Category Theory in Coq 8.5

Amin Timany     Bart Jacobs

iMinds-Distrinet KU Leuven

FSCD’16 – Porto, Portugal
June 22, 2016
List of the most important formalized notions

- **basic constructions:**
  - terminal/initial object
  - products/sums
  - equalizers/coequalizers
  - pullbacks/pushouts
  - exponentials
  - + ⊣ Δ ⊣ × and (− × a) ⊣ a−

- **external constructions:**
  - comma categories
  - product category

- **for Cat:** (Obj := Category, Hom := Functor)
  - cartesian closure
  - initial object

- **for Set:** (Obj := Type, Hom := fun A B ⇒ A → B)
  - initial object
  - sums
  - equalizers
  - coequalizers†
  - pullbacks
  - cartesian closure
  - local cartesian closure†
  - completeness
  - co-completeness†
  - sub-object classifier (Prop : Type)†
  - topos†

† uses the axioms of propositional extensionality and constructive indefinite description (axiom of choice) to construct quotient types.

- the Yoneda lemma
adjunction
- hom-functor adjunction, unit-counit adjunction, unit-universal morphism
  adjunction, universal morphism adjunction and their conversions
- duality: $F \dashv G \Rightarrow G^{\text{op}} \dashv F^{\text{op}}$
- uniqueness up to natural isomorphism
- category of adjunctions

kan extensions
- global definition
- local definition with both hom-functor and cones (along a functor)
- uniqueness
- preservation by adjoint functors
- pointwise kan extensions (preserved by representable functors)

(co)limits
- as (left)right local kan extensions along the unique functor to the terminal category
- (sum)product-(co)equalizer (co)limits
- pointwise (as kan extensions)

Freyd’s adjoint functor theorem

$T - (co)algebras$ (for an endofunctor $T$)
- we use functional extensionality
- we use proof irrelevance in many cases (mostly for proof of equality of arrows)

This implementation can be found at:
https://github.com/amintimany/Categories
This implementation uses some features of Coq 8.5, most notably:

- Primitive projections for records:
- Universe polymorphism: for relative smallness/largeness
Primitive projections for records:
- The $\eta$ rule for records: two instances of a record type are \textit{definitionally} equal if all their respective projections are.
- E.g., for $\{|x : A; y : A|}$ and $f \ u = \{|x := y \ u; y := x \ u|}$, we have $f \ (f \ u) \equiv u$. 

For Categories: $(C^{\text{op}})^{\text{op}} \equiv C$

For Functors: $(F^{\text{op}})^{\text{op}} \equiv F$

For Natural Transformations: $(N^{\text{op}})^{\text{op}} \equiv N$
Primitive projections for records:

- The $\eta$ rule for records: two instance of a record type are \textit{definitionally} equal if all their respective projections are.

- E.g., for $\{ | x : A; y : A | \}$ and $f \ u = \{ | x := y \ u; y := x \ u | \}$, we have $f \ (f \ u) \equiv u$.

- This provides \textit{definitional} equalities, e.g.: (similar to Coq/HoTT implementation)
  - For Categories: $(C^{\text{op}})^{\text{op}} \equiv C$
  - For Functors: $(F^{\text{op}})^{\text{op}} \equiv F$
  - For Natural Transformations: $(N^{\text{op}})^{\text{op}} \equiv N$
Smallness and largeness

A category $C$ is said to be small if its collection objects and collections of morphisms form a set. It is called locally-small if for any two objects $A$ and $B$, the set of morphisms $\text{hom}_C(A,B)$ forms a set but objects themselves fail to, e.g., $\text{Set}$. It is called large otherwise.

In ZF we can’t talk about non-small categories (everything is a set). In NGB (von Neumann–Gödel–Bernays), locally small and large categories can be formalized (there is a notion of proper class) but they are not very useful. For instance, $\text{Set}$ can’t be defined as its objects are already proper classes and there is no class of proper classes.

ZF with Grothendieck universes doesn’t suffer from these restrictions.
Smallness and largeness

A category $C$ is said to be small if its collection objects and collections of morphisms form a set.
- Smallness and largeness
  - A category $\mathcal{C}$ is said to be small if its collection objects and collections of morphisms form a set.
  - It is called locally-small if for any two objects $A$ and $B$, the set of morphisms $\text{hom}_\mathcal{C}(A, B)$ forms a set but objects themselves fail to, e.g., $\textbf{Set}$.

In ZF we can't talk about non-small categories (everything is a set), but in NGB (von Neumann–Gödel–Bernays), locally small and large categories can be formalized (there is a notion of proper class). However, they are not very useful. For instance, $\textbf{Set}$ can't be defined as its objects are already proper classes and there is no class of proper classes. ZF with Grothendieck universes doesn't suffer from these restrictions.
Smallness and largeness

- A category $C$ is said to be small if its collection objects and collections of morphisms form a set.

- It is called locally-small if for any two objects $A$ and $B$, the set of morphisms $\text{hom}_C(A, B)$ forms a set but objects themselves fail to, e.g., $\text{Set}$.

- It is called large otherwise.
Smallness and largeness

- A category $C$ is said to be small if its collection objects and collections of morphisms form a set.

