Type theory in Lean - 7

Riccardo Brasca

Université Paris Cité

November 25th 2023

Mathlib's axioms

Mathlib is Lean's standard mathematical library.

Mathlib's axioms

Mathlib is Lean's standard mathematical library.

It consists of more than 1.5 millions lines of code of mathematics

Mathlib's axioms

Mathlib is Lean's standard mathematical library.

It consists of more than 1.5 millions lines of code of mathematics, from undergraduate to research level.

Mathlib's axioms

Mathlib is Lean's standard mathematical library.

It consists of more than 1.5 millions lines of code of mathematics, from undergraduate to research level.

It adds three axioms to Lean's type theory:

Mathlib's axioms

Mathlib is Lean's standard mathematical library.

It consists of more than 1.5 millions lines of code of mathematics, from undergraduate to research level.

It adds three axioms to Lean's type theory:

- Propositional extensionality.
- Quotient types.
- The axiom of choice.

Propositional extensionality

```
axiom propext {a b : Prop} : (a ↔ b) → a = b
```

Propositional extensionality

```
axiom propext {a b : Prop} : (a ↔ b) → a = b
```

Propositional extensionality says that if two propositions are equivalent then they are equal.

Propositional extensionality

```
axiom propext {a b : Prop} : (a ↔ b) → a = b
```

Propositional extensionality says that if two propositions are equivalent then they are equal.

In particular if P is provable, meaning we can construct $(p : P)$, then $P = \mathrm{True}$.

Propositional extensionality

```
axiom propext {a b : Prop} : (a ↔ b) → a = b
```

Propositional extensionality says that if two propositions are equivalent then they are equal.

In particular if P is provable, meaning we can construct $(p : P)$, then $P = \mathrm{True}$. Indeed, any proposition implies True (since we can prove True for free), and if we have $(p : P)$ then $\mathrm{True} \to P$.

Propositional extensionality

```
axiom propext {a b : Prop} : (a ↔ b) → a = b
```

Propositional extensionality says that if two propositions are equivalent then they are equal.

In particular if P is provable, meaning we can construct $(p : P)$, then $P = \mathrm{True}$. Indeed, any proposition implies True (since we can prove True for free), and if we have $(p : P)$ then $\mathrm{True} \to P$. Similarly, if $\neg P$ holds, then $P = \mathrm{False}$.

Thinking about proposition as “sets” that are empty when false and singletons when provable, then propositional extensionality says that if we have functions

$$P \to Q \text{ and } Q \to P$$

then $P = Q$.

Thinking about proposition as "sets" that are empty when false and singletons when provable, then propositional extensionality says that if we have functions

$$P \to Q \text{ and } Q \to P$$

then $P = Q$.

This is reasonable since the existence of the two functions (that is the existence of the two implications) forces P and Q to be both empty or both singletons.

Quotient types

In set theory, if $\sim$ is an equivalence relation on a set X, one can explicitly build the quotient set $X/\sim$ using equivalence classes:

$$X/\sim \subseteq \mathcal{P}(X),$$

where $\mathcal{P}(X)$ is the power set of X.

Quotient types

In set theory, if $\sim$ is an equivalence relation on a set X, one can explicitly build the quotient set $X/\sim$ using equivalence classes:

$$X/\sim \subseteq \mathcal{P}(X),$$

where $\mathcal{P}(X)$ is the power set of X. The existence of the sets $\mathcal{P}(X)$ and $X/\sim$ are immediate consequences of the axioms of set theory.

Quotient types

In set theory, if $\sim$ is an equivalence relation on a set X, one can explicitly build the quotient set $X/\sim$ using equivalence classes:

$$X/\sim \subseteq \mathcal{P}(X),$$

where $\mathcal{P}(X)$ is the power set of X. The existence of the sets $\mathcal{P}(X)$ and $X/\sim$ are immediate consequences of the axioms of set theory.

Then one defines the canonical map $X \to X/\sim$ and one can prove the universal property.

Quotient types

In set theory, if $\sim$ is an equivalence relation on a set X, one can explicitly build the quotient set $X/\sim$ using equivalence classes:

$$X/\sim \subseteq \mathcal{P}(X),$$

where $\mathcal{P}(X)$ is the power set of X. The existence of the sets $\mathcal{P}(X)$ and $X/\sim$ are immediate consequences of the axioms of set theory.

Then one defines the canonical map $X \to X/\sim$ and one can prove the universal property.

How to build $X/\sim$ in type theory?

We only have two ways of building types:

- Dependent functions.
- Inductive types.

Neither of these two constructions allows to build $X/\sim$.

We only have two ways of building types:

- Dependent functions.
- Inductive types.

Neither of these two constructions allows to build $X/\sim$.

Lean's type theory adds a new axiom (a function in this case) that allows to build the quotient type.

```
axiom Quot :
  {X : Sort u} → (X → X → Prop) → Sort u
```

In particular, Quot allows to build a new type $\mathrm{Quot}\ R$ given any relation $R \colon X \to X \to X$ (we don't even need to assume that R is an equivalence relation).

In particular, Quot allows to build a new type $\mathrm{Quot}\ R$ given any relation $R \colon X \to X \to X$ (we don't even need to assume that R is an equivalence relation).

We also need the canonical map $X \to \mathrm{Quot}\ R$:

```
axiom Quot.mk :
  {X : Sort u} → (R : X → X → Prop) → X →
    Quot R
```

In particular, Quot allows to build a new type $\mathrm{Quot}\ R$ given any relation $R\colon X \to X \to X$ (we don't even need to assume that R is an equivalence relation).

We also need the canonical map $X \to \mathrm{Quot}\ R$:

```
axiom Quot.mk :
  {X : Sort u} → (R : X → X → Prop) → X →
    Quot R
```

At this point this is just any function, we don't know anything about it.

We also need to know that $\mathrm{Quot.mk}$ is surjective.

