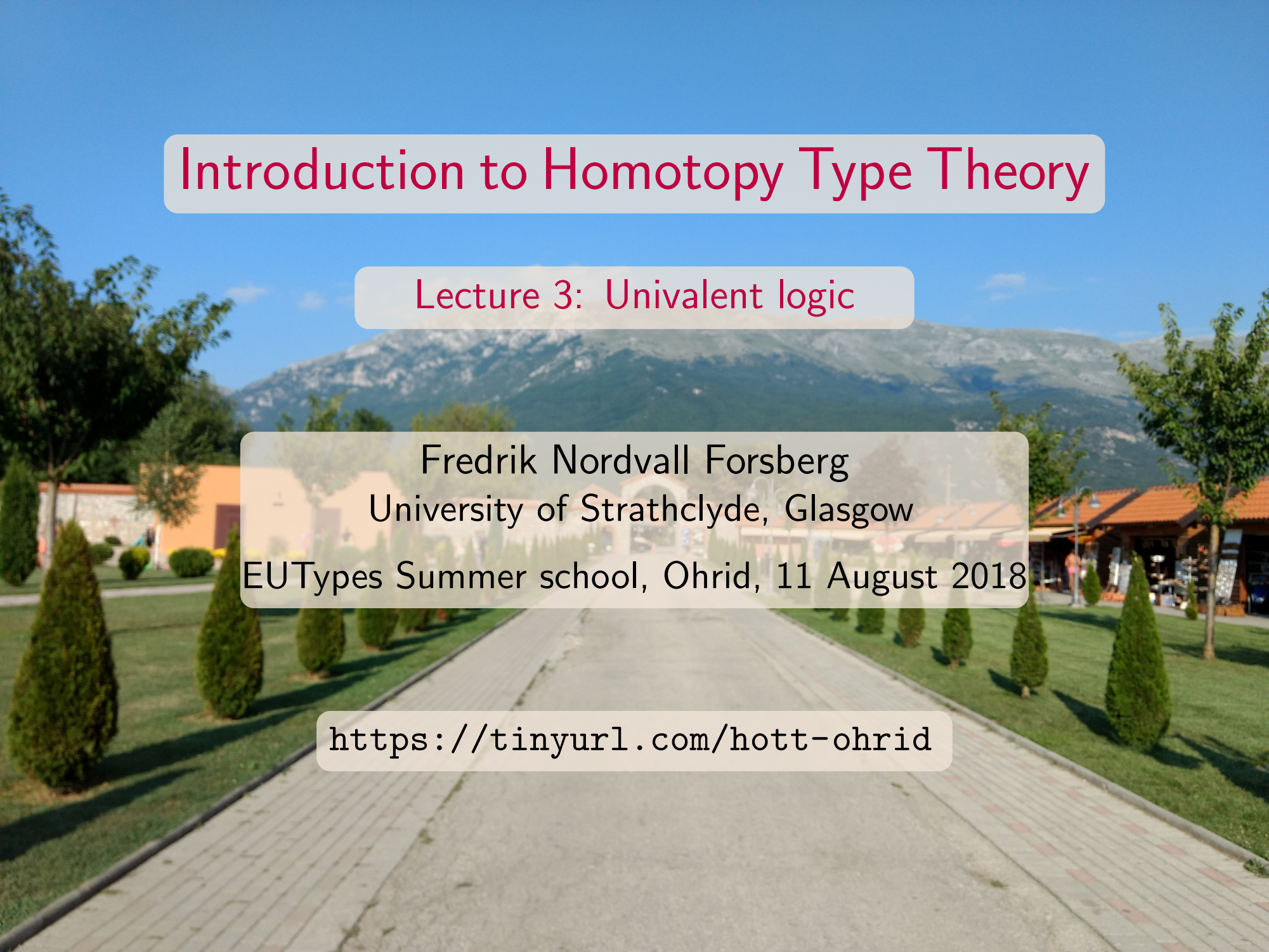


Introduction to Homotopy Type Theory

Lecture 3: Univalent logic

Fredrik Nordvall Forsberg
University of Strathclyde, Glasgow

EUTypes Summer school, Ohrid, 11 August 2018

`https://tinyurl.com/hott-ohrid`

Recap so far: paths

Operations on paths:

- $-^{-1} : a =_A b \to b =_A a$
- $\blacksquare : a =_A b \to b =_A c \to a =_A c$

Operations induced by paths:

- $\mathsf{transport}^B : a =_A a' \to B(a) \to B(a')$
- For $f : A \to B$, $\mathsf{ap}_f : a =_A a' \to f(a) =_B f(a')$

Recap so far: paths

Operations on paths:

- $-^{-1} : a =_A b \to b =_A a$
- $\blacksquare : a =_A b \to b =_A c \to a =_A c$

Operations induced by paths:

- $\mathsf{transport}^B : a =_A a' \to B(a) \to B(a')$
- For $f : A \to B$, $\mathsf{ap}_f : a =_A a' \to f(a) =_B f(a')$
- For $f : (\Pi x : A)B(x)$, $\mathsf{apd}_f : (\Pi p : a =_A a')(f(a) \underset{p}{=} f(a'))$

E.g. $b \underset{p}{=} b' \simeq \mathsf{transport}^B(p, b) =_{B(a')} b'$

Recap so far: the Univalence Axiom and hlevels

Univalence Axiom

The canonical function idtoeqv : $(A =_{\mathcal{U}} B) \to (A \simeq B)$ is an equivalence.

In particular, gives an inverse ua : $(A \simeq B) \to (A =_{\mathcal{U}} B)$.