- It is called locally-small if for any two objects $A$ and $B$, the set of morphisms $\text{hom}_C(A, B)$ forms a set but objects themselves fail to, e.g., $\text{Set}$.

- It is called large otherwise.

- In ZF we can’t talk about non-small categories (everything is a set).
Smallness and largeness

- A category $C$ is said to be small if its collection objects and collections of morphisms form a set.

- It is called locally-small if for any two objects $A$ and $B$, the set of morphisms $\text{hom}_C(A, B)$ forms a set but objects themselves fail to, e.g., $\text{Set}$.

- It is called large otherwise.

- In ZF we can’t talk about non-small categories (everything is a set).

- In NGB (von Neumann–Gödel–Bernays), locally small and large categories can be formalized (there is a notion of proper class) but they are not very useful.
Smallness and largeness

- A category $\mathcal{C}$ is said to be small if its collection objects and collections of morphisms form a set.

- It is called locally-small if for any two objects $A$ and $B$, the set of morphisms $\text{hom}_\mathcal{C}(A, B)$ forms a set but objects themselves fail to, e.g., Set.

- It is called large otherwise.

- In ZF we can’t talk about non-small categories (everything is a set).

- In NGB (von Neumann–Gödel–Bernays), locally small and large categories can be formalized (there is a notion of proper class) but they are not very useful.

- For instance, $\text{Set}^{\text{Set}}$ can’t be defined as its objects are already proper classes and there is no class of proper classes.
Smallness and largeness

- A category $\mathcal{C}$ is said to be small if its collection objects and collections of morphisms form a set.
- It is called locally-small if for any two objects $A$ and $B$, the set of morphisms $\text{hom}_\mathcal{C}(A, B)$ forms a set but objects themselves fail to, e.g., $\textbf{Set}$.
- It is called large otherwise.
- In ZF we can’t talk about non-small categories (everything is a set).
- In NGB (von Neumann–Gödel–Bernays), locally small and large categories can be formalized (there is a notion of proper class) but they are not very useful.
- For instance, $\textbf{Set}^{\text{Set}}$ can’t be defined as its objects are already proper classes and there is no class of proper classes.
- ZF with Grothendieck universes doesn’t suffer from this restrictions.
A Grothendieck universe is a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U} \Rightarrow \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U} \Rightarrow \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U} \Rightarrow \exists 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$

etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms. The Grothendieck Axiom (GA) states that for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$. GA + ZF implies a hierarchy of universes, as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{U} \in \mathcal{U}$ violates ZF axioms.

A set $A$ is said to be $\mathcal{U}$-small if $A \in \mathcal{U}$. A category is $\mathcal{U}$-small if the collection of its objects and the collection of its morphisms are $\mathcal{U}$-small. It is called $\mathcal{U}$-locally-small if its collections of morphisms are $\mathcal{U}$-small but not its objects, e.g., $\text{Set}_\mathcal{U}$: the category of sets in $\mathcal{U}$. It is called $\mathcal{U}$-large otherwise.

Note the similarity with Coq universes:

Coq's universe hierarchy:

- $\text{Type}_0$
- $\text{Type}_1$
- $\text{Type}_2$
- ... 

Cumulativity: $T : \text{Type}_i$ and $i \leq j$ then $T : \text{Type}_j$.

Function types: $A : \text{Type}_i$ and $B : \text{Type}_i$, $A \to B : \text{Type}_i$.

We can derive:

$A, B \in \mathcal{U} \Rightarrow B^A \in \mathcal{U}$

We use Coq's universes to represent relative smallness/largeness.
A Grothendieck universe a set $\mathcal{U}$ such that

\[ \forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U} \]
A Grothendieck universe is a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$

In short, a Grothendieck universe is a set that satisfies ZF axioms.

Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$.

GA + ZF imply a hierarchy of universes as for a universe $V$, there is a $\mathcal{U}$ such that $V \in \mathcal{U}$ and $V \in \mathcal{U}$ violates ZF axioms.

A set $A$ is said to be $\mathcal{U}$-small if $A \in \mathcal{U}$.

A category is $\mathcal{U}$-small if the collection of its objects and collections of its morphisms are $\mathcal{U}$-small.

It is called $\mathcal{U}$-locally-small if its collections of morphisms are $\mathcal{U}$-small but not its objects, e.g., $\text{Set}_\mathcal{U}$: the category of sets in $\mathcal{U}$.

It is called $\mathcal{U}$-large otherwise.

Note the similarity with Coq universes:

Coq's universe hierarchy:

- $\text{Type}_0$
- $\text{Type}_1$
- $\text{Type}_2$
- . . .

Cumulativity: $T : \text{Type}_i \text{ and } i \leq j \Rightarrow T : \text{Type}_j$

Function types: $A : \text{Type}_i \text{ and } B : \text{Type}_i \Rightarrow A \rightarrow B : \text{Type}_i$.

We can derive: $A, B \in \mathcal{U} \Rightarrow B \in \mathcal{U}$.

