H-Levels and Truncation

Thomas Somers & Niek Terol

7 December 2022

Outline

Set

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$. More precisely, the type A is a set if the following proposition is inhabited:

$$\texttt{isSet}(A) :\equiv \prod_{(x,y:A)} \prod_{(p,q:x=y)} (p = q).$$

Set

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$. More precisely, the type A is a set if the following proposition is inhabited:

$$\mathtt{isSet}(A) :\equiv \prod_{(x,y:A)} \prod_{(p,q:x=y)} (p = q).$$

- Sets in Homotopic Type Theory do not have a global membership predicate $\in$.

Set

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.
More precisely, the type A is a set if the following proposition is inhabited:

$$\texttt{isSet}(A) :\equiv \prod_{(x,y:A)} \prod_{(p,q:x=y)} (p = q).$$

- Sets in Homotopic Type Theory do not have a global membership predicate $\in$.

Example

The type $\mathbf{1}$ is a set, as for any $x, y : \mathbf{1}$, we have that $x = y \simeq \mathbf{1}$.
As for all $p', q' : \mathbf{1}, p' = q'$, we have that for all $p, q : x = y, p = q$.

More Sets

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.

Example

If A and B are sets, then $A \times B$ is also a set.

More Sets

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.

Example

If A and B are sets, then $A \times B$ is also a set.

Proof.

For any $x, y : A \times B$ and $p, q : x = y$, it holds that:
$p = pair_{=}(ap_{pr_1}(p), ap_{pr_2}(p))$ and $q = pair_{=}(ap_{pr_1}(q), ap_{pr_2}(q))$.

More Sets

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.

Example

If A and B are sets, then $A \times B$ is also a set.

Proof.

For any $x, y : A \times B$ and $p, q : x = y$, it holds that: $p = pair_{=}(ap_{pr_1}(p), ap_{pr_2}(p))$ and $q = pair_{=}(ap_{pr_1}(q), ap_{pr_2}(q))$. As A is a set, $ap_{pr_1}(p) = ap_{pr_1}(q)$, and as B is a set, $ap_{pr_2}(p) = ap_{pr_2}(q)$, thus $p = q$. □

More Sets

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.

Example

The type $\mathbf{0}$ is a set.

Example

The type $\mathbb{N}$ is a set, as for all $m, n : \mathbb{N}$, either $m = n \simeq \mathbf{0}$ or $m = n \simeq \mathbf{1}$.

Not all types are Sets

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.

Example

The type $\mathcal{U}$ is not a set.

Not all types are Sets

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.

Example

The type $\mathcal{U}$ is not a set.

Proof.

Take $A :\equiv 2$ in $\mathcal{U}$ and define $f : \mathbf{2} \to \mathbf{2}$ by $f(1_\mathbf{2}) :\equiv f(0_\mathbf{2})$ and $f(0_\mathbf{2}) :\equiv f(1_\mathbf{2})$.

Not all types are Sets

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.

Example

The type $\mathcal{U}$ is not a set.

Proof.

Take $A :\equiv 2$ in $\mathcal{U}$ and define $f : \mathbf{2} \to \mathbf{2}$ by $f(1_\mathbf{2}) :\equiv f(0_\mathbf{2})$ and $f(0_\mathbf{2}) :\equiv f(1_\mathbf{2})$.
Then f is an equivalence as $f(f(x)) = x$. By univalence f leads to a path $p : \mathbf{2} = \mathbf{2}$, and $refl_\mathbf{2} : \mathbf{2} = \mathbf{2}$ is also a path.

Not all types are Sets

Definition (3.1.3)

A type A is a set if for all $x, y : A$ and for all $p, q : x =_A y$, we have $p = q$.

Example

The type $\mathcal{U}$ is not a set.

Proof.

Take $A :\equiv 2$ in $\mathcal{U}$ and define $f : \mathbf{2} \to \mathbf{2}$ by $f(1_\mathbf{2}) :\equiv f(0_\mathbf{2})$ and $f(0_\mathbf{2}) :\equiv f(1_\mathbf{2})$.
Then f is an equivalence as $f(f(x)) = x$. By univalence f leads to a path $p : \mathbf{2} = \mathbf{2}$, and $refl_\mathbf{2} : \mathbf{2} = \mathbf{2}$ is also a path.
If $p = refl_\mathbf{2}$ then by univalence, f is the identity function, thus $0_\mathbf{2} = f(1_\mathbf{2}) = 1_\mathbf{2}$, which is a contradiction. □

Homotopy n-type

- Sets, also known as **0-types** have the property that there are no non-trivial paths between inhabitants.
  Similarly, types with no non-trivial paths between paths are known as **1-types**.

Definition (3.1.7)

A type A is a **1-type** if for all $x, y : A$, $p, q : x = y$ and $r, s : p = q$, it holds that $r = s$.

- Similarly, it is possible to define `2-types`, `3-types` for all $n \in \mathbb{N}$.

Homotopy 1-type

Lemma

If A is a set, then A is a **1-type**.

Proof.