We also need to know that $\mathrm{Quot.mk}$ is surjective. In analogy with the induction principle for an inductive type, this is stated by saying that to prove a proposition about all $(q : \mathrm{Quot}\ R)$, it is enough to prove it for those terms of the form $\mathrm{Quot.mk}\ R\ a$.

We also need to know that $\mathrm{Quot.mk}$ is surjective. In analogy with the induction principle for an inductive type, this is stated by saying that to prove a proposition about all $(q : \mathrm{Quot}\ R)$, it is enough to prove it for those terms of the form $\mathrm{Quot.mk}\ R\ a$.

```
axiom Quot.ind :
    ∀ {X : Sort u} {R : X → X → Prop}
    {P : Quot r → Prop},
    (∀ a, P (Quot.mk R a)) →
    ∀ q, P q
```

We also need to know that $\mathrm{Quot.mk}$ is surjective. In analogy with the induction principle for an inductive type, this is stated by saying that to prove a proposition about all $(q : \mathrm{Quot}\ R)$, it is enough to prove it for those terms of the form $\mathrm{Quot.mk}\ R\ a$.

```
axiom Quot.ind :
    ∀ {X : Sort u} {R : X → X → Prop}
    {P : Quot r → Prop},
    (∀ a, P (Quot.mk R a)) →
    ∀ q, P q
```

It implies that $\mathrm{Quot.mk}$ is surjective in the usual sense.

We also need to lift functions that are constant along the equivalence classes.

```
axiom Quot.lift :
  {X : Sort u} → {R : X → X → Prop} →
  {Y : Sort u} → (f : X → Y) →
  (∀ a b, R a b → f a = f b) →
  Quot R → Y
```

We also need to lift functions that are constant along the equivalence classes.

```
axiom Quot.lift :
  {X : Sort u} → {R : X → X → Prop} →
  {Y : Sort u} → (f : X → Y) →
  (∀ a b, R a b → f a = f b) →
  Quot R → Y
```

Note that, as for inductive types, we can now build terms of type $\mathrm{Quot}\ R$ (using $\mathrm{Quot.mk}$) and define functions (including proving propositions) out of $\mathrm{Quot}\ R$.

We want to relate the function given by $\mathrm{Quot.lift}$ with $\mathrm{Quot.mk}$.

We want to relate the function given by $\mathrm{Quot.lift}$ with $\mathrm{Quot.mk}$. If $f : X \to Y$ is such that $(H : \forall a\ b,\ R\ a\ b \to f\ a = f\ b)$, then we want $\mathrm{Quot.lift}\ H(\mathrm{Quot.mk}\ R\ x) = f\ x$ for all $(x : X)$.

We want to relate the function given by $\mathrm{Quot.lift}$ with $\mathrm{Quot.mk}$. If $f : X \to Y$ is such that $(H : \forall a\ b,\ R\ a\ b \to f\ a = f\ b)$, then we want $\mathrm{Quot.lift}\ H(\mathrm{Quot.mk}\ R\ x) = f\ x$ for all $(x : X)$. We add this computation rule as a *definitional equality*

$$\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) \equiv f\ x$$

We want to relate the function given by $\mathrm{Quot.lift}$ with $\mathrm{Quot.mk}$. If $f : X \to Y$ is such that $(H : \forall a\ b,\ R\ a\ b \to f\ a = f\ b)$, then we want $\mathrm{Quot.lift}\ H(\mathrm{Quot.mk}\ R\ x) = f\ x$ for all $(x : X)$. We add this computation rule as a *definitional equality*

$$\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) \equiv f\ x$$

Remark

These functions are left undefined.

We want to relate the function given by $\mathrm{Quot.lift}$ with $\mathrm{Quot.mk}$. If $f : X \to Y$ is such that $(H : \forall a\ b,\ R\ a\ b \to f\ a = f\ b)$, then we want $\mathrm{Quot.lift}\ H(\mathrm{Quot.mk}\ R\ x) = f\ x$ for all $(x : X)$. We add this computation rule as a *definitional equality*

$$\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) \equiv f\ x$$

Remark

These functions are left undefined. In practice we are assuming their existence, but we are not assuming any special property.

We want to relate the function given by $\mathrm{Quot.lift}$ with $\mathrm{Quot.mk}$. If $f : X \to Y$ is such that $(H : \forall a\ b,\ R\ a\ b \to f\ a = f\ b)$, then we want $\mathrm{Quot.lift}\ H(\mathrm{Quot.mk}\ R\ x) = f\ x$ for all $(x : X)$. We add this computation rule as a *definitional equality*

$$\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) \equiv f\ x$$

Remark

These functions are left undefined. In practice we are assuming their existence, but we are not assuming any special property. This existence is not a very strong assumption, for example $\mathrm{Quot}\ R$ *could be* X, $\mathrm{Quot.mk}$ *the identity and* $\mathrm{Quot.lift}\ f\ H = f$. *In this case* $\mathrm{Quot.ind}$ *and the computation rule above trivially hold.*

The axioms above are part of Lean's type theory, but what makes them really a construction of quotient types is the following axioms, added in mathlib.

The axioms above are part of Lean's type theory, but what makes them really a construction of quotient types is the following axioms, added in mathlib.

```
axiom Quot.sound :
  ∀ {X : Type u} {R : X → X → Prop} {a b : X},
    R a b → Quot.mk R a = Quot.mk R b
```

It does not follow from the previous axioms (try to prove it!).

The axioms above are part of Lean's type theory, but what makes them really a construction of quotient types is the following axioms, added in mathlib.

```
axiom Quot.sound :
  ∀ {X : Type u} {R : X → X → Prop} {a b : X},
    R a b → Quot.mk R a = Quot.mk R b
```

It does not follow from the previous axioms (try to prove it!).

Remark

- *If* $\mathrm{Quot}\ R = X$ *then* $\mathrm{Quot.sound}$ *does not hold. It is a genuine new axiom.*
- *Note that we didn't assume R to be an equivalence relation, but this is the situation where the axioms are most useful.*

Functional extensionality

Using quotient types we can now *prove* the following.