We use Coq's universes to represent relative smallness/largeness.
A Grothendieck universe is a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- A Grothendieck universe a set $U$ such that
  - $\forall A, B \in U. \{A, B\} \in U$
  - $\forall I \in U. \forall i \in I, A_i \in U \Rightarrow \bigcup_{i \in I} A_i \in U$
  - $\forall A \in U. 2^A \in U$
  - $\forall A, B. A \in B \land B \in U \Rightarrow A \in U$
A Grothendieck universe is a set $\mathcal{U}$ such that:

- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
- etc.
A Grothendieck universe is a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
- etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms.
A Grothendieck universe a set $\mathcal{U}$ such that
- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \wedge B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
- etc.
- In short, a Grothendieck universe is a set that satisfies ZF axioms

Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$
A Grothendieck universe is a set $\mathcal{U}$ such that
- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
- etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms.

Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$.

GA + ZF imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms.
A Grothendieck universe a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
- etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms

**Grothendieck Axiom (GA):** for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$

**GA + ZF** imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms

A set $A$ is said to be $\mathcal{U}$-small if $A \in \mathcal{U}$
- A Grothendieck universe is a set $\mathcal{U}$ such that
  - $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
  - $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
  - $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
  - $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
  - etc.

- In short, a Grothendieck universe is a set that satisfies ZF axioms

- Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$

- GA + ZF imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms

- A set $A$ is said to be $\mathcal{U}$-small if $A \in \mathcal{U}$

- A category is $\mathcal{U}$-small if the collection of its objects and collections its morphisms are $\mathcal{U}$-small
- A Grothendieck universe a set $\mathcal{U}$ such that
  - $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
  - $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
  - $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
  - $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
  - etc.
  - In short, a Grothendieck universe is a set that satisfies ZF axioms

- Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$
- GA + ZF imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms
- A set $A$ is said to be $\mathcal{U}$-small if $A \in \mathcal{U}$
- A category is $\mathcal{U}$-small if the collection of its objects and collections its morphisms are $\mathcal{U}$-small
- It is called $\mathcal{U}$-locally-small if its collections of morphisms are $\mathcal{U}$-small but not its objects, e.g., $\text{Set}_\mathcal{U}$: the category of sets in $\mathcal{U}$
A Grothendieck universe is a set \( U \) such that
- \( \forall A, B \in U. \{A, B\} \in U \)
- \( \forall I \in U. \forall i \in I, A_i \in U \Rightarrow \bigcup_{i \in I} A_i \in U \)
- \( \forall A \in U. 2^A \in U \)
- \( \forall A, B. A \in B \land B \in U \Rightarrow A \in U \)
- etc.
- In short, a Grothendieck universe is a set that satisfies ZF axioms.

Grothendieck Axiom (GA): for any set \( A \), there is a universe set \( U \) such that \( A \in U \).

GA + ZF imply a hierarchy of universes as for a universe \( V \), there is a \( U \) such that \( V \in U \) and \( V \in U \) violates ZF axioms.

A set \( A \) is said to be \( U \)-small if \( A \in U \).

A category is \( U \)-small if the collection of its objects and collections its morphisms are \( U \)-small.

It is called \( U \)-locally-small if its collections of morphisms are \( U \)-small but not its objects, e.g., \( \text{Set}_U \): the category of sets in \( U \).

It is called \( U \)-large otherwise.
A Grothendieck universe is a set $\mathcal{U}$ such that
- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
- etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms

Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$

GA + ZF imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms
A Grothendieck universe is a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$

etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms.

Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$.

GA + ZF imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms.

Note the similarity with Coq universes:
A Grothendieck universe is a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U}. \ \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \ \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. \ 2^A \in \mathcal{U}$
- $\forall A, B. \ A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
- etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms.

Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$

GA + ZF imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms (universe hierarchy)

Note the similarity with Coq universes:

- Coq’s universe hierarchy: $\text{Type}_0 : \text{Type}_1 : \text{Type}_2 : \ldots$
A Grothendieck universe a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$ (Cumulativity: when $B$ is a universe)
- etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms

- Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$
- GA + ZF imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms (universe hierarchy)

- Note the similarity with Coq universes:
  - Coq’s universe hierarchy: $\text{Type}_0 : \text{Type}_1 : \text{Type}_2 : \ldots$
  - Cumulativity: $T : \text{Type}_i$ and $i \leq j$ then $T : \text{Type}_j$
A Grothendieck universe is a set \( \mathcal{U} \) such that:

- \( \forall A, B \in \mathcal{U}. \{ A, B \} \in \mathcal{U} \)
- \( \forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U} \)
- \( \forall A \in \mathcal{U}. 2^A \in \mathcal{U} \)
- \( \forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U} \) (Cumulativity: when \( B \) is a universe)
- etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms.

Grothendieck Axiom (GA): for any set \( A \), there is a universe set \( \mathcal{U} \) such that \( A \in \mathcal{U} \).

GA + ZF imply a hierarchy of universes as for a universe \( \mathcal{V} \), there is a \( \mathcal{U} \) such that \( \mathcal{V} \in \mathcal{U} \) and \( \mathcal{V} \in \mathcal{U} \) violates ZF axioms (universe hierarchy).