If A is a set, then $f : \texttt{isSet}(A)$, and hence for all $x, y : A$, $p, q : x = y$, we have $f(x, y, p, q) : p = q$.

Homotopy 1-type

Lemma

If A is a set, then A is a **1-type**.

Proof.

If A is a set, then $f : \mathtt{isSet}(A)$, and hence for all $x, y : A$, $p, q : x = y$, we have $f(x, y, p, q) : p = q$.
Fix x, y, p and define $g(q) :\equiv f(x, y, p, q)$. For any $r : q = q'$ we have $apd_g(r) : r_*(g(q)) = g(q')$ and hence $g(q) \cdot r = g(q')$ by Lemma 2.11.2.

□

Lemma (2.11.2)

For any $A : \mathcal{U}$, $a, x_1, x_2 : A$ and $p : x_1 = x_2$ we have:

$$transport^{x \mapsto (a=x)}(p, q) = q \cdot p \quad \text{for } q : a = x_1.$$

Homotopy 1-type

Lemma

If A is a set, then A is a **1-type**.

Proof.

If A is a set, then $f : \mathtt{isSet}(A)$, and hence for all $x, y : A$, $p, q : x = y$, we have $f(x, y, p, q) : p = q$.
Fix x, y, p and define $g(q) :\equiv f(x, y, p, q)$. For any $r : q = q'$ we have $apd_g(r) : r_*(g(q)) = g(q')$ and hence $g(q) \cdot r = g(q')$ by Lemma 2.11.2.
In particular, we find for $r, s : q = q'$ that $g(q) \cdot r = g(q') = g(q) \cdot s$ and hence $r = s$. Thus A is a **1-type**. □

Lemma (2.11.2)

For any $A : \mathcal{U}$, $a, x_1, x_2 : A$ and $p : x_1 = x_2$ we have:

$$transport^{x \mapsto (a=x)}(p, q) = q \cdot p \quad \text{for } q : a = x_1.$$

Outline

Proposition

- Previously we used the propositions as types to describe propositions and predicates. For instance, "there exists an x in A such that $P(x)$" is interpreted by $\sum_{x:A} P(x)$.

Initial Representation of Axiom of Choice

If for all $x : X$ there exists an $a : A(X)$ such that $P(x, a)$, then there exists a $g : \prod_{x:X} A(x)$ such that for all $x : X$ we have $P(x, g(x))$.

- In Homotopic Type Theory the above is not an axiom as in classical logic.
- However, statements representing the *law of double negation* and *law of excluded middle* are not always true.

Law of Double Negation 1/5

Theorem (Law of Double Negation)

It is not true that for all $A : \mathcal{U}$, we have that $\neg(\neg A) \to A$.

Proof.

Let $f : \prod_{A:\mathcal{U}}(\neg\neg A \to A)$ be given.

Law of Double Negation 1/5

Theorem (Law of Double Negation)

It is not true that for all $A : \mathcal{U}$*, we have that* $\neg(\neg A) \to A$*.*

Proof.

Let $f : \prod_{A:\mathcal{U}}(\neg\neg A \to A)$ be given.
Let $e : \mathbf{2} \simeq \mathbf{2}$ be given by $e(0_{\mathbf{2}}) = 1_{\mathbf{2}}$ and $e(1_{\mathbf{2}}) = 0_{\mathbf{2}}$.
Then let $p : \mathbf{2} = \mathbf{2}$ be given by $p :\equiv ua(e)$.

Law of Double Negation 1/5

Theorem (Law of Double Negation)

It is not true that for all $A : \mathcal{U}$, we have that $\neg(\neg A) \to A$.

Proof.

Let $f : \prod_{A:\mathcal{U}}(\neg\neg A \to A)$ be given.
Let $e : \mathbf{2} \simeq \mathbf{2}$ be given by $e(0_{\mathbf{2}}) = 1_{\mathbf{2}}$ and $e(1_{\mathbf{2}}) = 0_{\mathbf{2}}$.
Then let $p : \mathbf{2} = \mathbf{2}$ be given by $p :\equiv ua(e)$.
Then $f(\mathbf{2}) : \neg\neg\mathbf{2} \to \mathbf{2}$ and:

$$apd_f(p) : transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2})) = f(\mathbf{2}),$$

Law of Double Negation 1/5

Theorem (Law of Double Negation)

It is not true that for all $A : \mathcal{U}$, we have that $\neg(\neg A) \to A$.

Proof.