Theorem (Functional extensionality)

Let $(A : \mathrm{Sort}\ u)$ *and* $(B : \mathrm{Sort}\ v)$. *Given two functions* $(f\ g : A \to B)$ *such that*

$$\forall (a : A),\ f\ a = g\ a$$

we have $f = g$.

Functional extensionality

Using quotient types we can now *prove* the following.

Theorem (Functional extensionality)

Let $(A : \mathrm{Sort}\ u)$ *and* $(B : \mathrm{Sort}\ v)$*. Given two functions* $(f\ g : A \to B)$ *such that*

$$\forall (a : A),\ f\ a = g\ a$$

we have $f = g$*.*

In particular, any theorem that uses functional extensionality will depend on $\mathrm{Quot.sound}$.

To prove it, let's define a relation on $A \to B$ via

$$R\ f\ g \iff \forall (a : A),\ f\ a = g\ a$$

To prove it, let's define a relation on $A \to B$ via

$$R\ f\ g \iff \forall (a : A),\ f\ a = g\ a$$

It is very easy to show that it is an equivalence relation, but we will not need it.

To prove it, let's define a relation on $A \to B$ via

$$R\ f\ g \iff \forall (a : A),\ f\ a = g\ a$$

It is very easy to show that it is an equivalence relation, but we will not need it.

We now call X the quotient type, so

$$X = \mathrm{Quot}\ R$$

is the type of functions $A \to B$ up to being pointwise equal.

If $(a : A)$ and $R\ f\ g$, then (by the very definition of R!) we have $f\ a = g\ a$, so the evaluation at a

$$(A \to B) \to B$$
$$f \mapsto f\ a$$

lifts (mathematically we usually say "descends") to a function

$$\mathrm{ev}\ a \colon X \to B$$

using $\mathrm{Quot.lift}$.

If $(a : A)$ and $R\ f\ g$, then (by the very definition of R!) we have $f\ a = g\ a$, so the evaluation at a

$$(A \to B) \to B$$
$$f \mapsto f\ a$$

lifts (mathematically we usually say "descends") to a function

$$\mathrm{ev}\ a \colon X \to B$$

using $\mathrm{Quot.lift}$.

This can be done for all $(a : A)$, obtaining via lambda abstraction a function $A \to B$.

Putting everything together, we obtain a function

$$F\colon X \to (A \to B)$$

Putting everything together, we obtain a function

$$F \colon X \to (A \to B)$$

Here we are simply saying that, since the value at any $(a : A)$ is constant on each equivalence class (by definition!) we can evaluate elements of X, and lambda abstraction allows to build a function $A \to B$ given $(x : X)$.

Putting everything together, we obtain a function

$$F \colon X \to (A \to B)$$

Here we are simply saying that, since the value at any $(a : A)$ is constant on each equivalence class (by definition!) we can evaluate elements of X, and lambda abstraction allows to build a function $A \to B$ given $(x : X)$.

The computation rule says that

$$F\ (\mathrm{Quot.mk}\ R\ f) \equiv \mathrm{fun}\ a \mapsto f\ a \equiv f$$

for all $(f : A \to B)$.

We can now finish the proof. Let $(f\ g : A \to B)$ be such that $\forall (a : A),\ f\ a = g\ a$, i.e. such that $R\ f\ g$.

We can now finish the proof. Let $(f\ g : A \to B)$ be such that $\forall (a : A),\ f\ a = g\ a$, i.e. such that $R\ f\ g$.

Using $\mathrm{Quot.sound}$ we have then

$$\mathrm{Quot.mk}\ R\ f = \mathrm{Quot.mk}\ R\ g$$

We can now finish the proof. Let $(f\ g : A \to B)$ be such that $\forall (a : A),\ f\ a = g\ a$, i.e. such that $R\ f\ g$.

Using $\mathrm{Quot.sound}$ we have then

$$\mathrm{Quot.mk}\ R\ f = \mathrm{Quot.mk}\ R\ g$$

Using the substitution principle and symmetry of definitional equality we now have

$$f \equiv F\ (\mathrm{Quot.mk}\ R\ f) = F\ (\mathrm{Quot.mk}\ R\ g) \equiv g$$

We can now finish the proof. Let $(f\ g : A \to B)$ be such that $\forall (a : A),\ f\ a = g\ a$, i.e. such that $R\ f\ g$.

Using $\mathrm{Quot.sound}$ we have then

$$\mathrm{Quot.mk}\ R\ f = \mathrm{Quot.mk}\ R\ g$$

Using the substitution principle and symmetry of definitional equality we now have

$$f \equiv F\ (\mathrm{Quot.mk}\ R\ f) = F\ (\mathrm{Quot.mk}\ R\ g) \equiv g$$

In particular, since definitional equality implies propositional equality and the latter is transitive, we have $f = g$ as wanted.

There is a (very) subtle point here. The equality $F\ (\mathrm{Quot.mk}\ R\ f) = f$ holds because definitional equality implies propositional equality and it is a consequence of the computation rule

$$\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) \equiv f\ x$$

There is a (very) subtle point here. The equality $F\ (\mathrm{Quot.mk}\ R\ f) = f$ holds because definitional equality implies propositional equality and it is a consequence of the computation rule

$$\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) \equiv f\ x \tag{1}$$

What happens if we simply add $\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) = f\ x$ as an axiom instead of a definitional equality?

There is a (very) subtle point here. The equality $F\ (\mathrm{Quot.mk}\ R\ f) = f$ holds because definitional equality implies propositional equality and it is a consequence of the computation rule

$$\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) \equiv f\ x \tag{1}$$

What happens if we simply add $\mathrm{Quot.lift}\ H\ (\mathrm{Quot.mk}\ R\ x) = f\ x$ as an axiom instead of a definitional equality?
When proving $F\ (\mathrm{Quot.mk}\ R\ f) = f$ using (1), unravelling all the definitions we end up with

$$\mathrm{fun}\ a \mapsto \mathrm{Quot.lift}\ (\mathrm{fun}\ f \mapsto f\ a)\ \_\ (\mathrm{Quot.mk}\ R\ f) = f$$

Here the underscore $\_$ is just the proof that the function $\mathrm{fun}\ f \mapsto f\ a$ is constant on any equivalence class.