Note the similarity with Coq universes:

- Coq’s universe hierarchy: \( \text{Type}_0 : \text{Type}_1 : \text{Type}_2 : \ldots \)
- Cumulativity: \( T : \text{Type}_i \) and \( i \leq j \) then \( T : \text{Type}_j \)
- Function types: \( A : \text{Type}_i \) and \( B : \text{Type}_i \), \( A \rightarrow B : \text{Type}_i \)

we can derive: \( A, B \in \mathcal{U} \Rightarrow B^A \in \mathcal{U} \)
A Grothendieck universe is a set $\mathcal{U}$ such that

- $\forall A, B \in \mathcal{U}. \{A, B\} \in \mathcal{U}$
- $\forall I \in \mathcal{U}. \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
- $\forall A \in \mathcal{U}. 2^A \in \mathcal{U}$
- $\forall A, B. A \in B \land B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$ (Cumulativity: when $B$ is a universe)
- etc.

In short, a Grothendieck universe is a set that satisfies ZF axioms

Grothendieck Axiom (GA): for any set $A$, there is a universe set $\mathcal{U}$ such that $A \in \mathcal{U}$

GA + ZF imply a hierarchy of universes as for a universe $\mathcal{V}$, there is a $\mathcal{U}$ such that $\mathcal{V} \in \mathcal{U}$ and $\mathcal{V} \in \mathcal{U}$ violates ZF axioms (universe hierarchy)

Note the similarity with Coq universes:

- Coq’s universe hierarchy: $\text{Type}_0 : \text{Type}_1 : \text{Type}_2 : \ldots$
- Cumulativity: $T : \text{Type}_i$ and $i \leq j$ then $T : \text{Type}_j$
- Function types: $A : \text{Type}_i$ and $B : \text{Type}_i$, $A \rightarrow B : \text{Type}_i$
  we can derive: $A, B \in \mathcal{U} \Rightarrow B^A \in \mathcal{U}$
- We use Coq’s universes to represent relative smallness/largeness
In Coq without universe polymorphism
In Coq without universe polymorphism

\begin{verbatim}
Definition Tp := Type
\end{verbatim}
In Coq **without universe polymorphism**

```
Definition Tp := Type
```

is internally represented as: (for a fixed \( i \))

```
Definition Tp := Type@{i}
```

Therefore,

```
Definition Tp Tp := Tp
```

is rejected as it would require the constraint

\( i < i \)

With universe polymorphism enabled

```
Tp above is internally represented as:
```

```
Definition Tp@{i} := Type@{i}
```

where \( i \) is a parameter of the definition.

Therefore:

```
Definition Tp Tp@{i j} := Tp@{i} := Tp@{j}
```

with the side constraint:

\( j < i \)

Universe levels in definitions and theorems are inferred by Coq and never appear in our source code (it is possible to manually specify universe levels and constraints in Coq).
In Coq without universe polymorphism

```
Definition Tp := Type
```

is internally represented as: (for a fixed $i$)

```
Definition Tp := Type@\{i\}
```

Therefore,

```
Definition TpTp : Tp := Tp
```

is rejected as it would require the constraint $i < i$
In Coq without universe polymorphism

Definition Tp := Type

is internally represented as: (for a fixed i)

Definition Tp := Type@{i}

Therefore,

Definition Tp Tp : Tp := Tp

is rejected as it would require the constraint i < i

With universe polymorphism enabled

Tp above is internally represented as:

Definition Tp@{i} := Type@{i}

where i is a parameter of the definition.
In Coq **without universe polymorphism**

```
Definition Tp := Type
```

is internally represented as: (for a fixed \( i \))

```
Definition Tp := Type@{i}
```

Therefore,

```
Definition TpTp : Tp := Tp
```

is rejected as it would require the constraint \( i < i \)

**With universe polymorphism enabled**

\( Tp \) above is internally represented as:

```
Definition Tp@{i} := Type@{i}
```

where \( i \) is a parameter of the definition. Therefore:

```
Definition TpTp : Tp := Tp
```

is accepted
In Coq **without universe polymorphism**

Definition $Tp := Type$

is internally represented as: (for a fixed $i$)

Definition $Tp := Type@\{i\}$

Therefore,

Definition $Tp Tp : Tp := Tp$

is rejected as it would require the constraint $i < i$

**With universe polymorphism enabled**

$Tp$ above is internally represented as:

Definition $Tp@\{i\} := Type@\{i\}$

where $i$ is a parameter of the definition. Therefore:

Definition $Tp Tp : Tp := Tp$

is accepted and internally represented as:

Definition $Tp Tp@\{i \ j\} : Tp@\{i\} := Tp@\{j\}$

with the side constraint: $j < i$
In Coq **without universe polymorphism**

```
Definition Tp := Type
```

is internally represented as: (for a fixed \(i\))

```
Definition Tp := Type@\{i\}
```

Therefore,

```
Definition TpTp : Tp := Tp
```

is rejected as it would require the constraint \(i < i\)