Let $f : \prod_{A:\mathcal{U}}(\neg\neg A \to A)$ be given.
Let $e : \mathbf{2} \simeq \mathbf{2}$ be given by $e(0_\mathbf{2}) = 1_\mathbf{2}$ and $e(1_\mathbf{2}) = 0_\mathbf{2}$.
Then let $p : \mathbf{2} = \mathbf{2}$ be given by $p :\equiv ua(e)$.
Then $f(\mathbf{2}) : \neg\neg\mathbf{2} \to \mathbf{2}$ and:

$$apd_f(p) : transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2})) = f(\mathbf{2}),$$

hence for $u : \neg\neg\mathbf{2}$:

$$happly(apd_f(p), u) : transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) = f(\mathbf{2})(u).$$

Law of Double Negation 2/5

Proof.

$$apd_f(p) : transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2})) = f(\mathbf{2}).$$

Law of Double Negation 2/5

Proof.

$$apd_f(p) : transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2})) = f(\mathbf{2}).$$

Then by Lemma 2.9.4 we find that:

$$transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) = transport^{A \mapsto A}(p, f(\mathbf{2})(transport^{A \mapsto \neg\neg A}(p^{-1}, u))).$$

Lemma (2.9.4)

$$transport^{x \mapsto A(x) \to B(x)}(p, f) = (x \mapsto transport^{x \mapsto B(x)}(p, f(transport^{x \mapsto A(x)}(p^{-1}, x))$$

Law of Double Negation 3/5

Proof.

$$\mathit{transport}^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) =$$
$$\mathit{transport}^{A \mapsto A}(p, f(\mathbf{2}(\mathit{transport}^{A \mapsto \neg\neg A}(p^{-1}, u)))).$$

Law of Double Negation 3/5

Proof.

$$transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) =$$
$$transport^{A \mapsto A}(p, f(\mathbf{2}(transport^{A \mapsto \neg\neg A}(p^{-1}, u)))).$$

However any $u, v : \neg\neg\mathbf{2}$ are equal as $u(x) : \mathbf{0}$ and $v(x) : \mathbf{0}$ for all $x : \neg\mathbf{2}$. Hence we find that $u(x) = v(x)$, and by function extensionality $u = v$

Law of Double Negation 3/5

Proof.

$$transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) =$$
$$transport^{A \mapsto A}(p, f(\mathbf{2}(transport^{A \mapsto \neg\neg A}(p^{-1}, u))).$$

However any $u, v : \neg\neg\mathbf{2}$ are equal as $u(x) : \mathbf{0}$ and $v(x) : \mathbf{0}$ for all $x : \neg\mathbf{2}$. Hence we find that $u(x) = v(x)$, and by function extensionality $u = v$ From this and $p : \mathbf{2} = \mathbf{2}$ we find that:

$$transport^{A \mapsto \neg\neg A}(p^{-1}, u) = u.$$

Law of Double Negation 4/5

Proof.

- $transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) =$
  $transport^{A \mapsto A}(p, f(\mathbf{2})(transport^{A \mapsto \neg\neg A}(p^{-1}, u)))$,
- $transport^{A \mapsto \neg\neg A}(p^{-1}, u) = u$,
- $happly(apd_f(p), u) : transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) = f(\mathbf{2})(u)$.

Law of Double Negation 4/5

Proof.

- $transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) =$
  $transport^{A \mapsto A}(p, f(\mathbf{2})(transport^{A \mapsto \neg\neg A}(p^{-1}, u)))$,
- $transport^{A \mapsto \neg\neg A}(p^{-1}, u) = u$,
- $happly(apd_f(p), u) : transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) = f(\mathbf{2})(u)$.

From these we may derive that:

$$transport^{A \mapsto A}(p, f(\mathbf{2})(u)) = f(\mathbf{2})(u),$$

Law of Double Negation 4/5

Proof.

- $transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) =$
  $transport^{A \mapsto A}(p, f(\mathbf{2})(transport^{A \mapsto \neg\neg A}(p^{-1}, u)))$,
- $transport^{A \mapsto \neg\neg A}(p^{-1}, u) = u$,
- $happly(apd_f(p), u) : transport^{A \mapsto (\neg\neg A \to A)}(p, f(\mathbf{2}))(u) = f(\mathbf{2})(u)$.

From these we may derive that:

$$transport^{A \mapsto A}(p, f(\mathbf{2})(u)) = f(\mathbf{2})(u),$$

but taking into account $p \equiv ua(e)$ we find that:

$$e(f(\mathbf{2})(u)) = f(\mathbf{2})(u).$$

Law of Double Negation 5/5

Proof.

We determined that:

$$e(f(\mathbf{2})(u)) = f(\mathbf{2})(u).$$

But by case distinction it can also be proven that:

$$\prod_{x:\mathbf{2}} \neg(e(x) = x).$$

Law of Excluded Middle

Corollary (Law of Excluded Middle)

It is not true that for all $A : \mathcal{U}$, we have $A + \neg A$.

Proof.

Suppose we have $g : \sum_{A:\mathcal{U}}(A + \neg A)$, then we will show that $\prod_{A:\mathcal{U}}(\neg\neg A \to A)$.

Law of Excluded Middle

Corollary (Law of Excluded Middle)

It is not true that for all $A : \mathcal{U}$, we have $A + \neg A$.

Proof.

Suppose we have $g : \sum_{A:\mathcal{U}}(A + \neg A)$, then we will show that $\prod_{A:\mathcal{U}}(\neg\neg A \to A)$.
Let $A : \mathcal{U}$ and $u : \neg\neg A$. Then $g(A) : A + \neg A$.