Recap so far: the Univalence Axiom and hlevels

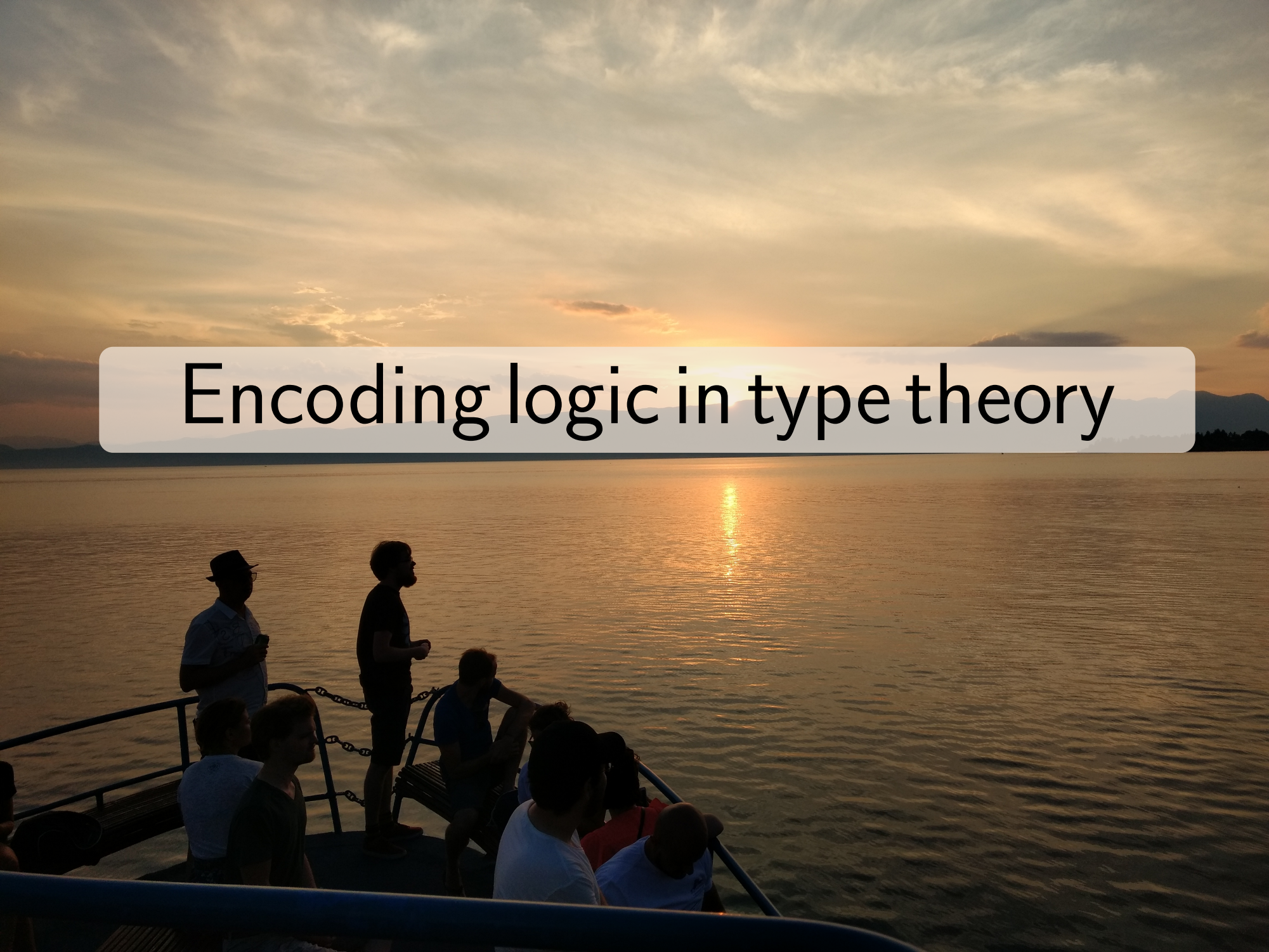
Univalence Axiom

The canonical function idtoeqv : $(A =_{\mathcal{U}} B) \to (A \simeq B)$ is an equivalence.

In particular, gives an inverse ua : $(A \simeq B) \to (A =_{\mathcal{U}} B)$.

Homotopy levels: Hierarchy of "complexity" of the type A:

$$\mathsf{isContr}(A) \to \mathsf{isProp}(A) \to \mathsf{isSet}(A) \to \ \ldots$$


Encoding logic in type theory

Encoding logic in type theory

Curry-Howard logic:

- Propositions are types
- Proofs are terms

Univalent logic:

- Propositions are. . . propositions (subsingleton types)
- Proofs are terms

Note: Univalent propositions are not the same as Coq's propositions!

Encoding logic in type theory

Curry-Howard logic:

- Propositions are types
- Proofs are terms

Univalent logic:

- Propositions are. . . propositions (subsingleton types)
- Proofs are terms

Note: Univalent propositions are not the same as Coq's propositions!

Both constructive by default, but allows the "axiomatic freedom" to assume additional classical (or anti-classical) principles.

Curry-Howard logic

Conjunction $A \wedge B \equiv A \times B$

A proof of $A \wedge B$ is a proof of A and a proof of B.

Disjunction $A \vee B \equiv A + B$

A proof of $A \wedge B$ is a proof of A or a proof of B.

Implication $A \Rightarrow B \equiv A \to B$

A proof of $A \Rightarrow B$ is a method for transforming proofs of A into proofs of B.

Negation $\neg A \equiv A \to \mathbf{0}$

A proof of $\neg A$ is a proof that there are no proofs of A.

Existential quantification $(\exists x : A)B(x) \equiv (\Sigma x : A)B(x)$

A proof of $(\exists x : A)B(x)$ is a witness $a : A$ and a proof of $B(a)$.

Universal quantification $(\forall x : A)B(x) \equiv (\Pi x : A)B(x)$

A proof of $(\forall x : A)B(x)$ is a method for proving $B(a)$ for any given $a : A$.

Example

What does the following type express?

$$(\Pi n : \mathbb{N})(\Sigma p : \mathbb{N})\big(\mathsf{isPrime}(p) \times (p > n)\big)$$

Example

What does the following type express?

$$(\Pi n : \mathbb{N})(\Sigma p : \mathbb{N})\big(\mathsf{isPrime}(p) \times (p > n)\big)$$

For every n, there is a prime greater than n — the infinitude of primes.

Example

What does the following type express?

$$(\Pi n : \mathbb{N})(\Sigma p : \mathbb{N})\big(\mathsf{isPrime}(p) \times (p > n)\big)$$

For every n, there is a prime greater than n — the infinitude of primes.

A proof of this is a function, which given n produces a natural number p, together with proofs that p is prime, and that $p > n$.

Assuming classical logic in Curry-Howard

The Law of Excluded Middle becomes $(\Pi A : \mathcal{U})(A + \neg A)$.

Of course, this is not provable, but we could assume it as an axiom.

Assuming classical logic in Curry-Howard

The Law of Excluded Middle becomes $(\Pi A : \mathcal{U})(A + \neg A)$.

Of course, this is not provable, but we could assume it as an axiom.

Double Negation Elimination becomes $(\Pi A : \mathcal{U})(\neg\neg A \to A)$.

Assuming classical logic in Curry-Howard

The Law of Excluded Middle becomes $(\Pi A : \mathcal{U})(A + \neg A)$.

Of course, this is not provable, but we could assume it as an axiom.

Double Negation Elimination becomes $(\Pi A : \mathcal{U})(\neg\neg A \to A)$.

Lemma (exercise)

$(\Pi A : \mathcal{U})(A + \neg A) \leftrightarrow (\Pi A : \mathcal{U})(\neg\neg A \to A)$.

Assuming classical logic in Curry-Howard

The Law of Excluded Middle becomes $(\Pi A : \mathcal{U})(A + \neg A)$.

Of course, this is not provable, but we could assume it as an axiom.

Double Negation Elimination becomes $(\Pi A : \mathcal{U})(\neg\neg A \to A)$.

Lemma (exercise)
$(\Pi A : \mathcal{U})(A + \neg A) \leftrightarrow (\Pi A : \mathcal{U})(\neg\neg A \to A)$.

- MLTT+LEM is consistent (by a model using classical logic).
- MLTT+UA is consistent (by the simplicial sets model).

But adding axioms is not modular. . .