**With universe polymorphism enabled**

Tp above is internally represented as:

```
Definition Tp@\{i\} := Type@\{i\}
```

where \(i\) is a parameter of the definition. Therefore:

```
Definition TpTp : Tp := Tp
```

is accepted and internally represented as:

```
Definition TpTp@\{i j\} : Tp@\{i\} := Tp@\{j\}
```

with the side constraint: \(j < i\)

Universe levels in definitions and theorems are inferred by Coq and **never** appear in our source code.
In Coq **without universe polymorphism**

```coq
Definition Tp := Type
```

is internally represented as: (for a fixed $i$)

```coq
Definition Tp := Type\@\{i\}
```

Therefore,

```coq
Definition TpTp : Tp := Tp
```

is rejected as it would require the constraint $i < i$

With universe polymorphism enabled

$Tp$ above is internally represented as:

```coq
Definition Tp\@\{i\} := Type\@\{i\}
```

where $i$ is a parameter of the definition. Therefore:

```coq
Definition TpTp : Tp := Tp
```

is accepted and internally represented as:

```coq
Definition TpTp\@\{i j\} : Tp\@\{i\} := Tp\@\{j\}
```

with the side constraint: $j < i$

Universe levels in definitions and theorems are inferred by Coq and **never** appear in our source code (it is possible to manually specify universe levels and constraints in Coq)
Universe polymorphism: for relative smallness/largeness
Universe polymorphism: for relative smallness/largeness

Record Category@{i j} : Type@{max(i+1, j+1)} :=
{ Obj : Type@{i} Hom : Obj → Obj → Type@{j} id : forall a : Obj, Hom a a compose : forall a b c, (f : Hom a b) (g : Hom c d) : Hom a c
Universe polymorphism: for relative smallness/largeness

Record Category@{i j} : Type@{max(i+1, j+1)} :=
{ Obj : Type@{i} Hom : Obj → Obj → Type@{j} id : forall a : Obj, Hom a a compose : forall a b c, (f : Hom a b) (g : Hom c d) : Hom a c }

Category is universe polymorphic
Universe polymorphism: for relative smallness/largeness

```
Record Category@{i j} : Type@{max(i+1, j+1)} :=
{
  Obj : Type@{i}
  Hom : Obj → Obj → Type@{j}
  id : forall a : Obj, Hom a a
  compose : forall a b c, (f : Hom a b) (g : Hom c d) : Hom a c
  : 
}
```

Category is universe polymorphic

For each pair of levels \((i, j)\), \(\text{Category@}{i, j}\) is a copy at level \((i, j)\)
Universe polymorphism: for relative smallness/largeness

Record Category@{i j} : Type@{max(i+1, j+1)} :=
{ 
  Obj : Type@{i}
  Hom : Obj → Obj → Type@{j}
  id : forall a : Obj, Hom a a
  compose : forall a b c, (f : Hom a b) (g : Hom c d) : Hom a c
  :
}

Category is universe polymorphic

For each pair of levels \((i, j)\), Category@{i, j} is a copy at level \((i, j)\)

For each definition, theorem, etc., we get some constraints on universe levels
Universe polymorphism: for relative smallness/largeness

Record Category@{i j} : Type@{max(i+1, j+1)} :=
{ Obj : Type@{i}
  Hom : Obj → Obj → Type@{j}
  id : forall a : Obj, Hom a a
  compose : forall a b c, (f : Hom a b) (g : Hom c d) : Hom a c
  ...
}

Category is universe polymorphic

- For each pair of levels \((i, j)\), Category@{i, j} is a copy at level \((i, j)\)
- For each definition, theorem, etc., we get some constraints on universe levels
- The definition, theorem, etc. only works for those copies that satisfy these side constraint
This notion of smallness/largeness using universe levels works so well that we can define \textbf{Cat} directly as:

\begin{itemize}
  \item \textbf{Instance} \texttt{Cat : Category := \{Obj := Category; Hom := Functor; \ldots\}}
\end{itemize}
This notion of smallness/largeness using universe levels works so well that we can define \textbf{Cat} directly as:

\begin{itemize}
  \item \textbf{Instance Cat} : \texttt{Category} := \{\texttt{Obj} := \texttt{Category}; \texttt{Hom} := \texttt{Functor}; \ldots\}\n  \item Or prove the following for a (particular but fixed) \( z \):
  \begin{align*}
    \textbf{Theorem Complete_Preorder} \ (C : \texttt{Category}) \ (CC : \texttt{Complete} \ C) : \forall \ x \ y \ : \ (\texttt{Obj} \ C), \ \texttt{Hom} \ x \ z \ \simeq \ ((\texttt{Arrow} \ C) \to \texttt{Hom} \ x \ y)
  \end{align*}
\end{itemize}
This notion of smallness/largeness using universe levels works so well that we can define $\textbf{Cat}$ directly as:

**Instance** $\textbf{Cat} : \text{Category} := \{\text{Obj} := \text{Category}; \text{Hom} := \text{Functor}; \ldots\}$

Or prove the following for a (particular but fixed) $z$:

**Theorem** Complete_Preorder ($C : \text{Category}$) ($\text{CC} : \text{Complete} C$):

forall $x$ $y : (\text{Obj} C)$, $\text{Hom} x z \simeq ((\text{Arrow} C) \to \text{Hom} x y)$