Law of Excluded Middle

Corollary (Law of Excluded Middle)

It is not true that for all $A : \mathcal{U}$, we have $A + \neg A$.

Proof.

Suppose we have $g : \sum_{A:\mathcal{U}}(A + \neg A)$, then we will show that $\prod_{A:\mathcal{U}}(\neg\neg A \to A)$.
Let $A : \mathcal{U}$ and $u : \neg\neg A$. Then $g(A) : A + \neg A$.
By case distinction, either $g(A) \equiv inl(a)$ and $a : A$ or $g(A) \equiv inr(w)$ and $w : \neg A$.

Law of Excluded Middle

Corollary (Law of Excluded Middle)

It is not true that for all $A : \mathcal{U}$, we have $A + \neg A$.

Proof.

Suppose we have $g : \sum_{A:\mathcal{U}}(A + \neg A)$, then we will show that $\prod_{A:\mathcal{U}}(\neg\neg A \to A)$.
Let $A : \mathcal{U}$ and $u : \neg\neg A$. Then $g(A) : A + \neg A$.
By case distinction, either $g(A) \equiv inl(a)$ and $a : A$ or $g(A) \equiv inr(w)$ and $w : \neg A$.
In the first case, we have $a : A$.
In the second case we have $u(w) : \mathbf{0}$, from which we can obtain an $a : A$. □

Outline

Mere Propositions

- A non-trivial univalance (e) on $\mathbf{2}$ was important to disprove the Law of Double Negation.
- In a classical logic, a proposition only has the information whether it is `true` or `false`.
- Using types however, we also know specifically which inhabitant we have for the type.

Mere Propositions

- A non-trivial univalance (e) on $\mathbf{2}$ was important to disprove the Law of Double Negation.
- In a classical logic, a proposition only has the information whether it is `true` or `false`.
- Using types however, we also know specifically which inhabitant we have for the type.

Definition (3.3.1)

A type A is a **mere proposition** if for all $x, y : A$, we have $x = y$. Given as a type this is equivalent to:

$$\texttt{isProp}(A) :\equiv \prod_{x,y:A} x = y.$$

Example

The type $\mathbf{1}$ is a mere proposition, as by induction $x \equiv \star$ and $y \equiv \star$, thus $\mathit{refl}_\star : x = y$.

Mere Propositions Properties

Lemma (3.3.3)

If P, Q *are* **mere propositions** *such that* $P \to Q$ *and* $Q \to P$*, then* $P \simeq Q$*.*

Mere Propositions Properties

Lemma (3.3.3)

If P, Q are **mere propositions** *such that $P \to Q$ and $Q \to P$, then $P \simeq Q$.*

Proof.

Let P, Q be **mere propositions**, $f : P \to Q$ and $g : Q \to P$, then for all $x : P$ we have $g(f(x)) : P$ so $g(f(x)) = x$.
Similarly for $y : Q$, we have $f(g(y)) = y$. So f, g are quasi-inverses. □

Mere Propositions Properties

Lemma (3.3.3)

If P, Q are **mere propositions** *such that $P \to Q$ and $Q \to P$, then $P \simeq Q$.*

Proof.

Let P, Q be **mere propositions**, $f : P \to Q$ and $g : Q \to P$, then for all $x : P$ we have $g(f(x)) : P$ so $g(f(x)) = x$.
Similarly for $y : Q$, we have $f(g(y)) = y$. So f, g are quasi-inverses. □

Lemma (3.3.2)

If P is a **mere proposition** *and $x_0 : P$ then $P \simeq \mathbf{1}$.*

- The latter lemma corresponds to a space being contractible in homotopy theory.

Mere Proposition Is A Set

Lemma (3.3.4)

Every **mere proposition** *is a* **set**.

Proof.

Similar to proving that **sets** are **1-types**. □

Mere Proposition Is A Set

Lemma (3.3.4)

Every **mere proposition** *is a* **set**.

Proof.

Suppose $f : \mathit{isProp}(A)$, then for all $x, y : A$, $f(x, y) : x = y$. Fix x and define $g(y) :\equiv f(x, y)$. Then for all $y, z : A$ and $p : y = z$ we have $\mathit{apd}_g(p) : p_*(g(y)) = g(z)$ and hence $g(y) \cdot p = g(z)$ and hence $p = g(y)^{-1} \cdot g(z)$. Choosing any $y, z : A$ and $p, q : y = z$ we find that $p = g(y)^{-1} \cdot g(z) = q$. □

More Mere Propositions

Lemma (3.3.5a)

For any type A, $isProp(A)$ is a mere proposition.

Lemma (3.3.5b)

For any type A, $isSet(A)$ is a mere proposition.

More Mere Propositions

Lemma (3.3.5a)

For any type A, $isProp(A)$ is a mere proposition.

Lemma (3.3.5b)

For any type A, $isSet(A)$ is a mere proposition.

Proof.