MLTT+UA+LEM is inconsistent

Theorem
MLTT+UA+LEM proves $\mathbf{0}$.

Proof.
Let $f : (\Pi A : \mathcal{U})(\neg\neg A \to A)$ be given by LEM. We derive a contradiction from the fact that f must respect equivalences, by UA.

MLTT+UA+LEM is inconsistent

Theorem
MLTT+UA+LEM proves $\mathbf{0}$.

Proof.
Let $f : (\Pi A : \mathcal{U})(\neg\neg A \to A)$ be given by LEM. We derive a contradiction from the fact that f must respect equivalences, by UA.

Consider the non-identity equivalence $\mathsf{swap} : \mathbf{2} \to \mathbf{2}$. We have $\mathsf{ua}(\mathsf{swap}) : \mathbf{2} = \mathbf{2}$, and hence

$$\mathsf{apd}_f(\mathsf{ua}(\mathsf{swap})) : f(\mathbf{2}) \underset{\mathsf{ua}(\mathsf{swap})}{=} f(\mathbf{2})$$

MLTT+UA+LEM is inconsistent

Theorem
MLTT+UA+LEM proves $\mathbf{0}$.

Proof.
Let $f : (\Pi A : \mathcal{U})(\neg\neg A \to A)$ be given by LEM. We derive a contradiction from the fact that f must respect equivalences, by UA.

Consider the non-identity equivalence $\mathsf{swap} : \mathbf{2} \to \mathbf{2}$. We have $\mathsf{ua}(\mathsf{swap}) : \mathbf{2} = \mathbf{2}$, and hence

$$\mathsf{apd}_f(\mathsf{ua}(\mathsf{swap})) : f(\mathbf{2}) \underset{\mathsf{ua}(\mathsf{swap})}{=} f(\mathbf{2})$$

$$\simeq \mathsf{transport}^{z \mapsto \neg\neg z \to z}(\mathsf{ua}(\mathsf{swap}), f(\mathbf{2})) =_{\neg\neg\mathbf{2} \to \mathbf{2}} f(\mathbf{2})$$

Proof. (cont.)

$$\mathsf{transport}^{z \mapsto \neg\neg z \to z}(\mathsf{ua}(\mathsf{swap}), f(\mathbf{2})) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})$$

Proof. (cont.)

$$\mathsf{transport}^{z \mapsto \neg\neg z \to z}(\mathsf{ua}(\mathsf{swap}), f(\mathbf{2})) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})$$

Need to know three facts:

1. How $\mathsf{transport}^{z \mapsto B(z)\to C(z)}(p, -)$ works.
2. How $\mathsf{transport}^{z \mapsto \neg\neg z}(p, -)$ works.
3. How $\mathsf{transport}^{z \mapsto z}(p, -)$ works.

Transport in function spaces

Lemma

Let $p : x = y$ *and* $g : B(x) \to C(x)$.

$$\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, g) =$$

$$\{? : B(y) \to C(y)\}$$

Transport in function spaces

Lemma

Let $p : x = y$ *and* $g : B(x) \to C(x)$.

$$\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, g) =$$

$$\lambda(b : B(y)).\ \{? : C(y)\}$$

Transport in function spaces

Lemma

Let $p : x = y$ *and* $g : B(x) \to C(x)$.

$$\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, g) =$$

$$\lambda(b : B(y)).\, \mathsf{transport}^{C}(p, \{? : C(x)\})$$

Transport in function spaces

Lemma

Let $p : x = y$ *and* $g : B(x) \to C(x)$.

$$\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, g) =$$

$$\lambda(b : B(y)).\, \mathsf{transport}^{C}(p, g(\{? : B(x)\}))$$

Transport in function spaces

Lemma

Let $p : x = y$ *and* $g : B(x) \to C(x)$.

$$\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, g) =$$
$$\lambda(b : B(y)).\, \mathsf{transport}^{C}(p, g(\mathsf{transport}^{B}(p^{-1}, b)))$$

Transport in function spaces

Lemma

Let $p : x = y$ *and* $g : B(x) \to C(x)$.

$$\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, g) =$$
$$\lambda(b : B(y)).\, \mathsf{transport}^{C}(p, g(\mathsf{transport}^{B}(p^{-1}, b)))$$

Proof.

By path induction. □

Proof. (cont.)

$$\mathsf{transport}^{z \mapsto \neg\neg z \to z}(p, f(\mathbf{2})) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})$$

$p = \mathsf{ua}(\mathsf{swap})$

Need to know three facts:

1. How $\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, -)$ works. ✓
2. How $\mathsf{transport}^{z \mapsto \neg\neg z}(p, -)$ works.
3. How $\mathsf{transport}^{z \mapsto z}(p, -)$ works.

Proof. (cont.)

$$\mathsf{transport}^{z \mapsto \neg\neg z \to z}(p, f(\mathbf{2})) =_{\neg\neg\mathbf{2} \to \mathbf{2}} f(\mathbf{2}) \simeq$$
$$(\lambda u.\, \mathsf{transport}^{z \mapsto z}(p, f(\mathbf{2})(\mathsf{transport}^{z \to \neg\neg z}(p^{-1}, u))) =_{\neg\neg\mathbf{2} \to \mathbf{2}} f(\mathbf{2}))$$

$p = \mathsf{ua}(\mathsf{swap})$

Need to know three facts:

1. How $\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, -)$ works. ✓
2. How $\mathsf{transport}^{z \mapsto \neg\neg z}(p, -)$ works.
3. How $\mathsf{transport}^{z \mapsto z}(p, -)$ works.

Transport in $A \mapsto \neg\neg A$

Lemma

$\neg\neg A$ is a proposition. In particular, $\mathsf{transport}^{z \mapsto \neg\neg z}(p, u) = u$.

Transport in $A \mapsto \neg\neg A$

Lemma

$\neg\neg A$ is a proposition. In particular, $\mathsf{transport}^{z \mapsto \neg\neg z}(p, u) = u$.

Proof.

Let $u, v : \neg\neg A \equiv (A \to \mathbf{0}) \to \mathbf{0}$. We prove $u(x) =_{\mathbf{0}} v(x)$ for every $x : \neg A$.

Transport in $A \mapsto \neg\neg A$

Lemma

$\neg\neg A$ is a proposition. In particular, $\mathsf{transport}^{z \mapsto \neg\neg z}(p, u) = u$.

Proof.

Let $u, v : \neg\neg A \equiv (A \to \mathbf{0}) \to \mathbf{0}$. We prove $u(x) =_{\mathbf{0}} v(x)$ for every $x : \neg A$. This is easy: given x, we have $u(x) : \mathbf{0}$, so anything follows. □

Proof. (cont.)

$$(\mathsf{transport}^{z \mapsto \neg\neg z \to z}(p, f(\mathbf{2})) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})) \simeq$$
$$(\lambda u.\, \mathsf{transport}^{z \mapsto z}(p, f(\mathbf{2})(\mathsf{transport}^{z \to \neg\neg z}(p^{-1}, u))) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2}))$$

$p = \mathsf{ua}(\mathsf{swap})$

Need to know three facts:

1. How $\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, -)$ works. ✓
2. How $\mathsf{transport}^{z \mapsto \neg\neg z}(p^{-1}, -)$ works. ✓
3. How $\mathsf{transport}^{z \mapsto z}(p, -)$ works.