This theorem results in a contradiction as soon as there are objects $a$ and $b$ in $C$ such that $|\text{hom}(a, b)| \geq 2$
This notion of smallness/largeness using universe levels works so well that we can define \textbf{Cat} directly as:

\textbf{Instance} \textbf{Cat} : \textbf{Category} := \{\textbf{Obj} := \textbf{Category}; \textbf{Hom} := \textbf{Functor}; \ldots\}\n
Or prove the following for a (particular but fixed) \( z \):

\textbf{Theorem} Complete\_Preorder \((C : \textbf{Category})(CC : \textbf{Complete}\ C) : \\forall x y : (\textbf{Obj} C), \textbf{Hom} x z \simeq ((\textbf{Arrow} C) \rightarrow \textbf{Hom} x y)\)

This theorem results in a contradiction as soon as there are objects \( a \) and \( b \) in \( C \) such that \(|\text{hom}(a, b)| \geq 2\)

In other words, it says that any complete category is a mere pre-order relation
This notion of smallness/largeness using universe levels works so well that we can define \textbf{Cat} directly as:

\textbf{Instance Cat : Category := \{Obj := Category; Hom := Functor; \ldots\}}

Or prove the following for a (particular but fixed) \(z\):

\textbf{Theorem Complete_Preorder (C : Category) (CC : Complete C) :
  forall x y : (Obj C), Hom x z \simeq ((Arrow C) \rightarrow Hom x y)}

This theorem results in a contradiction as soon as there are objects \(a\) and \(b\) in \(C\) such that \(|\text{hom}(a, b)| \geq 2\)

In other words, it says that any complete category is a mere pre-order relation

This theorem indeed holds, but only for small categories
This notion of smallness/largeness using universe levels works so well that we can define **Cat** directly as:

**Instance Cat : Category := {Obj := Category; Hom := Functor; ...}**

Or prove the following for a (particular but fixed) \( z \):

**Theorem Complete_Preorder (C : Category) (CC : Complete C) :**

\[
\forall x \, y : \text{Obj} \, C, \text{Hom} \, x \, z \simeq ((\text{Arrow} \, C) \to \text{Hom} \, x \, y)
\]

This theorem results in a contradiction as soon as there are objects \( a \) and \( b \) in \( C \) such that \( |\text{hom}(a, b)| \geq 2 \)

In other words, it says that any complete category is a mere pre-order relation

This theorem indeed holds, but only for small categories

This can be seen in universe constraints of this theorem
This notion of smallness/largeness using universe levels works so well that we can define \texttt{Cat} directly as:

\begin{verbatim}
Instance \texttt{Cat} : Category := \{Obj := Category; Hom := Functor; ...\}
\end{verbatim}

Or prove the following for a (particular but fixed) \( z \):

\textbf{Theorem Complete_Preorder} (\( C : \text{Category} \)) (\( \text{CC} : \text{Complete } C \)) :
  \hspace{1em} \forall x y : (\text{Obj } C), \text{Hom } x z \simeq ((\text{Arrow } C) \to \text{Hom } x y)

This theorem results in a contradiction as soon as there are objects \( a \) and \( b \) in \( C \) such that \( |\text{hom}(a, b)| \geq 2 \)

In other words, it says that any complete category is a mere pre-order relation

This theorem indeed holds, but only for small categories

This can be seen in universe constraints of this theorem

  For \( \text{C} : \text{Category}@\{k, l\} \) we get the restriction that \( k \leq l \)
  This is in contradiction with the fact that \( \text{Set} : \text{Category}@\{m, n\} \) with \( m > n \)
This notion of smallness/largeness using universe levels works so well that we can define \textbf{Cat} directly as:

\begin{itemize}
  \item \textbf{Instance} \textbf{Cat} : \textbf{Category} := \{\textbf{Obj} := \textbf{Category}; \textbf{Hom} := \textbf{Functor}; \ldots\}\end{itemize}

Or prove the following for a (particular but fixed) \(z\):

\textbf{Theorem} \textbf{Complete_Preorder} (\(C : \textbf{Category}\)) (\(CC : \textbf{Complete} C\)) :

\begin{verbatim}
forall x y : (\textbf{Obj} C), Hom x z \simeq ((\textbf{Arrow} C) \rightarrow Hom x y)
\end{verbatim}

This theorem results in a contradiction as soon as there are objects \(a\) and \(b\) in \(C\) such that \(|hom(a, b)| \geq 2\)

In other words, it says that any complete category is a mere pre-order relation

This theorem indeed holds, but only for small categories

This can be seen in universe constraints of this theorem

\begin{itemize}
  \item For \(C : \textbf{Category}@\{k, l}\) we get the restriction that \(k \leq l\)
  \item This is in contradiction with the fact that \textbf{Set} : \textbf{Category}@\{m, n\} with \(m > n\)
\end{itemize}