Suppose $f, g : isSet(A)$. Let $x, y : A$ and $p, q : x = y$ be given, then $f(x, y, p, q) : p = q$ and $g(x, y, p, q) : p = q$.
As A is a set, A is a 1-type, therefore $f(x, y, p, q) = g(x, y, p, q)$.
Hence by function extensionality, $f = g$. □

Outline

Laws that are not laws

General law of excluded middle

$$\mathsf{LEM}_\infty :\equiv \prod_{A:\mathcal{U}} (A + \neg A)$$

Laws that are not laws

General law of excluded middle

$$\mathsf{LEM}_\infty :\equiv \prod_{A:\mathcal{U}} (A + \neg A)$$

- It is not the case that for all $A : \mathcal{U}$ we have $(A + \neg A)$.

Laws that are not laws

General law of excluded middle

$$\mathsf{LEM}_\infty :\equiv \prod_{A:\mathcal{U}} (A + \neg A)$$

- It is not the case that for all $A : \mathcal{U}$ we have $(A + \neg A)$.

General law of double negation

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

Laws that are not laws

General law of excluded middle

$$\mathsf{LEM}_\infty :\equiv \prod_{A:\mathcal{U}} (A + \neg A)$$

- It is not the case that for all $A : \mathcal{U}$ we have $(A + \neg A)$.

General law of double negation

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

- It is also not the case that for all $A : \mathcal{U}$ we have $(\neg\neg A \to A)$.

Actual laws

Mere law of excluded middle

$$\mathsf{LEM} :\equiv \prod_{A:\mathcal{U}} (\mathsf{isProp}(A) \to (A + \neg A))$$

Actual laws

Mere law of excluded middle

$$\mathsf{LEM} :\equiv \prod_{A:\mathcal{U}} (\mathsf{isProp}(A) \to (A + \neg A))$$

Mere law of double negation

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

Actual laws

Mere law of excluded middle

$$\mathsf{LEM} :\equiv \prod_{A:\mathcal{U}} (\mathsf{isProp}(A) \to (A + \neg A))$$

Mere law of double negation

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

- These laws are equivalent to each other.

Necessity of LEM

- LEM is not a consequence of basic type theory, but it can be consistently assumed as an axiom.

Necessity of LEM

- LEM is not a consequence of basic type theory, but it can be consistently assumed as an axiom.
- However, LEM is often used unnecessarily.

Necessity of LEM

- LEM is not a consequence of basic type theory, but it can be consistently assumed as an axiom.
- However, LEM is often used unnecessarily.

Example

The assumption that a set A is nonempty means that it is not the case that A contains no elements. What is often meant is that A contains at least one element. This is an unnecessary use of double negation.

Necessity of LEM

- LEM is not a consequence of basic type theory, but it can be consistently assumed as an axiom.
- However, LEM is often used unnecessarily.

Example

The assumption that a set A is nonempty means that it is not the case that A contains no elements. What is often meant is that A contains at least one element. This is an unnecessary use of double negation.

Example

A proof by contradiction uses the law of double negation. We assume $\neg A$ and derive a contradiction, implying $\neg\neg A$, which in turn implies A by double negation. However, a proof by contradiction can often be rephrased slightly into a direct proof of A.

Advantages of not using LEM

- There are advantages to not using LEM:

Advantages of not using LEM

- There are advantages to not using LEM:
  - A theorem is more powerful when it uses less assumptions, since it applies to more situations.

Advantages of not using LEM

- There are advantages to not using LEM:
  - A theorem is more powerful when it uses less assumptions, since it applies to more situations.
  - One of the virtues of type theory is its computable character. LEM does not have this computable character, since it asserts that certain things exist without giving a way to compute them.

Decidable propositions

Definition

A type A is **decidable** if $A + \neg A$.

Decidable propositions

Definition

A type A is **decidable** if $A + \neg A$.

- LEM is the statement that all mere propositions are decidable.

Decidable propositions

Definition

A type A is **decidable** if $A + \neg A$.

- LEM is the statement that all mere propositions are decidable.

Definition

A type family $B : A \to \mathcal{U}$ is **decidable** if $\prod_{(a:A)} B(a) + \neg B(a)$.

Decidable propositions

Definition

A type A is **decidable** if $A + \neg A$.

- LEM is the statement that all mere propositions are decidable.

Definition

A type family $B : A \to \mathcal{U}$ is **decidable** if $\prod_{(a:A)} B(a) + \neg B(a)$.

- All families of mere propositions are decidable.

Decidable propositions

Definition

A type A is **decidable** if $A + \neg A$.

- LEM is the statement that all mere propositions are decidable.

Definition

A type family $B : A \to \mathcal{U}$ is **decidable** if $\prod_{(a:A)} B(a) + \neg B(a)$.

- All families of mere propositions are decidable.

Definition

A type A has **decidable equality** if $\prod_{(a,b:A)}((a = b) + \neg(a = b))$.

Decidable propositions

Definition

A type A is **decidable** if $A + \neg A$.

- LEM is the statement that all mere propositions are decidable.

Definition

A type family $B : A \to \mathcal{U}$ is **decidable** if $\prod_{(a:A)} B(a) + \neg B(a)$.

- All families of mere propositions are decidable.