Proof. (cont.)

$$(\mathsf{transport}^{z \mapsto \neg\neg z \to z}(p, f(\mathbf{2})) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})) \simeq$$
$$(\lambda u.\, \mathsf{transport}^{z \mapsto z}(p, f(\mathbf{2})(\mathsf{transport}^{z \to \neg\neg z}(p^{-1}, u))) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})) \simeq$$
$$(\lambda u.\, \mathsf{transport}^{z \mapsto z}(p, f(\mathbf{2})(u)) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2}))$$

$p = \mathsf{ua}(\mathsf{swap})$

Need to know three facts:

1. How $\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, -)$ works. ✓
2. How $\mathsf{transport}^{z \mapsto \neg\neg z}(p^{-1}, -)$ works. ✓
3. How $\mathsf{transport}^{z \mapsto z}(p, -)$ works.

Transport in $A \mapsto A$ (coercion)

Let $p : A =_{\mathcal{U}} B$.

First attempt:

$$\mathsf{transport}^{z \mapsto z}(p, a) =_B \{? : B\}$$

Not much we can say... In particular

$$\mathsf{transport}^{z \mapsto z}(p, a) =_B aa$$

is ill-typed.

Transport in $A \mapsto A$ (coercion)

Let $p : A =_{\mathcal{U}} B$.

First attempt:

$$\mathsf{transport}^{z \mapsto z}(p, a) =_B \{? : B\}$$

Not much we can say... In particular

$$\cancel{\mathsf{transport}^{z \mapsto z}(p, a) =_B aa}$$

is ill-typed.

Transport in $A \mapsto A$ (coercion)

Let $p : A =_{\mathcal{U}} B$.

First attempt:

$$\mathsf{transport}^{z \mapsto z}(p, a) =_B \{? : B\}$$

Not much we can say... In particular

$$\cancel{\mathsf{transport}^{z \mapsto z}(p, a) =_B aa}$$

is ill-typed.

However, we only need to know

$$\mathsf{transport}^{z \mapsto z}(\mathsf{ua}(e), a) =_B \{? : B\}$$

for an equivalence $e : A \to B$.

Transport in $A \mapsto A$ (coercion)

Let $p : A =_{\mathcal{U}} B$.

First attempt:

$$\mathsf{transport}^{z \mapsto z}(p, a) =_B \{? : B\}$$

Not much we can say... In particular

$$\cancel{\mathsf{transport}^{z \mapsto z}(p, a) =_B aa}$$

is ill-typed.

However, we only need to know

$$\mathsf{transport}^{z \mapsto z}(\mathsf{ua}(e), a) =_B e(a)$$

for an equivalence $e : A \to B$.

Transport in $A \mapsto A$ (coercion)

Let $p : A =_{\mathcal{U}} B$.

First attempt:

$$\mathsf{transport}^{z \mapsto z}(p, a) =_B \{? : B\}$$

Not much we can say... In particular

$$\cancel{\mathsf{transport}^{z \mapsto z}(p, a) =_B aa}$$

is ill-typed.

However, we only need to know

$$\mathsf{transport}^{z \mapsto z}(\mathsf{ua}(e), a) =_B e(a)$$

for an equivalence $e : A \to B$.

This is one of the roundtrips of the Univalence Axiom!

Proof. (cont.)

$$(\mathsf{transport}^{z \mapsto \neg\neg z \to z}(p, f(\mathbf{2})) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})) \simeq$$
$$(\lambda u.\, \mathsf{transport}^{z \mapsto z}(p, f(\mathbf{2})(\mathsf{transport}^{z \to \neg\neg z}(p^{-1}, u))) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})) \simeq$$
$$(\lambda u.\, \mathsf{transport}^{z \mapsto z}(p, f(\mathbf{2})(u)) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2}))$$

$p = \mathsf{ua}(\mathsf{swap})$

Need to know three facts:

1. How $\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, -)$ works. ✓
2. How $\mathsf{transport}^{z \mapsto \neg\neg z}(p^{-1}, -)$ works. ✓
3. How $\mathsf{transport}^{z \mapsto z}(\mathsf{ua}(e), -)$ works. ✓

Proof. (cont.)

$$(\mathsf{transport}^{z \mapsto \neg\neg z \to z}(p, f(\mathbf{2})) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})) \simeq$$
$$(\lambda u.\, \mathsf{transport}^{z \mapsto z}(p, f(\mathbf{2})(\mathsf{transport}^{z \to \neg\neg z}(p^{-1}, u))) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})) \simeq$$
$$(\lambda u.\, \mathsf{transport}^{z \mapsto z}(p, f(\mathbf{2})(u)) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})) \simeq$$
$$\lambda u.\, \mathsf{swap}(f(\mathbf{2})(u)) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})$$

$p = \mathsf{ua}(\mathsf{swap})$

Need to know three facts:

1. How $\mathsf{transport}^{z \mapsto B(z) \to C(z)}(p, -)$ works. ✓
2. How $\mathsf{transport}^{z \mapsto \neg\neg z}(p^{-1}, -)$ works. ✓
3. How $\mathsf{transport}^{z \mapsto z}(\mathsf{ua}(e), -)$ works. ✓

Finishing the proof

Proof. (cont.)

From the assumption that $f : (\Pi A : \mathcal{U})(\neg\neg A \to A)$, we derived

$$\lambda u.\, \mathsf{swap}(f(\mathbf{2})(u)) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})$$

Finishing the proof

Proof. (cont.)

From the assumption that $f : (\Pi A : \mathcal{U})(\neg\neg A \to A)$, we derived

$$\lambda u.\, \mathsf{swap}(f(\mathbf{2})(u)) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})$$

In other words, $\mathsf{swap}(f(\mathbf{2})(t)) =_{\mathbf{2}} f(\mathbf{2})(t)$ for any $t : \neg\neg\mathbf{2}$.

Finishing the proof

Proof. (cont.)

From the assumption that $f : (\Pi A : \mathcal{U})(\neg\neg A \to A)$, we derived

$$\lambda u.\, \mathsf{swap}(f(\mathbf{2})(u)) =_{\neg\neg\mathbf{2}\to\mathbf{2}} f(\mathbf{2})$$

In other words, $\mathsf{swap}(f(\mathbf{2})(t)) =_{\mathbf{2}} f(\mathbf{2})(t)$ for any $t : \neg\neg\mathbf{2}$.

But $\mathsf{swap}(x) \neq_{\mathbf{2}} x$ for every $x : \mathbf{2}$ by definition. This gives a contradiction. □

Propositions to the rescue

Conclusion: we cannot use Curry-Howard logic in HoTT if we want to have the freedom to assume classical logic.