Since f is (definitionally) equal to $\mathrm{fun}\ a \mapsto f\ a$ we can prove

$$\mathrm{fun}\ a \mapsto \mathrm{Quot.lift}\ (\mathrm{fun}\ f \mapsto f\ a)\ \_\ (\mathrm{Quot.mk}\ R\ f) = \mathrm{fun}\ a \mapsto f\ a$$

Since f is (definitionally) equal to $\mathrm{fun}\ a \mapsto f\ a$ we can prove

$$\mathrm{fun}\ a \mapsto \mathrm{Quot.lift}\ (\mathrm{fun}\ f \mapsto f\ a)\ \_\ (\mathrm{Quot.mk}\ R\ f) = \mathrm{fun}\ a \mapsto f\ a \tag{2}$$

Now, if $(a : A)$ is given, the computation rule says exactly that

$$\mathrm{Quot.lift}\ (\mathrm{fun}\ f \mapsto f\ a)\ \_\ (\mathrm{Quot.mk}\ R\ f) = f\ a$$

Since f is (definitionally) equal to $\text{fun } a \mapsto f\ a$ we can prove

$$\text{fun } a \mapsto \text{Quot.lift } (\text{fun } f \mapsto f\ a)\ \_\ (\text{Quot.mk } R\ f) = \text{fun } a \mapsto f\ a \tag{2}$$

Now, if $(a : A)$ is given, the computation rule says exactly that

$$\text{Quot.lift } (\text{fun } f \mapsto f\ a)\ \_\ (\text{Quot.mk } R\ f) = f\ a \tag{3}$$

But this is not enough to prove (2), since (3) says exactly that the LHS and the RHS of (2) have *the same value*, and we are precisely proving that this implies that the two functions are equal.

Since f is (definitionally) equal to $\mathrm{fun}\ a \mapsto f\ a$ we can prove

$$\mathrm{fun}\ a \mapsto \mathrm{Quot.lift}\ (\mathrm{fun}\ f \mapsto f\ a)\ \_\ (\mathrm{Quot.mk}\ R\ f) = \mathrm{fun}\ a \mapsto f\ a \tag{2}$$

Now, if $(a : A)$ is given, the computation rule says exactly that

$$\mathrm{Quot.lift}\ (\mathrm{fun}\ f \mapsto f\ a)\ \_\ (\mathrm{Quot.mk}\ R\ f) = f\ a \tag{3}$$

But this is not enough to prove (2), since (3) says exactly that the LHS and the RHS of (2) have *the same value*, and we are precisely proving that this implies that the two functions are equal.

Since the computation rule is a definitional equality, this problem does not appear and we can finish the proof.

Remark

An important observation is the following. Suppose that $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a = g\ a$$

Remark

An important observation is the following. Suppose that $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a = g\ a$$

Proving that $f = g$ *is impossible without a new axiom. But suppose now that* $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a \equiv g\ a$$

is a definitional equality.

Remark

An important observation is the following. Suppose that $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a = g\ a$$

Proving that $f = g$ *is impossible without a new axiom. But suppose now that* $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a \equiv g\ a$$

is a definitional equality.
Then we have

$$f \equiv \mathrm{fun}\ a \mapsto f\ a \equiv \mathrm{fun}\ a \mapsto g\ a \equiv g$$

Remark

An important observation is the following. Suppose that $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a = g\ a$$

Proving that $f = g$ *is impossible without a new axiom. But suppose now that* $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a \equiv g\ a$$

is a definitional equality.
Then we have

$$f \equiv \text{fun } a \mapsto f\ a \equiv \text{fun } a \mapsto g\ a \equiv g$$

so $f \equiv g$*.*

Remark

An important observation is the following. Suppose that $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a = g\ a$$

Proving that $f = g$ *is impossible without a new axiom. But suppose now that* $(f\ g : A \to B)$ *are such that*

$$\forall (a : A),\ f\ a \equiv g\ a$$

is a definitional equality.
Then we have

$$f \equiv \mathrm{fun}\ a \mapsto f\ a \equiv \mathrm{fun}\ a \mapsto g\ a \equiv g$$

so $f \equiv g$*. The point is that the* $\equiv$ *in the middle holds since* $f\ a \equiv g\ a$ *and so we can replace the former by the latter for free.*

Sets

Remember that given $(A : \mathrm{Type}\ u)$, we have defined $\mathrm{Set}\ A$ as

$$A \to \mathrm{Prop}.$$

Sets

Remember that given $(A : \mathrm{Type}\ u)$, we have defined $\mathrm{Set}\ A$ as

$$A \to \mathrm{Prop}.$$

Moreover, if $(a : A)$ and $(S : \mathrm{Set}\ A)$, then $a \in S$ means

$$S\ a$$

Sets

Remember that given $(A : \mathrm{Type}\ u)$, we have defined $\mathrm{Set}\ A$ as

$$A \to \mathrm{Prop}.$$

Moreover, if $(a : A)$ and $(S : \mathrm{Set}\ A)$, then $a \in S$ means

$$S\ a$$

We can now prove *extensionality for sets*. If $(S\ T : \mathrm{Set}\ A)$, then

$$S = T \iff \forall (a : A),\ a \in S \iff a \in T.$$

Sets

Remember that given $(A : \mathrm{Type}\ u)$, we have defined $\mathrm{Set}\ A$ as

$$A \to \mathrm{Prop}.$$

Moreover, if $(a : A)$ and $(S : \mathrm{Set}\ A)$, then $a \in S$ means

$$S\ a$$

We can now prove *extensionality for sets*. If $(S\ T : \mathrm{Set}\ A)$, then

$$S = T \iff \forall (a : A),\ a \in S \iff a \in T.$$

We already proved one implication, and the other is a direct application of functional extensionality and propositional extensionality.