Definition

A type A has **decidable equality** if $\prod_{(a,b:A)}((a = b) + \neg(a = b))$.

- LEM implies that all sets have decidable equality.

Outline

Type theory vs set theory

- In type theory there is only one basic notion: types.

Type theory vs set theory

- In type theory there is only one basic notion: types.
- In set theory there are two basic notions: sets and propositions.

Type theory vs set theory

- In type theory there is only one basic notion: types.
- In set theory there are two basic notions: sets and propositions.
- A similar distinction can be made in type theory, by letting the mere propositions be the equivalent of set-theoretic propositions.

Type theory vs set theory

- In type theory there is only one basic notion: types.
- In set theory there are two basic notions: sets and propositions.
- A similar distinction can be made in type theory, by letting the mere propositions be the equivalent of set-theoretic propositions.
- The logical connectives can be represented by restricting the corresponding type formers to mere propositions.

Type theory vs set theory

- In type theory there is only one basic notion: types.
- In set theory there are two basic notions: sets and propositions.
- A similar distinction can be made in type theory, by letting the mere propositions be the equivalent of set-theoretic propositions.
- The logical connectives can be represented by restricting the corresponding type formers to mere propositions.
- This requires knowing that these type formers preserve mere propositions.

Type formers and mere propositions

- If A and B are mere propositions, then so is $A \times B$.

Type formers and mere propositions

- If A and B are mere propositions, then so is $A \times B$.

Proof.

Given any $p, q : A \times B$, we have $\mathrm{pr}_1(p) =_A \mathrm{pr}_1(q)$ and $\mathrm{pr}_2(p) =_B \mathrm{pr}_2(q)$, since A and B are mere propositions. So $p = q$. □

Type formers and mere propositions

- If A and B are mere propositions, then so is $A \times B$.

Proof.

Given any $p, q : A \times B$, we have $\mathrm{pr}_1(p) =_A \mathrm{pr}_1(q)$ and $\mathrm{pr}_2(p) =_B \mathrm{pr}_2(q)$, since A and B are mere propositions. So $p = q$. □

- If A is any type and $B : A \to \mathcal{U}$ is such that for all $x : A$, the type $B(x)$ is a mere proposition, then $\prod_{(x:A)} B(x)$ is a mere proposition.

Type formers and mere propositions

- If A and B are mere propositions, then so is $A \times B$.

Proof.

Given any $p, q : A \times B$, we have $\mathrm{pr}_1(p) =_A \mathrm{pr}_1(q)$ and $\mathrm{pr}_2(p) =_B \mathrm{pr}_2(q)$, since A and B are mere propositions. So $p = q$. □

- If A is any type and $B : A \to \mathcal{U}$ is such that for all $x : A$, the type $B(x)$ is a mere proposition, then $\prod_{(x:A)} B(x)$ is a mere proposition.

Proof.

Given $f, g : \prod_{(x:A)} B(x)$, for any $x : A$, we have $f(x) = g(x)$, since $B(x)$ is a mere proposition. By function extensionality, we have $f = g$. □

Type formers and mere propositions

- If A and B are mere propositions, then so is $A \times B$.

Proof.

Given any $p, q : A \times B$, we have $\mathrm{pr}_1(p) =_A \mathrm{pr}_1(q)$ and $\mathrm{pr}_2(p) =_B \mathrm{pr}_2(q)$, since A and B are mere propositions. So $p = q$. □

- If A is any type and $B : A \to \mathcal{U}$ is such that for all $x : A$, the type $B(x)$ is a mere proposition, then $\prod_{(x:A)} B(x)$ is a mere proposition.

Proof.

Given $f, g : \prod_{(x:A)} B(x)$, for any $x : A$, we have $f(x) = g(x)$, since $B(x)$ is a mere proposition. By function extensionality, we have $f = g$. □

- In particular, if B is a mere proposition, then so is $A \to B$.

Type formers and mere propositions

- If A and B are mere propositions, then so is $A \times B$.

Proof.

Given any $p, q : A \times B$, we have $\mathrm{pr}_1(p) =_A \mathrm{pr}_1(q)$ and $\mathrm{pr}_2(p) =_B \mathrm{pr}_2(q)$, since A and B are mere propositions. So $p = q$. □

- If A is any type and $B : A \to \mathcal{U}$ is such that for all $x : A$, the type $B(x)$ is a mere proposition, then $\prod_{(x:A)} B(x)$ is a mere proposition.

Proof.

Given $f, g : \prod_{(x:A)} B(x)$, for any $x : A$, we have $f(x) = g(x)$, since $B(x)$ is a mere proposition. By function extensionality, we have $f = g$. □

- In particular, if B is a mere proposition, then so is $A \to B$.
- Since $\mathbf{0}$ is a mere proposition, so is $\neg A \equiv (A \to \mathbf{0})$.

Type formers and mere propositions

- If A and B are mere propositions, then $A + B$ is not necessarily a mere proposition.

Type formers and mere propositions

- If A and B are mere propositions, then $A + B$ is not necessarily a mere proposition.