To be clear: Curry-Howard logic is consistent with HoTT. But our options to extend it to classical logic are limited.

Propositions to the rescue

Conclusion: we cannot use Curry-Howard logic in HoTT if we want to have the freedom to assume classical logic.

To be clear: Curry-Howard logic is consistent with HoTT. But our options to extend it to classical logic are limited.

Instead we will encode logic not using arbitrary types, but using propositions only.

Propositions to the rescue

Conclusion: we cannot use Curry-Howard logic in HoTT if we want to have the freedom to assume classical logic.

To be clear: Curry-Howard logic is consistent with HoTT. But our options to extend it to classical logic are limited.

Instead we will encode logic not using arbitrary types, but using propositions only.

Recall:

Definition
A type A is a proposition (subsingleton) if we can prove

$$\mathsf{isProp}(A) :\equiv (\Pi x, y : A)(x =_A y)$$

A proposition

$$(\Sigma n : \mathbb{N})\big(\mathsf{isOdd}(n) \times (\Pi a, b : \mathbb{N})(n! \neq_{\mathbb{N}} a^2 - b^2)\big)$$

A proposition

$$(\Sigma n : \mathbb{N})\big(\mathsf{isOdd}(n) \times (\Pi a, b : \mathbb{N})(n! \neq_{\mathbb{N}} a^2 - b^2)\big)$$

Propositions are not "proof-irrelevant"; they contain computational content.

A proposition

$$(\Sigma n : \mathbb{N})\big(\mathsf{isOdd}(n) \times (\Pi a, b : \mathbb{N})(n! \neq_{\mathbb{N}} a^2 - b^2)\big)$$

Propositions are not "proof-irrelevant"; they contain computational content.

From any proof of the above proposition, the number 3 can be extracted.

A proposition

$$(\Sigma n : \mathbb{N})\big(\mathsf{isOdd}(n) \times (\Pi a, b : \mathbb{N})(n! \neq_{\mathbb{N}} a^2 - b^2)\big)$$

Propositions are not "proof-irrelevant"; they contain computational content.

From any proof of the above proposition, the number 3 can be extracted.

Another example: The type $\mathsf{isEquiv}(f)$ is always a proposition for any $f : A \to B$. From any proof of it, a function $g : B \to A$ can be extracted.

Closure of propositions under logical connectives

Theorem

- $\mathrm{isProp}(\mathbf{1})$.
- $\mathrm{isProp}(\mathbf{0})$.
- *If* $\mathrm{isProp}(P)$ *and* $\mathrm{isProp}(Q)$ *then* $\mathrm{isProp}(P \times Q)$.
- *If* $\mathrm{isProp}(Q)$ *then* $\mathrm{isProp}(P \to Q)$.
- *If* $\mathrm{isProp}(P(x))$ *for every* $x : A$, *then* $\mathrm{isProp}((\Pi x : A)P(x))$.

The two missing cases

The two missing cases

Propositions are not closed under $+$:

The two missing cases

Propositions are not closed under $+$:

$\mathbf{1}$ is a proposition, but $\mathbf{1} + \mathbf{1}$ is not.

The two missing cases

Propositions are not closed under $+$:

$\mathbf{1}$ is a proposition, but $\mathbf{1} + \mathbf{1}$ is not.

Propositions are not closed under Σ:

The two missing cases

Propositions are not closed under $+$:

$\mathbf{1}$ is a proposition, but $\mathbf{1} + \mathbf{1}$ is not.

Propositions are not closed under Σ:

$\mathbf{1}$ is a proposition, but $(\Sigma x : \mathbf{2})\mathbf{1}$ is not.

The two missing cases

Propositions are not closed under $+$:

$$\mathbf{1} \text{ is a proposition, but } \mathbf{1} + \mathbf{1} \text{ is not.}$$

Propositions are not closed under Σ:

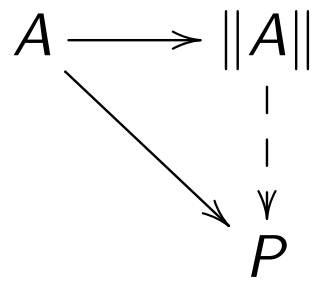
$$\mathbf{1} \text{ is a proposition, but } (\Sigma x : \mathbf{2})\mathbf{1} \text{ is not.}$$

We need a "best possible way" to map these types to propositions.

Propositional truncation

A propositional truncation $\|A\|$ of a type A is a universal solution to mapping A into a proposition, in the following sense:

arbitrary type



proposition

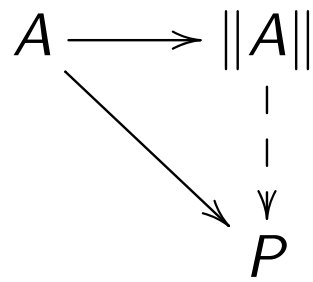
proposition

$\|A\|$ is the smallest proposition A maps into.

Propositional truncation

A propositional truncation $\|A\|$ of a type A is a universal solution to mapping A into a proposition, in the following sense:

arbitrary type



proposition

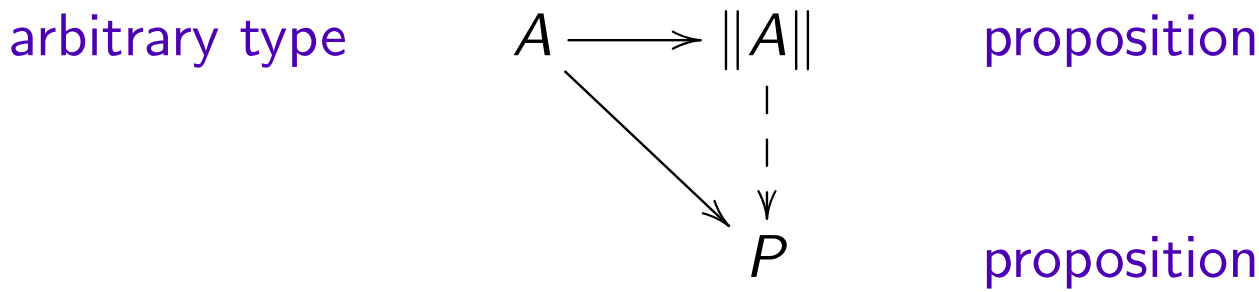
proposition

$\|A\|$ is the smallest proposition A maps into.

Some types can be shown to be propositional truncations of other types.

Propositional truncation

A propositional truncation $\|A\|$ of a type A is a universal solution to mapping A into a proposition, in the following sense:



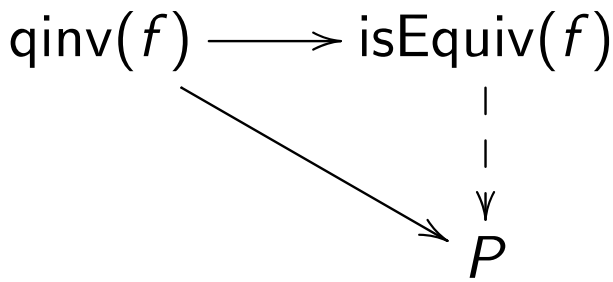
$\|A\|$ is the smallest proposition A maps into.