The axiom of choice

Remember the definition of $\mathrm{Nonempty}\ A$

```
inductive Nonempty (A : Sort u) : Prop
| intro (val : A) : Nonempty A
```

It is an inductive *proposition* with only one constructor: if $(a : A)$, then

$$(\langle a \rangle : \mathrm{Nonempty}\ A)$$

The axiom of choice

Remember the definition of $\mathrm{Nonempty}\ A$

```
inductive Nonempty (A : Sort u) : Prop
| intro (val : A) : Nonempty A
```

It is an inductive *proposition* with only one constructor: if $(a : A)$, then

$$(\langle a \rangle : \mathrm{Nonempty}\ A)$$

It is easy to prove that

$$\mathrm{Nonempty}\ A \iff \exists (a : A), \mathrm{True}$$

We explained that $\mathrm{Nonempty}\ A$ does not contain data, and that it is impossible to build a function

$$\mathrm{default} : \mathrm{Nonempty}\ A \to A$$

such that

$$\mathrm{default}\ \langle a \rangle = a$$

for all $(a : A)$.

We explained that $\mathrm{Nonempty}\ A$ does not contain data, and that it is impossible to build a function

$$\mathrm{default} : \mathrm{Nonempty}\ A \to A$$

such that

$$\mathrm{default}\ \langle a \rangle = a$$

for all $(a : A)$.

The axiom of choice is the following function

```
axiom choice {A : Sort u} : Nonempty A → A
```

We have that choice magically produces an element of A given the assumption $\mathrm{Nonempty}\ A$. For Lean it is a well defined function (even if it has no definition).

We have that choice magically produces an element of A given the assumption $\mathrm{Nonempty}\ A$. For Lean it is a well defined function (even if it has no definition).

It is important to understand that if, say, $(h : \mathrm{Nonempty}\ \mathbb{N})$ (something easy to prove), the natural number $(\mathrm{choice}\ h : \mathbb{N})$ is *well defined and fixed once and for all*.

We have that choice magically produces an element of A given the assumption $\mathrm{Nonempty}\ A$. For Lean it is a well defined function (even if it has no definition).

It is important to understand that if, say, $(h : \mathrm{Nonempty}\ \mathbb{N})$ (something easy to prove), the natural number $(\mathrm{choice}\ h : \mathbb{N})$ is *well defined and fixed once and for all*. It is always the same, in all proofs where we use the axiom of choice.

We have that choice magically produces an element of A given the assumption $\mathrm{Nonempty}\ A$. For Lean it is a well defined function (even if it has no definition).

It is important to understand that if, say, $(h : \mathrm{Nonempty}\ \mathbb{N})$ (something easy to prove), the natural number $(\mathrm{choice}\ h : \mathbb{N})$ is *well defined and fixed once and for all*. It is always the same, in all proofs where we use the axiom of choice. Of course it is impossible to say anything about it (except that it is a natural number), but for Lean is perfectly well defined.

We have that choice magically produces an element of A given the assumption $\mathrm{Nonempty}\ A$. For Lean it is a well defined function (even if it has no definition).

It is important to understand that if, say, $(h : \mathrm{Nonempty}\ \mathbb{N})$ (something easy to prove), the natural number $(\mathrm{choice}\ h : \mathbb{N})$ is *well defined and fixed once and for all*. It is always the same, in all proofs where we use the axiom of choice. Of course it is impossible to say anything about it (except that it is a natural number), but for Lean is perfectly well defined. Moreover, it does not depend on h, since any other proof of $\mathrm{Nonempty}\ \mathbb{N}$ is definitionally equal to h by proof irrelevance.

We have that choice magically produces an element of A given the assumption $\mathrm{Nonempty}\ A$. For Lean it is a well defined function (even if it has no definition).

It is important to understand that if, say, $(h : \mathrm{Nonempty}\ \mathbb{N})$ (something easy to prove), the natural number $(\mathrm{choice}\ h : \mathbb{N})$ is *well defined and fixed once and for all*. It is always the same, in all proofs where we use the axiom of choice. Of course it is impossible to say anything about it (except that it is a natural number), but for Lean is perfectly well defined. Moreover, it does not depend on h, since any other proof of $\mathrm{Nonempty}\ \mathbb{N}$ is definitionally equal to h by proof irrelevance.

In practice we have fixed (once and for all) a term $(a : A)$ for all A such that $\mathrm{Nonempty}\ A$.

Remark

- *As we explained it is impossible to have*

$$\text{choice } \langle a \rangle = a$$

for all $(a : A)$.

Remark

- *As we explained it is impossible to have*

$$\mathrm{choice}\ \langle a \rangle = a$$

  for all $(a : A)$.
- *This version of the axiom of choice is slightly stronger than the usual version in set theory (the product of nonempty sets is nonempty) and it is called the axiom of global choice.*

Remark

- *As we explained it is impossible to have*

$$\text{choice}\ \langle a \rangle = a$$

for all $(a : A)$.

- *This version of the axiom of choice is slightly stronger than the usual version in set theory (the product of nonempty sets is nonempty) and it is called the axiom of global choice. But remember that Lean's type theory plus mathlib's three axioms is equivalent to ZFC plus the existence of countably many inaccessible cardinals.*

Remark

- *As we explained it is impossible to have*

$$\text{choice } \langle a \rangle = a$$

  for all $(a : A)$.
- *This version of the axiom of choice is slightly stronger than the usual version in set theory (the product of nonempty sets is nonempty) and it is called the axiom of global choice. But remember that Lean's type theory plus mathlib's three axioms is equivalent to ZFC plus the existence of countably many inaccessible cardinals. In particular the usual axiom of choice holds in Lean.*

Excluded middle

Excluded middle is the following property, for any $(P : \mathrm{Prop})$,

$$P \vee \neg P$$

Excluded middle

Excluded middle is the following property, for any $(P : \mathrm{Prop})$,

$$P \vee \neg P$$

Remember that by definition $\neg P$ is the implication $P \to \mathrm{False}$

Excluded middle

Excluded middle is the following property, for any $(P : \mathrm{Prop})$,

$$P \vee \neg P$$

Remember that by definition $\neg P$ is the implication $P \to \mathrm{False}$
Excluded middle says that P holds or P implies false.