\textbf{Set}@\{m, n\} :=

\begin{verbatim}
{| |
  \textbf{Obj} := \textbf{Type}@\{n\} : \textbf{Type}@\{m\};
  \textbf{Hom} := \textbf{fun} A B \Rightarrow A \rightarrow B : \textbf{Obj} \rightarrow \textbf{Obj} \rightarrow \textbf{Type}@\{n\}; \ldots
|} : \textbf{Category}@\{m, n\}
\end{verbatim}
There are also cases where universe polymorphism of Coq is not flexible enough

Remember the definition of \texttt{Cat} in Coq:

\begin{verbatim}
Instance Cat : Category @ {i, j}:={ Obj := Category @ {k, l};
Hom := Functor; ... }
\end{verbatim}

But according to Coq’s universe polymorphism, if \texttt{C : Category @ {k, l}} and \texttt{C : Category @ {k', l'}}, we must have \(k = k'\) and \(l = l'\).

This means \(\texttt{Cat} @ \{i, j, k, l\}\) is not the category of all categories at level \((k, l)\) or lower but only at level \((k, l)\).

We can lift category:

\begin{verbatim}
lift (C : Category @ {k, l}) : Category @ {k', l'} := { \{ Obj := Obj C; Hom := Hom C; ... \} }
\end{verbatim}

for \(k < k'\) and \(l < l'\).

But we can’t prove or even specify (universe inconsistency)

\begin{verbatim}
forall (C : Category), C = lift C
\end{verbatim}

This makes working with liftings impractical.

This issue (and the next which is similar) would have been solved if Coq had cumulativity for inductive types. In such a system, \(C : \texttt{Category} @ \{i, j\}\) we also have \(C : \texttt{Category} @ \{k, l\}\) if \(i \leq k\) and \(j \leq l\).
There are also cases where universe polymorphism of Coq is not flexible enough.

These are more hindrances than any real issues though.
There are also cases where universe polymorphism of Coq is not flexible enough.
These are more hindrances than any real issues though.
Remember the definition of \texttt{Cat} in Coq:

\begin{verbatim}
Instance \texttt{Cat} : Category@\{i, j\} := \{Obj := Category@\{k, l\}; Hom := Functor; \ldots\}
\end{verbatim}
- There are also cases where universe polymorphism of Coq is not flexible enough.
- These are more hindrances than any real issues though.
- Remember the definition of `Cat` in Coq:
  
  ```coq
  Instance Cat : Category@{i, j} := {Obj := Category@{k, l}; Hom := Functor; ...}
  ```

- But according to Coq’s universe polymorphism, if `C : Category@{k, l}` and `C : Category@{k', l'}`, we must have `k = k'` and `l = l'`.
There are also cases where universe polymorphism of Coq is not flexible enough
These are more hindrances than any real issues though
Remember the definition of $\textbf{Cat}$ in Coq:

\[ \text{Instance } \textbf{Cat} : \text{Category} @ \{i, j\} := \{ \text{Obj} := \text{Category} @ \{k, l\}; \text{Hom} := \text{Functor}; \ldots \} \]

But according to Coq’s universe polymorphism, if $C : \text{Category} @ \{k, l\}$ and $C : \text{Category} @ \{k’, l’\}$, we must have $k = k’$ and $l = l’$
This means $\text{Cat} @ \{i, j, k, l\}$ is not the category of all categories at level $(k, l)$ or lower but only at level $(k, l)$
There are also cases where universe polymorphism of Coq is not flexible enough.

These are more hindrances than any real issues though.

Remember the definition of \texttt{Cat} in Coq:

\begin{verbatim}
Instance \texttt{Cat} : \texttt{Category@\{i, j\} := \{Obj := \texttt{Category@\{k, l\}}; Hom := \texttt{Functor}; ...\}}
\end{verbatim}

But according to Coq’s universe polymorphism, if \(\texttt{C} : \texttt{Category@\{k, l\}}\) and \(\texttt{C} : \texttt{Category@\{k', l'\}}\), we must have \(k = k'\) and \(l = l'\).

This means \texttt{Cat@\{i, j, k, l\}} is not the category of all categories at level \((k, l)\) or lower but \textit{only} at level \((k, l)\).

We can lift category:

\begin{verbatim}
\texttt{lift (C : Category@\{k, l\}) : Category@\{k', l'\} := \{| Obj := Obj C; Hom := Hom C; ... \} for k < k' and l < l'\}}
\end{verbatim}
There are also cases where universe polymorphism of Coq is not flexible enough.

These are more hindrances than any real issues though.

Remember the definition of \textbf{Cat} in Coq:

\begin{verbatim}
Instance Cat : Category@{i, j} := {Obj := Category@{k, l}; Hom := Functor; ...}
\end{verbatim}

But according to Coq’s universe polymorphism, if \( C : \text{Category}@{k, l} \) and \( C : \text{Category}@{k’, l’} \), we must have \( k = k’ \) and \( l = l’ \).

This means \( \text{Cat}@{i, j, k, l} \) is not the category of all categories at level \((k, l)\) or lower but \textit{only} at level \((k, l)\).

We can lift category:

\begin{verbatim}
lift (C : \text{Category}@{k, l}) : \text{Category}@{k’, l’} := {{Obj := Obj C; Hom := Hom C; ...}}
\end{verbatim}

for \( k < k’ \) and \( l < l’ \).