Some types can be shown to be propositional truncations of other types.

But usually the theory is extended to get propositional truncations for all types.

Example

not always proposition

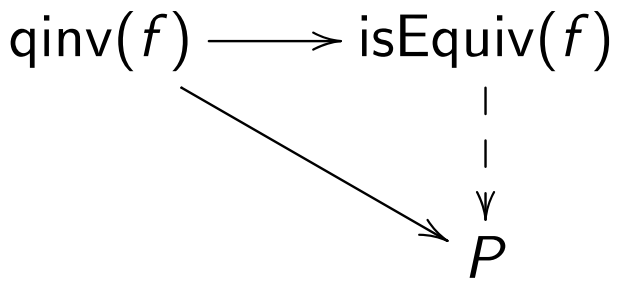


proposition

proposition

Example

not always proposition



proposition

proposition

isEquiv(f) is a canonical way of making qinv(f) into a proposition.

Propositional truncation in rule form

Propositional truncation in rule form

Formation If A is a type then $\|A\|$ is a type.

Propositional truncation in rule form

Formation If A is a type then $\|A\|$ is a type.

Introduction If $a : A$ then $|a| : \|A\|$.
Gives map $|-| : A \to \|A\|$.
If $x, y : \|A\|$ then $x =_{\|A\|} y$.
Makes $\|A\|$ a proposition.

Propositional truncation in rule form

Formation If A is a type then $\|A\|$ is a type.

Introduction If $a : A$ then $|a| : \|A\|$.
Gives map $|-| : A \to \|A\|$.
If $x, y : \|A\|$ then $x =_{\|A\|} y$.
Makes $\|A\|$ a proposition.

Elimination If P is a proposition and $g : A \to P$, then there is $\bar{g} : \|A\| \to P$.

Exercise: State and show that a more traditional dependent elimination rule is equivalent to the given rule.

Propositional truncation in rule form

Formation If A is a type then $\|A\|$ is a type.

Introduction If $a : A$ then $|a| : \|A\|$.
Gives map $|-| : A \to \|A\|$.
If $x, y : \|A\|$ then $x =_{\|A\|} y$.
Makes $\|A\|$ a proposition.

Elimination If P is a proposition and $g : A \to P$, then there is $\bar{g} : \|A\| \to P$.

Computation If $a : A$ then $\bar{g}(|a|) \equiv g(a) : P$.

Exercise: State and show that a more traditional dependent elimination rule is equivalent to the given rule.

Propositional truncation in rule form

Formation If A is a type then $\|A\|$ is a type.

Introduction If $a : A$ then $|a| : \|A\|$.
Gives map $|-| : A \to \|A\|$.
If $x, y : \|A\|$ then $x =_{\|A\|} y$.
Makes $\|A\|$ a proposition.

Elimination If P is a proposition and $g : A \to P$, then there is $\bar{g} : \|A\| \to P$.

Computation If $a : A$ then $\bar{g}(|a|) \equiv g(a) : P$.

Exercise: State and show that a more traditional dependent elimination rule is equivalent to the given rule.

Tomorrow: can be defined as a higher inductive type.

Intuitive meaning of the elimination rule

Elimination If P is a proposition and $g : A \to P$, then there is $\bar{g} : \|A\| \to P$.

Intuitive meaning of the elimination rule

Elimination If P is a proposition and $g : A \to P$, then there is $\bar{g} : \|A\| \to P$.

"If we are mapping from $\|A\|$ into a proposition, we can forget about the $\| - \|$."

Intuitive meaning of the elimination rule

Elimination If P is a proposition and $g : A \to P$, then there is $\bar{g} : \|A\| \to P$.

"If we are mapping from $\|A\|$ into a proposition, we can forget about the $\| - \|$."

Note: From $\|A\|$ we can still construct maps into non-propositions, using cleverness to factor through a proposition. Two examples:

Intuitive meaning of the elimination rule

Elimination If P is a proposition and $g : A \to P$, then there is $\bar{g} : \|A\| \to P$.

"If we are mapping from $\|A\|$ into a proposition, we can forget about the $\| - \|$."

Note: From $\|A\|$ we can still construct maps into non-propositions, using cleverness to factor through a proposition. Two examples:

1. If we can describe the codomain precisely, we can factor

$$\|A\| \to (\Sigma x : B)(x =_B b_0) \to B$$

Intuitive meaning of the elimination rule

Elimination If P is a proposition and $g : A \to P$, then there is $\bar{g} : \|A\| \to P$.

"If we are mapping from $\|A\|$ into a proposition, we can forget about the $\| - \|$."

Note: From $\|A\|$ we can still construct maps into non-propositions, using cleverness to factor through a proposition. Two examples:

1. If we can describe the codomain precisely, we can factor

$$\|A\| \to (\Sigma x : B)(x =_B b_0) \to B$$

2. If there is a canonical choice, e.g. for a decidable family $P : \mathbb{N} \to \text{Prop}$, we can factor

$$\|A\| \to (\Sigma x : \mathbb{N})(P(n) \times "x \text{ is least sat. } P") \to (\Sigma x : \mathbb{N})P(n)$$

Univalent logic

Curry-Howard logic, except:

Disjunction $A \vee B \equiv \|A + B\|$

Existential quantification $(\exists x : A)B(x) \equiv \|(\Sigma x : A)B(x)\|$

Univalent logic

Curry-Howard logic, except:

Disjunction $A \vee B \equiv \|A + B\|$

Existential quantification $(\exists x : A)B(x) \equiv \|(\Sigma x : A)B(x)\|$

Consequence: All formulas are interpreted as propositions.
Since atomic formulas are propositions, and all connectives now preserve propositions.

Σ vs $\exists$: Unexpected Curry-Howard images

Image for a function $f : A \to B$ translated into Curry-Howard:

$$\mathsf{image}_{CH}\, f :\equiv (\Sigma y : B)(\Sigma x : A)(f\, x =_B y)$$

Σ vs $\exists$: Unexpected Curry-Howard images

Image for a function $f : A \to B$ translated into Curry-Howard:

$$\mathsf{image}_{CH}\, f :\equiv (\Sigma y : B)(\Sigma x : A)(f\, x =_B y)$$

Unexpected: $\mathsf{image}_{CH}\, f \simeq A$ (by vacuum cord principle).

Σ vs $\exists$: Unexpected Curry-Howard images

Image for a function $f : A \to B$ translated into Curry-Howard:

$$\mathsf{image}_{CH}\, f :\equiv (\Sigma y : B)(\Sigma x : A)(f\, x =_B y)$$

Unexpected: $\mathsf{image}_{CH}\, f \simeq A$ (by vacuum cord principle).
E.g. would expect the image of the unique function $\mathbb{N} \to \mathbf{1}$ to be $\mathbf{1}$, not $\mathbb{N}$.