Example

$\mathbf{1}$ is a mere proposition, but $\mathbf{1} + \mathbf{1} = \mathbf{2}$ is not a mere proposition.

Type formers and mere propositions

- If A and B are mere propositions, then $A + B$ is not necessarily a mere proposition.

Example

$\mathbf{1}$ is a mere proposition, but $\mathbf{1} + \mathbf{1} = \mathbf{2}$ is not a mere proposition.

- An inhabitant of $A + B$ contains the additional information of which disjunct is true.

Type formers and mere propositions

- If A and B are mere propositions, then $A + B$ is not necessarily a mere proposition.

Example

$\mathbf{1}$ is a mere proposition, but $\mathbf{1} + \mathbf{1} = \mathbf{2}$ is not a mere proposition.

- An inhabitant of $A + B$ contains the additional information of which disjunct is true.
- To get the 'or' of classical logic, we need to truncate this type into a mere proposition by forgetting the additional information.

Type formers and mere propositions

- If A and B are mere propositions, then $A + B$ is not necessarily a mere proposition.

Example

$\mathbf{1}$ is a mere proposition, but $\mathbf{1} + \mathbf{1} = \mathbf{2}$ is not a mere proposition.

- An inhabitant of $A + B$ contains the additional information of which disjunct is true.
- To get the 'or' of classical logic, we need to truncate this type into a mere proposition by forgetting the additional information.
- It is similar for the Σ-type, representing 'there exists', because an inhabitant of a Σ-type remembers the witness.

Propositional truncation

Definition

The **propositional truncation** is an additional type former which truncates a type down to a mere proposition.

Propositional truncation

Definition

The **propositional truncation** is an additional type former which truncates a type down to a mere proposition.

Propositional truncation

For any type A, there is a type $||A||$. It has two constructors:

- For any $a : A$, we have $|a| : ||A||$.
- For any $x, y : ||A||$, we have $x = y$.

Propositional truncation

Definition

The **propositional truncation** is an additional type former which truncates a type down to a mere proposition.

Propositional truncation

For any type A, there is a type $||A||$. It has two constructors:

- For any $a : A$, we have $|a| : ||A||$.
- For any $x, y : ||A||$, we have $x = y$.

- The first constructor ensures that if A is inhabited, then so is $||A||$.

Propositional truncation

Definition

The **propositional truncation** is an additional type former which truncates a type down to a mere proposition.

Propositional truncation

For any type A, there is a type $||A||$. It has two constructors:

- For any $a : A$, we have $|a| : ||A||$.
- For any $x, y : ||A||$, we have $x = y$.

- The first constructor ensures that if A is inhabited, then so is $||A||$.
- The second constructor ensures that $||A||$ is a mere proposition.

Recursion principle

Recursion principle of truncation

If B is a mere proposition and we have $f : A \to B$, then there is an induced $g : ||A|| \to B$ such that $g(|a|) \equiv f(a)$ for all $a : A$.

- In other words, any mere proposition that follows from the inhabitedness of A, already follows from $||A||$.

Mere propositional disjunction

- $||A + B||$ is a mere propositional version of $A + B$ that does not remember which of the disjuncts is true.

Mere propositional disjunction

- $||A + B||$ is a mere propositional version of $A + B$ that does not remember which of the disjuncts is true.
- When attempting to prove a mere proposition, it is still possible to do case analysis on $||A + B||$, due to the recursion principle.

Mere propositional disjunction

- $||A + B||$ is a mere propositional version of $A + B$ that does not remember which of the disjuncts is true.
- When attempting to prove a mere proposition, it is still possible to do case analysis on $||A + B||$, due to the recursion principle.

Example

- Suppose we have an assumption $||A + B||$ and we are trying to prove a mere proposition Q.

Mere propositional disjunction

- $||A + B||$ is a mere propositional version of $A + B$ that does not remember which of the disjuncts is true.
- When attempting to prove a mere proposition, it is still possible to do case analysis on $||A + B||$, due to the recursion principle.

Example

- Suppose we have an assumption $||A + B||$ and we are trying to prove a mere proposition Q.
- This means that we are trying to find an element of $||A + B|| \to Q$.

Mere propositional disjunction

- $||A + B||$ is a mere propositional version of $A + B$ that does not remember which of the disjuncts is true.
- When attempting to prove a mere proposition, it is still possible to do case analysis on $||A + B||$, due to the recursion principle.

Example

- Suppose we have an assumption $||A + B||$ and we are trying to prove a mere proposition Q.
- This means that we are trying to find an element of $||A + B|| \to Q$.
- Since Q is a mere proposition, it suffices to construct a function $A + B \to Q$, because of the recursion principle.

Mere propositional disjunction

- $||A + B||$ is a mere propositional version of $A + B$ that does not remember which of the disjuncts is true.
- When attempting to prove a mere proposition, it is still possible to do case analysis on $||A + B||$, due to the recursion principle.

Example

- Suppose we have an assumption $||A + B||$ and we are trying to prove a mere proposition Q.
- This means that we are trying to find an element of $||A + B|| \to Q$.
- Since Q is a mere proposition, it suffices to construct a function $A + B \to Q$, because of the recursion principle.
- Now it is possible to use case analysis on $A + B$.

