to show $\Sigma(A : \mathcal{U}).(\mathsf{isProp}\, A \times (A \sqcup \neg A)) \simeq \mathsf{Bool}$. From left-to-right we pick true or false depending on if we have A or $\neg A$.
From right-to-left we pick the function picking 0 or 1 depending on the boolean.

Decidable propositions

In fact, assuming that the logic is classical amounts to assuming that all propositions are booleans.

Recall that a proposition A is **decidable** when $A \sqcup \neg A$ holds.

Proposition
Decidable propositions are booleans.

Proof (sketch).
We want to show $\Sigma(A : \mathcal{U}).(\mathsf{isProp}\, A \times (A \sqcup \neg A)) \simeq \mathsf{Bool}$. From left-to-right we pick true or false depending on if we have A or $\neg A$.
From right-to-left we pick the function picking 0 or 1 depending on the boolean.
The identity on Bool is simple, for the other side we need to use the fact that a (resp. not) inhabited proposition is contractible and thus equal to 1 (resp. 0). □

Equivalence induction

Equivalences behave like identities.

In particular, we have an analogous of J called **equivalence induction**:

Proposition ([Uni13, Corollary 5.8.5])
Suppose given a property $P : \{A\, B : \mathcal{U}\} \to (A \simeq B) \to \mathcal{U}$.

If for every $A : \mathcal{U}$, *we have* $r_A : P\ \mathrm{id}_A$, *then for every* $e : A \simeq B$ *we have* $P\, e$.

Equivalence induction

Equivalences behave like identities.

In particular, we have an analogous of J called **equivalence induction**:

Proposition ([Uni13, Corollary 5.8.5])
Suppose given a property $P : \{A\, B : \mathcal{U}\} \to (A \simeq B) \to \mathcal{U}$.

If for every $A : \mathcal{U}$*, we have* $r_A : P\ \mathrm{id}_A$*, then for every* $e : A \simeq B$ *we have* $P\, e$.

Proof.
By path induction, we have $f : (p : A = B) \to P\,(\mathrm{idtoequiv}\ p)$, namely $f\ \mathrm{refl} \hat{=} r_A$.

Equivalence induction

Equivalences behave like identities.

In particular, we have an analogous of J called **equivalence induction**:

Proposition ([Uni13, Corollary 5.8.5])
Suppose given a property $P : \{A\, B : \mathcal{U}\} \to (A \simeq B) \to \mathcal{U}$.

If for every $A : \mathcal{U}$*, we have* $r_A : P\ \mathrm{id}_A$*, then for every* $e : A \simeq B$ *we have* $P\, e$.

Proof.
By path induction, we have $f : (p : A = B) \to P\,(\mathrm{idtoequiv}\ p)$, namely $f\ \mathrm{refl} \hat{=} r_A$.

We thus have $P\,(\mathrm{idtoequiv}\,(\mathrm{ua}\ e))$

Equivalence induction

Equivalences behave like identities.

In particular, we have an analogous of J called **equivalence induction**:

Proposition ([Uni13, Corollary 5.8.5])
Suppose given a property $P : \{A\, B : \mathcal{U}\} \to (A \simeq B) \to \mathcal{U}$.

If for every $A : \mathcal{U}$, *we have* $r_A : P\ \mathrm{id}_A$, *then for every* $e : A \simeq B$ *we have* $P\, e$.

Proof.
By path induction, we have $f : (p : A = B) \to P\,(\mathrm{idtoequiv}\ p)$, namely $f\ \mathrm{refl} \hat{=} r_A$.

We thus have $P\,(\mathrm{idtoequiv}\,(\mathrm{ua}\ e))$ and thus $P\, e$ by the computation rule for univalence. □

Bibliography i

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