Σ vs $\exists$: Unexpected Curry-Howard images

Image for a function $f : A \to B$ translated into Curry-Howard:

$$\mathsf{image}_{CH}\, f :\equiv (\Sigma y : B)(\Sigma x : A)(f\, x =_B y)$$

Unexpected: $\mathsf{image}_{CH}\, f \simeq A$ (by vacuum cord principle).
E.g. would expect the image of the unique function $\mathbb{N} \to \mathbf{1}$ to be $\mathbf{1}$, not $\mathbb{N}$.

Instead we should define

$$\mathsf{image}\, f :\equiv (\Sigma y : B)\|(\Sigma x : A)(f\, x =_B y)\|$$

Σ vs $\exists$: Unexpected Curry-Howard images

Image for a function $f : A \to B$ translated into Curry-Howard:

$$\mathrm{image}_{CH}\, f :\equiv (\Sigma y : B)(\Sigma x : A)(f\, x =_B y)$$

Unexpected: $\mathrm{image}_{CH}\, f \simeq A$ (by vacuum cord principle).
E.g. would expect the image of the unique function $\mathbb{N} \to \mathbf{1}$ to be $\mathbf{1}$, not $\mathbb{N}$.

Instead we should define

$$\mathrm{image}\, f :\equiv (\Sigma y : B)\|(\Sigma x : A)(f\, x =_B y)\|$$

Similarly, the right notion of surjectivity is

$$\mathrm{isSurjective}(f) :\equiv (\Pi b : B)\|(\Sigma x : A)(f\, x =_B y)\|$$

— with untruncated Σ, every "surjection" would come with a choice of an inverse.

Classical principles in univalent logic

The right formulation of the Law of Excluded Middle in univalent logic is now $(\prod P : \mathcal{U})(\mathsf{isProp}(P) \to P + \neg P)$.

Classical principles in univalent logic

The right formulation of the Law of Excluded Middle in univalent logic is now $(\prod P : \mathcal{U})(\mathsf{isProp}(P) \to P + \neg P)$.

Exercise: Show that if $\mathsf{isProp}(A)$ and $\mathsf{isProp}(B)$ and $\neg(A \times B)$, then $\mathsf{isProp}(A + B)$. In particular, no truncation is needed around $P + \neg P$.

Classical principles in univalent logic

The right formulation of the Law of Excluded Middle in univalent logic is now $(\prod P : \mathcal{U})(\mathsf{isProp}(P) \to P + \neg P)$.

Exercise: Show that if $\mathsf{isProp}(A)$ and $\mathsf{isProp}(B)$ and $\neg(A \times B)$, then $\mathsf{isProp}(A + B)$. In particular, no truncation is needed around $P + \neg P$.

And the right notion of Double Negation Elimination is $(\prod P : \mathcal{U})(\mathsf{isProp}(P) \to \neg\neg P \to P)$.

Classical principles in univalent logic

The right formulation of the Law of Excluded Middle in univalent logic is now $(\prod P : \mathcal{U})(\mathsf{isProp}(P) \to P + \neg P)$.

Exercise: Show that if $\mathsf{isProp}(A)$ and $\mathsf{isProp}(B)$ and $\neg(A \times B)$, then $\mathsf{isProp}(A + B)$. In particular, no truncation is needed around $P + \neg P$.

And the right notion of Double Negation Elimination is $(\prod P : \mathcal{U})(\mathsf{isProp}(P) \to \neg\neg P \to P)$.

These axioms are not provable, but they are consistent with UA (since they are true in the simplicial sets model).

The Axiom of Choice

The type-theoretic Axiom of Choice

$$\big((\Pi x : A)(\Sigma y : B) R\, x\, y\big) \to (\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

is provable.

The Axiom of Choice

The type-theoretic Axiom of Choice

$$\big((\Pi x : A)(\Sigma y : B) R\, x\, y\big) \simeq (\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

is provable.

The Axiom of Choice

The type-theoretic Axiom of Choice

$$\big((\Pi x : A)(\Sigma y : B) R\, x\, y\big) \simeq (\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

is provable.

Exercise: Prove this.

The Axiom of Choice

The type-theoretic Axiom of Choice

$$\big((\Pi x : A)(\Sigma y : B) R\, x\, y\big) \simeq (\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

is provable.

Exercise: Prove this.

This is strange, because the Axiom of Choice is usually considered highly non-constructive.

The Axiom of Choice

The type-theoretic Axiom of Choice

$$\big((\Pi x : A)(\Sigma y : B) R\, x\, y\big) \simeq (\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

is provable.

Exercise: Prove this.

This is strange, because the Axiom of Choice is usually considered highly non-constructive.

Indeed, in univalent logic things are different:

$$\big((\Pi x : A)\|(\Sigma y : B) R\, x\, y\big)\| \to \|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

Here A and B are sets, and $R : A \to B \to \mathrm{Prop}$.

The Axiom of Choice

The type-theoretic Axiom of Choice

$$\big((\Pi x : A)(\Sigma y : B) R\, x\, y\big) \simeq (\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

is provable.

Exercise: Prove this.

This is strange, because the Axiom of Choice is usually considered highly non-constructive.

Indeed, in univalent logic things are different:

$$\big((\Pi x : A)\|(\Sigma y : B) R\, x\, y\big)\| \to \|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

Here A and B are sets, and $R : A \to B \to \mathrm{Prop}$.

This axiom is consistent with UA (simplicial sets model again).

The univalent Axiom of Choice is non-constructive

Theorem (after Diaconescu [1975])

The univalent Axiom of Choice

$$\big((\Pi x : A)\|(\Sigma y : B) R\, x\, y\big)\| \to \|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

implies the univalent Law of Excluded Middle.

The univalent Axiom of Choice is non-constructive

Theorem (after Diaconescu [1975])

The univalent Axiom of Choice

$$\big((\Pi x : A)\|(\Sigma y : B) R\, x\, y\big)\| \to \|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

implies the univalent Law of Excluded Middle.

Proof.

We assume AC and instantiate

$$\begin{aligned} A &:\equiv (\Sigma Q : \mathbf{2} \to \mathrm{Prop})\|(\Sigma b : \mathbf{2}) Q(b)\| \\ B &:\equiv \mathbf{2} \\ R\,(Q, q)\, b &:\equiv Q(b) \end{aligned}$$

The univalent Axiom of Choice is non-constructive

Theorem (after Diaconescu [1975])

The univalent Axiom of Choice

$$\big((\Pi x : A)\|(\Sigma y : B) R\, x\, y\big)\| \to \|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

implies the univalent Law of Excluded Middle.

Proof.

We assume AC and instantiate

$$\begin{aligned} A &:\equiv (\Sigma Q : \mathbf{2} \to \mathrm{Prop})\|(\Sigma b : \mathbf{2}) Q(b)\| \\ B &:\equiv \mathbf{2} \\ R\,(Q, q)\, b &:\equiv Q(b) \end{aligned}$$

The premise of AC says "every inhabited predicate is inhabited", which is clearly true (proof term $\lambda(Q, q).\, q$).