Excluded middle

Excluded middle is the following property, for any $(P : \mathrm{Prop})$,

$$P \vee \neg P$$

Remember that by definition $\neg P$ is the implication $P \to \mathrm{False}$ Excluded middle says that P holds or P implies false. In classical logic it is one of the basic assumptions.

Excluded middle

Excluded middle is the following property, for any $(P : \mathrm{Prop})$,

$$P \vee \neg P$$

Remember that by definition $\neg P$ is the implication $P \to \mathrm{False}$ Excluded middle says that P holds or P implies false. In classical logic it is one of the basic assumptions.

It is used for example to do proofs by contradiction.

Excluded middle

Excluded middle is the following property, for any $(P : \mathrm{Prop})$,

$$P \vee \neg P$$

Remember that by definition $\neg P$ is the implication $P \to \mathrm{False}$ Excluded middle says that P holds or P implies false. In classical logic it is one of the basic assumptions.

It is used for example to do proofs by contradiction. To prove P we need to consider the two cases of $P \vee \neg P$:

- If P holds we are done (since we want to prove P! In practice this case is never explicitly considered).
- If $\neg P$ holds we still have to prove P, but of couse our assumption cannot help.

Excluded middle

Excluded middle is the following property, for any $(P : \mathrm{Prop})$,

$$P \vee \neg P$$

Remember that by definition $\neg P$ is the implication $P \to \mathrm{False}$ Excluded middle says that P holds or P implies false. In classical logic it is one of the basic assumptions.

It is used for example to do proofs by contradiction. To prove P we need to consider the two cases of $P \vee \neg P$:

- If P holds we are done (since we want to prove P! In practice this case is never explicitly considered).
- If $\neg P$ holds we still have to prove P, but of couse our assumption cannot help. But it is (as always) enough to prove False.

Excluded middle

Excluded middle is the following property, for any $(P : \mathrm{Prop})$,

$$P \vee \neg P$$

Remember that by definition $\neg P$ is the implication $P \to \mathrm{False}$ Excluded middle says that P holds or P implies false. In classical logic it is one of the basic assumptions.

It is used for example to do proofs by contradiction. To prove P we need to consider the two cases of $P \vee \neg P$:

- If P holds we are done (since we want to prove P! In practice this case is never explicitly considered).
- If $\neg P$ holds we still have to prove P, but of couse our assumption cannot help. But it is (as always) enough to prove False. So we can prove that $\neg P \to \mathrm{False}$, that is $\neg\neg P$.

Excluded middle is not assumed in Lean's type theory

Excluded middle is not assumed in Lean's type theory and indeed we didn't use it until now (and we never did a proof by contradiction).

Excluded middle is not assumed in Lean's type theory and indeed we didn't use it until now (and we never did a proof by contradiction).

Using the three new axioms we can now *prove* excluded middle.

Excluded middle is not assumed in Lean's type theory and indeed we didn't use it until now (and we never did a proof by contradiction).

Using the three new axioms we can now *prove* excluded middle.

Theorem (Diaconescu)

Let $(P : \mathrm{Prop})$*. Using functional extensionality (hence quotient types and propositional extensionality) and the axiom of (global) choice, we have*

$$P \vee \neg P$$

We will prove Diaconescu's theorem later, but let's first of all see some consequences.

Theorem

Let $(P : \mathrm{Prop})$*. Then* $P = \mathrm{True} \vee P = \mathrm{False}$*.*

Proof.

Using that $P \vee \neg P$ holds we have to consider two cases. In both we will use propositional extensionality.

- If P holds then it is easy to prove that $P \iff \mathrm{True}$ (since both hold), so $P = \mathrm{True}$.
- If $\neg P$ holds, we prove that $P \iff \mathrm{False}$, hence $P = \mathrm{False}$.

We will prove Diaconescu's theorem later, but let's first of all see some consequences.

Theorem

Let $(P : \mathrm{Prop})$. *Then* $P = \mathrm{True} \vee P = \mathrm{False}$.

Proof.

Using that $P \vee \neg P$ holds we have to consider two cases. In both we will use propositional extensionality.

- If P holds then it is easy to prove that $P \iff \mathrm{True}$ (since both hold), so $P = \mathrm{True}$.
- If $\neg P$ holds, we prove that $P \iff \mathrm{False}$, hence $P = \mathrm{False}$. To prove $P \Rightarrow \mathrm{False}$ note that this is exactly $\neg P$, that is our assumption.

We will prove Diaconescu's theorem later, but let's first of all see some consequences.

Theorem

Let $(P : \mathrm{Prop})$*. Then* $P = \mathrm{True} \vee P = \mathrm{False}$*.*

Proof.

Using that $P \vee \neg P$ holds we have to consider two cases. In both we will use propositional extensionality.

- If P holds then it is easy to prove that $P \iff \mathrm{True}$ (since both hold), so $P = \mathrm{True}$.
- If $\neg P$ holds, we prove that $P \iff \mathrm{False}$, hence $P = \mathrm{False}$. To prove $P \Rightarrow \mathrm{False}$ note that this is exactly $\neg P$, that is our assumption. The other implication is $\mathrm{False} \Rightarrow P$, that always holds (for any P).

$\square$

Theorem

For all $(P : \mathrm{Prop})$ *we have*

$$\neg\neg P \to P$$

Proof.

We consider again the two cases $P \vee \neg P$.

- If P holds there is nothing to prove.

Theorem

For all $(P : \mathrm{Prop})$ *we have*

$$\neg\neg P \to P$$

Proof.

We consider again the two cases $P \vee \neg P$.

- If P holds there is nothing to prove.
- Suppose that $\neg P$ holds, so that $P \Rightarrow \mathrm{False}$. To prove P it is enough to prove False (using $\mathrm{False.rec}$). Since we are supposing $\neg\neg P$ we can prove $\neg P$ and we are done.