Mere propositional Σ-type

- For a type family $P : A \to \mathcal{U}$, $\| \sum_{(x:A)} P(x) \|$ is a mere propositional version of 'there exists an $x : A$ such that $P(x)$ holds'.

Mere propositional Σ-type

- For a type family $P : A \to \mathcal{U}$, $\| \sum_{(x:A)} P(x) \|$ is a mere propositional version of 'there exists an $x : A$ such that $P(x)$ holds'.
- When trying to prove a mere proposition, if we have an assumption $\| \sum_{(x:A)} P(x) \|$, then by the recursion principle we may introduce the new assumptions $x : A$ and $y : P(x)$.

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

$$\bot :\equiv \mathbf{0}$$

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

$$\bot :\equiv \mathbf{0}$$

$$P \wedge Q :\equiv P \times Q$$

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

$$\bot :\equiv \mathbf{0}$$

$$P \wedge Q :\equiv P \times Q$$

$$P \Rightarrow Q :\equiv P \to Q$$

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

$$\bot :\equiv \mathbf{0}$$

$$P \wedge Q :\equiv P \times Q$$

$$P \Rightarrow Q :\equiv P \to Q$$

$$P \Leftrightarrow Q :\equiv P = Q$$

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

$$\bot :\equiv \mathbf{0}$$

$$P \wedge Q :\equiv P \times Q$$

$$P \Rightarrow Q :\equiv P \to Q$$

$$P \Leftrightarrow Q :\equiv P = Q$$

$$\neg P :\equiv P \to \mathbf{0}$$

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

$$\bot :\equiv \mathbf{0}$$

$$P \wedge Q :\equiv P \times Q$$

$$P \Rightarrow Q :\equiv P \to Q$$

$$P \Leftrightarrow Q :\equiv P = Q$$

$$\neg P :\equiv P \to \mathbf{0}$$

$$P \vee Q :\equiv \|P + Q\|$$

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

$$\bot :\equiv \mathbf{0}$$

$$P \wedge Q :\equiv P \times Q$$

$$P \Rightarrow Q :\equiv P \to Q$$

$$P \Leftrightarrow Q :\equiv P = Q$$

$$\neg P :\equiv P \to \mathbf{0}$$

$$P \vee Q :\equiv \|P + Q\|$$

$$\forall (x : A).P(x) :\equiv \prod_{x:A} P(x)$$

Logical notation

Definition

Traditional logical notation is defined using truncation as follows, where P and Q denote mere propositions (or families of mere propositions):

$$\top :\equiv \mathbf{1}$$

$$\bot :\equiv \mathbf{0}$$

$$P \wedge Q :\equiv P \times Q$$

$$P \Rightarrow Q :\equiv P \to Q$$

$$P \Leftrightarrow Q :\equiv P = Q$$

$$\neg P :\equiv P \to \mathbf{0}$$

$$P \vee Q :\equiv \|P + Q\|$$

$$\forall (x : A).P(x) :\equiv \prod_{x:A} P(x)$$

$$\exists (x : A).P(x) :\equiv \| \sum_{x:A} P(x) \|$$

Set notation

- We can now use traditional notation for the instersections, unions and complements of sets.

Set notation

- We can now use traditional notation for the instersections, unions and complements of sets.

Set notation

$$\{x : A | P(x)\} \cap \{x : A | Q(x)\} :\equiv \{x : A | P(x) \wedge Q(x)\}$$

Set notation

- We can now use traditional notation for the instersections, unions and complements of sets.

Set notation

$$\{x : A | P(x)\} \cap \{x : A | Q(x)\} :\equiv \{x : A | P(x) \wedge Q(x)\}$$
$$\{x : A | P(x)\} \cup \{x : A | Q(x)\} :\equiv \{x : A | P(x) \vee Q(x)\}$$

Set notation

- We can now use traditional notation for the instersections, unions and complements of sets.

Set notation

$$\{x : A | P(x)\} \cap \{x : A | Q(x)\} :\equiv \{x : A | P(x) \wedge Q(x)\}$$
$$\{x : A | P(x)\} \cup \{x : A | Q(x)\} :\equiv \{x : A | P(x) \vee Q(x)\}$$
$$A \setminus \{x : A | P(x)\} :\equiv \{x : A | \neg P(x)\}$$

Set notation

- We can now use traditional notation for the instersections, unions and complements of sets.

Set notation

$$\{x : A | P(x)\} \cap \{x : A | Q(x)\} :\equiv \{x : A | P(x) \wedge Q(x)\}$$
$$\{x : A | P(x)\} \cup \{x : A | Q(x)\} :\equiv \{x : A | P(x) \vee Q(x)\}$$
$$A \setminus \{x : A | P(x)\} :\equiv \{x : A | \neg P(x)\}$$

- Note that in the absence of LEM, it is possible that $B \cup (A \setminus B) \neq A$, with B a subset of A.

The end

Are there questions?