Proof. (cont.)

Hence we get

$$\|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

Proof. (cont.)

Hence we get

$$\|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

From this, we are trying to prove the proposition $P + \neg P$, so we can forget about the truncation:

$$(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

Proof. (cont.)

Hence we get

$$\|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

From this, we are trying to prove the proposition $P + \neg P$, so we can forget about the truncation:

$$(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

Consider the inhabited predicates $T, F : \mathbf{2} \to \mathrm{Prop}$

$$T\, b :\equiv \|(b =_{\mathbf{2}} \mathsf{true}) + P\|$$
$$F\, b :\equiv \|(b =_{\mathbf{2}} \mathsf{false}) + P\|$$

Proof. (cont.)

Hence we get

$$\|(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))\|$$

From this, we are trying to prove the proposition $P + \neg P$, so we can forget about the truncation:

$$(\Sigma f : A \to B)(\Pi x : A)(R\, x\, (f\, x))$$

Consider the inhabited predicates $T, F : \mathbf{2} \to \mathrm{Prop}$

$$T\, b :\equiv \|(b =_{\mathbf{2}} \mathsf{true}) + P\|$$
$$F\, b :\equiv \|(b =_{\mathbf{2}} \mathsf{false}) + P\|$$

There is $f : A \to \mathbf{2}$ such that $T\, (f\ T)$ and $F\, (f\ F)$.

Proof. (cont.)

$\mathbf{2}$ has decidable equality, so there are four possibilities:

$f\ T$	$f\ F$	
false	false	$T(f\ T) \equiv \|\mathsf{false} = \mathsf{true} + P\| \simeq \|\mathbf{0} + P\| \simeq P$
false	true	$F(f\ F) \equiv \|\mathsf{true} = \mathsf{false} + P\| \simeq \|\mathbf{0} + P\| \simeq P$
true	false	$T(f\ T) \simeq \mathbf{1} \simeq F(f\ F)$
true	true	$F(f\ F) \equiv \|\mathsf{true} = \mathsf{false} + P\| \simeq \|\mathbf{0} + P\| \simeq P$

Proof. (cont.)

$\mathbf{2}$ has decidable equality, so there are four possibilities:

$f\ T$	$f\ F$	
false	false	$T(f\ T) \equiv \|\mathsf{false} = \mathsf{true} + P\| \simeq \|\mathbf{0} + P\| \simeq P$
false	true	$F(f\ F) \equiv \|\mathsf{true} = \mathsf{false} + P\| \simeq \|\mathbf{0} + P\| \simeq P$
true	false	$T(f\ T) \simeq \mathbf{1} \simeq F(f\ F)$
true	true	$F(f\ F) \equiv \|\mathsf{true} = \mathsf{false} + P\| \simeq \|\mathbf{0} + P\| \simeq P$

In three cases we clearly have P. If $f\ T \equiv \mathsf{true}$ and $f\ F \equiv \mathsf{false}$, we prove $\neg P$:

Proof. (cont.)

$\mathbf{2}$ has decidable equality, so there are four possibilities:

$f\ T$	$f\ F$	
false	false	$T(f\ T) \equiv \|\mathsf{false} = \mathsf{true} + P\| \simeq \|\mathbf{0} + P\| \simeq P$
false	true	$F(f\ F) \equiv \|\mathsf{true} = \mathsf{false} + P\| \simeq \|\mathbf{0} + P\| \simeq P$
true	false	$T(f\ T) \simeq \mathbf{1} \simeq F(f\ F)$
true	true	$F(f\ F) \equiv \|\mathsf{true} = \mathsf{false} + P\| \simeq \|\mathbf{0} + P\| \simeq P$

In three cases we clearly have P. If $f\ T \equiv \mathsf{true}$ and $f\ F \equiv \mathsf{false}$, we prove $\neg P$:

Assume P. Then $T\ b \simeq F\ b$ for every $b : \mathbf{2}$, hence $T = F$ by univalence.

Proof. (cont.)

$\mathbf{2}$ has decidable equality, so there are four possibilities:

$f\ T$	$f\ F$	
false	false	$T(f\ T) \equiv \|\mathsf{false} = \mathsf{true} + P\| \simeq \|\mathbf{0} + P\| \simeq P$
false	true	$F(f\ F) \equiv \|\mathsf{true} = \mathsf{false} + P\| \simeq \|\mathbf{0} + P\| \simeq P$
true	false	$T(f\ T) \simeq \mathbf{1} \simeq F(f\ F)$
true	true	$F(f\ F) \equiv \|\mathsf{true} = \mathsf{false} + P\| \simeq \|\mathbf{0} + P\| \simeq P$

In three cases we clearly have P. If $f\ T \equiv \mathsf{true}$ and $f\ F \equiv \mathsf{false}$, we prove $\neg P$:

Assume P. Then $T\ b \simeq F\ b$ for every $b : \mathbf{2}$, hence $T = F$ by univalence.

But if $T = F$ then $f\ T = f\ F$, i.e. $\mathsf{true} = \mathsf{false}$, a contradiction. Hence we have proven $\neg P$.

Hence in each case we have proven $P + \neg P$. □

Exercises

1. Prove $(\Pi A : \mathcal{U})(A + \neg A) \leftrightarrow (\Pi A : \mathcal{U})(\neg\neg A \to A)$.

2. State and show that a more traditional dependent elimination rule is equivalent to the given elimination rule for propositional truncation on slide 22.

3. Show that if isProp(A) and isProp(B) and $\neg(A \times B)$, then isProp$(A + B)$. In particular, no truncation is needed around $P + \neg P$ in the statement of univalent LEM.

4. Prove

$$\big((\Pi x : A)(\Sigma y : B)R\,x\,y\big) \simeq (\Sigma f : A \to B)(\Pi x : A)(R\,x\,(f\,x))$$

5. Show that
$(\Sigma x : \mathbb{N})(f(x) = 0 \times ((\Pi y : \mathbb{N})(f(y) = 0 \to x \leq y))$ is the propositional truncation of $(\Sigma x : \mathbb{N})f(x) = 0$.

Summary

Univalent logic: propositions as subsingleton types.

Propositional truncation: used to interpret $\vee$ and $\exists$.

Makes it possible to assume non-constructive principles such as the Law of Excluded Middle and the Axiom of Choice without giving up the Univalence Axiom.

Σ used for structure, $\exists$ for properties. In general an art to decide which one to use.

Coming up tomorrow:

- Higher Inductive Types
- Synthetic homotopy theory

References


N. Kraus, M. Escardó, T. Coquand and T. Altenkirch
Notions of anonymous existence in Martin-Löf Type Theory
Logical Methods in Computer Science, Vol. 13(1:15)2017, pp. 1–36, 2017


A. Booij, M. Escardó, P. Lumsdaine, M. Shulman
Parametricity, automorphisms of the universe, and excluded middle
To appear in the post-proceedings of TYPES 2016


R. Diaconescu
Axiom of choice and complementation
Proceedings of AMS, 51:176–178, 1975