$\square$

Corollary

Let $(P\ Q : \mathrm{Prop})$*. If both* $P \to Q$ *and* $\neg P \to Q$ *hold then* Q *holds.*

Proof.

It is an immediate consequence that $P \vee \neg P$ holds.

Corollary

Let $(P\ Q : \mathrm{Prop})$*. If both* $P \to Q$ *and* $\neg P \to Q$ *hold then* Q *holds.*

Proof.

It is an immediate consequence that $P \vee \neg P$ holds. □

In particular we can prove theorems by cases, supposing P or $\neg P$ holds: in Lean we can use the `by_cases` tactic.

Corollary

Let $(P\ Q : \mathrm{Prop})$. If both $P \to Q$ and $\neg P \to Q$ hold then Q holds.

Proof.

It is an immediate consequence that $P \vee \neg P$ holds. $\square$

In particular we can prove theorems by cases, supposing P or $\neg P$ holds: in Lean we can use the `by_cases` tactic.

To reason by contradiction (i.e. to use $\neg\neg P \to P$ to prove P) we can use the `by_contra'` tactic. Indeed, `by_contra' h` will create (and simplify) an assumption $(h : \neg P)$, where P is the current goal, and replace the goal with False.

We now give a slightly imprecise proof of excluded middle, We now give a slightly imprecise proof of excluded middle, ignoring the difference between sets and types (we will prove it in Lean).

We now give a slightly imprecise proof of excluded middle, ignoring the difference between sets and types (we will prove it in Lean). Let $(P : \mathrm{Prop})$, considered as the empty set or a singleton. We consider the following sets, given as functions $\mathrm{Prop} \to \mathrm{Prop}$

$$U := \mathrm{fun}\ x \mapsto (x = \mathrm{True}) \lor P$$
$$V := \mathrm{fun}\ x \mapsto (x = \mathrm{False}) \lor P$$

We now give a slightly imprecise proof of excluded middle, ignoring the difference between sets and types (we will prove it in Lean). Let $(P : \mathrm{Prop})$, considered as the empty set or a singleton. We consider the following sets, given as functions $\mathrm{Prop} \to \mathrm{Prop}$

$$U := \mathrm{fun}\ x \mapsto (x = \mathrm{True}) \vee P$$
$$V := \mathrm{fun}\ x \mapsto (x = \mathrm{False}) \vee P$$

If P holds, then both $(x = \mathrm{True}) \vee P$ and $(x = \mathrm{False}) \vee P$ hold, so these sets have the same elements. More precisely, we have the following lemma.

We now give a slightly imprecise proof of excluded middle, ignoring the difference between sets and types (we will prove it in Lean). Let $(P : \mathrm{Prop})$, considered as the empty set or a singleton. We consider the following sets, given as functions $\mathrm{Prop} \to \mathrm{Prop}$

$$U := \mathrm{fun}\ x \mapsto (x = \mathrm{True}) \vee P$$
$$V := \mathrm{fun}\ x \mapsto (x = \mathrm{False}) \vee P$$

If P holds, then both $(x = \mathrm{True}) \vee P$ and $(x = \mathrm{False}) \vee P$ hold, so these sets have the same elements. More precisely, we have the following lemma.

Lemma

If we have $(p : P)$ (i.e. if P holds), then

$$U = V.$$

Proof.

Suppose P holds. By extensionality, to prove $U = V$ we can prove that $(x \in U) = (x \in V)$ for all x.

Proof.

Suppose P holds. By extensionality, to prove $U = V$ we can prove that $(x \in U) = (x \in V)$ for all x. Since $(U\ V : \mathrm{Prop} \to \mathrm{Prop})$, this is the same as $U\ x = V\ x$ for all x. By propositional extensionality we can then prove that, for all x,

$$U\ x \iff V\ x$$

Let $(x : \mathrm{Prop})$ and let's prove the two implications.

Proof.

Suppose P holds. By extensionality, to prove $U = V$ we can prove that $(x \in U) = (x \in V)$ for all x. Since $(U\ V : \mathrm{Prop} \to \mathrm{Prop})$, this is the same as $U\ x = V\ x$ for all x. By propositional extensionality we can then prove that, for all x,

$$U\ x \iff V\ x$$

Let $(x : \mathrm{Prop})$ and let's prove the two implications.

- Let $(h : U\ x)$ be fixed (we will not use it). By definition, $V\ x$ is the proposition $(x = \mathrm{False}) \vee P$.

Proof.

Suppose P holds. By extensionality, to prove $U = V$ we can prove that $(x \in U) = (x \in V)$ for all x. Since $(U\ V : \mathrm{Prop} \to \mathrm{Prop})$, this is the same as $U\ x = V\ x$ for all x. By propositional extensionality we can then prove that, for all x,

$$U\ x \iff V\ x$$

Let $(x : \mathrm{Prop})$ and let's prove the two implications.

- Let $(h : U\ x)$ be fixed (we will not use it). By definition, $V\ x$ is the proposition $(x = \mathrm{False}) \vee P$. This holds because we supposed that P holds.

Proof.

Suppose P holds. By extensionality, to prove $U = V$ we can prove that $(x \in U) = (x \in V)$ for all x. Since $(U\ V : \mathrm{Prop} \to \mathrm{Prop})$, this is the same as $U\ x = V\ x$ for all x. By propositional extensionality we can then prove that, for all x,

$$U\ x \iff V\ x$$

Let $(x : \mathrm{Prop})$ and let's prove the two implications.

- Let $(h : U\ x)$ be fixed (we will not use it). By definition, $V\ x$ is the proposition $(x = \mathrm{False}) \vee P$. This holds because we supposed that P holds.
- Similarly, supposing $(h : V\ x)$, we can prove $U\ x$, that is $(x = \mathrm{True}) \vee P$, since P holds.

$\square$

Let's go back to the proof of excluded middle. By reflexivity of $=$, we have

$$U \ \mathrm{True} \text{ and } V \ \mathrm{False}$$

In particular,

$$\mathrm{True} \in U \text{ and } \mathrm{False} \in V$$

so we obtain

$$(\mathrm{exU} : \mathrm{Nonempty}\ U) \text{ and } (\mathrm{exV} : \mathrm{Nonempty}\ V)$$

Let's go back to the proof of excluded middle. By reflexivity of $=$, we have

$$U \ \mathrm{True} \text{ and } V \ \mathrm{False}$$

In particular,

$$\mathrm{True} \in U \text{ and } \mathrm{False} \in V$$

so we obtain

$$(\mathrm{exU} : \mathrm{Nonempty}\ U) \text{ and } (\mathrm{exV} : \mathrm{Nonempty}\ V)$$

Using the axiom of choice, we obtain propositions

$$u := \mathrm{choice\ exU} \text{ and } v := \mathrm{choice\ exV}$$

such that

$$u \in U \text{ and } v \in V$$

Lemma

Suppose P holds. Then, for all $(hU : \mathrm{Nonempty}\ U)$ *and for all* $(hV : \mathrm{Nonempty}\ V)$ *we have*

$$\mathrm{choice}\ hU = \mathrm{choice}\ hV$$

Proof.

If P holds then, by the previous lemma, we have $U = V$. It follows that hU and hV are two proofs of the same proposition, and in particular $hU \equiv hV$. The lemma is now immediate since choice is a well defined function.

Lemma

Suppose P holds. Then, for all $(hU : \mathrm{Nonempty}\ U)$ *and for all* $(hV : \mathrm{Nonempty}\ V)$ *we have*

$$\mathrm{choice}\ hU = \mathrm{choice}\ hV$$

Proof.

If P holds then, by the previous lemma, we have $U = V$. It follows that hU and hV are two proofs of the same proposition, and in particular $hU \equiv hV$. The lemma is now immediate since choice is a well defined function. $\square$

Remark

- *The fact that* choice *always gives the same term is crucial.*

Lemma

Suppose P holds. Then, for all $(hU : \mathrm{Nonempty}\ U)$ *and for all* $(hV : \mathrm{Nonempty}\ V)$ *we have*

$$\mathrm{choice}\ hU = \mathrm{choice}\ hV$$

Proof.

If P holds then, by the previous lemma, we have $U = V$. It follows that hU and hV are two proofs of the same proposition, and in particular $hU \equiv hV$. The lemma is now immediate since choice is a well defined function. □

Remark

- *The fact that* choice *always gives the same term is crucial.*
- *To use the eliminator for $=$ (in particular to find the motive), we need to state the theorem using "for all hU and for all hV". We cannot prove $u = v$ without generalizing them.*

We can now finish the proof. To prove $P \vee \neg P$, since $u \in U$ and $v \in V$, and U and V are defined by disjunction, we have four cases to consider.

We can now finish the proof. To prove $P \vee \neg P$, since $u \in U$ and $v \in V$, and U and V are defined by disjunction, we have four cases to consider. In three of these cases u or v have type P, so we have a term of type P and we can prove P.

We can now finish the proof. To prove $P \vee \neg P$, since $u \in U$ and $v \in V$, and U and V are defined by disjunction, we have four cases to consider. In three of these cases u or v have type P, so we have a term of type P and we can prove P.
The forth case is when u is True and v is False. In this case we prove $\neg P$.

We can now finish the proof. To prove $P \vee \neg P$, since $u \in U$ and $v \in V$, and U and V are defined by disjunction, we have four cases to consider. In three of these cases u or v have type P, so we have a term of type P and we can prove P.
The forth case is when u is True and v is False. In this case we prove $\neg P$. So we assume that P holds and we need to prove False.

We can now finish the proof. To prove $P \vee \neg P$, since $u \in U$ and $v \in V$, and U and V are defined by disjunction, we have four cases to consider. In three of these cases u or v have type P, so we have a term of type P and we can prove P.
The forth case is when u is True and v is False. In this case we prove $\neg P$. So we assume that P holds and we need to prove False.

Lemma

We have $u \neq v$.

Proof.

If $u = v$ we have that $\mathrm{True} = \mathrm{False}$ (since we are in the case $u = \mathrm{True}$ and $v = \mathrm{False}$). So, to prove False we can prove True, that always holds. □

Knowing that $u \neq v$, to prove False it is enough to prove that $u = v$.

Knowing that $u \neq v$, to prove False it is enough to prove that $u = v$. But we supposed that P holds, so this is a consequence of the lemma above.

Knowing that $u \neq v$, to prove False it is enough to prove that $u = v$. But we supposed that P holds, so this is a consequence of the lemma above.
We have considered all the cases, so this finishes the proof of excluded middle.

Knowing that $u \neq v$, to prove False it is enough to prove that $u = v$. But we supposed that P holds, so this is a consequence of the lemma above.
We have considered all the cases, so this finishes the proof of excluded middle.

Remark

The only problem with this proof is that we treated U and V, that are sets, as types. This is solved as follows. Given $(S : \mathrm{Set}\ A)$ we can form the type $\uparrow S$ whose terms are pairs $\langle a, h \rangle$ where $(a : A)$ and $(h : a \in S)$ (technically it is defined as an inductive type).

Knowing that $u \neq v$, to prove False it is enough to prove that $u = v$. But we supposed that P holds, so this is a consequence of the lemma above.
We have considered all the cases, so this finishes the proof of excluded middle.

Remark

The only problem with this proof is that we treated U and V, that are sets, as types. This is solved as follows. Given $(S : \mathrm{Set}\ A)$ we can form the type $\uparrow S$ whose terms are pairs $\langle a, h\rangle$ where $(a : A)$ and $(h : a \in S)$ (technically it is defined as an inductive type). In particular, if $(x : \uparrow S)$ then

$$(x.1 : A) \text{ and } (x.2 : x.1 \in S)$$

Replacing U and V with $\uparrow U$ and $\uparrow V$ makes the proof perfectly formal.