But we can’t prove or even specify (universe inconsistency)

\begin{verbatim}
forall (C : \text{Category}), C = lift C
\end{verbatim}
There are also cases where universe polymorphism of Coq is not flexible enough.
These are more hindrances than any real issues though.
Remember the definition of \texttt{Cat} in Coq:
\begin{verbatim}
Instance \texttt{Cat} : Category@\{i, j\} := \{\texttt{Obj} := Category@\{k, l\}; \texttt{Hom} := \texttt{Functor}; ...\}
\end{verbatim}
But according to Coq’s universe polymorphism, if \(C : \text{Category}@\{k, l\}\) and \(C : \text{Category}@\{k', l'\}\), we must have \(k = k'\) and \(l = l'\).
This means \texttt{Cat}@\{i, j, k, l\} is not the category of all categories at level \((k, l)\) or lower but \emph{only} at level \((k, l)\).
We can lift category:
\begin{verbatim}
\texttt{lift} (C : Category@\{k, l\}) : Category@\{k', l'\} := \{| \texttt{Obj} := \texttt{Obj} C; \texttt{Hom} := \texttt{Hom} C; ... |\}
\end{verbatim}
for \(k < k'\) and \(l < l'\).
But we can’t prove or even specify (universe inconsistency)
\begin{verbatim}
\texttt{forall} (C : \text{Category}), C = \texttt{lift} C
\end{verbatim}
This makes working with liftings impractical.
There are also cases where universe polymorphism of Coq is not flexible enough

These are more hindrances than any real issues though

Remember the definition of \texttt{Cat} in Coq:

\begin{verbatim}
Instance \texttt{Cat} : Category@\{i, j\} := \{Obj := Category@\{k, l\}; Hom := Functor; ...\}
\end{verbatim}

But according to Coq’s universe polymorphism, if \(C : \text{Category}@\{k, l\}\) and \(C : \text{Category}@\{k’, l’\}\), we must have \(k = k’\) and \(l = l’\)

This means \texttt{Cat@\{i, j, k, l\}} is not the category of all categories at level \((k, l)\) or lower but \textit{only} at level \((k, l)\)

We can lift category:

\begin{verbatim}
lift (C : Category@\{k, l\}) : Category@\{k’, l’\} := \{| Obj := Obj C; Hom := Hom C; ... |\}
\end{verbatim}

for \(k < k’\) and \(l < l’\)

But we can’t prove or even specify (universe inconsistency)

\begin{verbatim}
forall (C : Category), C = lift C
\end{verbatim}

This makes working with liftings impractical

This issue (and the next which is similar) would have been solved if Coq had cumulativity for inductive types
There are also cases where universe polymorphism of Coq is not flexible enough.

These are more hindrances than any real issues though.

Remember the definition of `Cat` in Coq:

```
Instance Cat : Category@{i, j} := {Obj := Category@{k, l}; Hom := Functor; ...}
```

But according to Coq’s universe polymorphism, if `C : Category@{k, l}` and `C : Category@{k', l'}`, we must have `k = k'` and `l = l'`.

This means `Cat@{i, j, k, l}` is not the category of all categories at level `(k, l)` or lower but *only* at level `(k, l)`.

We can lift category:

```
lift (C : Category@{k, l}) : Category@{k', l'} := {Obj := Obj C; Hom := Hom C; ...}
```

for `k < k'` and `l < l'`.

But we can’t prove or even specify (universe inconsistency)

```
forall (C : Category), C = lift C
```

This makes working with liftings impractical.

This issue (and the next which is similar) would have been solved if Coq had cumulativity for inductive types.

In such a system `C : Category@{i j}` we also have `C : Category@{k l}` if `i ≤ k` and `j ≤ l`.
If we show that $\text{Cat}^\otimes\{i, j, k, l\}$ has exponentials, we get the constraints that $j = k = l$.
- If we show that $\text{Cat} @ \{i, j, k, l\}$ has exponentials, we get the constraints that $j = k = l$
- Therefore, no copy of $\text{Set}$ is in a copy of $\text{Cat}$ in which we have exponentials
If we show that \( \text{Cat}^\circ \{i, j, k, l\} \) has exponentials, we get the constraints that \( j = k = l \).

Therefore, no copy of \( \text{Set} \) is in a copy of \( \text{Cat} \) in which we have exponentials.

That means we can’t define Yoneda embedding as exponential transpose (currying) of the hom functor.
If we show that $\text{Cat}_{i,j,k,l}$ has exponentials, we get the constraints that $j = k = l$

Therefore, no copy of $\text{Set}$ is in a copy of $\text{Cat}$ in which we have exponentials

That means we can’t define Yoneda embedding as exponential transpose (currying) of the hom functor

Defining Yoneda separately, it still can only be applied in a category $C : \text{Category}_{i,j}$ if $i = j$. 
If we show that \( \text{Cat}^{\{i, j, k, l\}} \) has exponentials, we get the constraints that
\( j = k = l \)

Therefore, no copy of \( \text{Set} \) is in a copy of \( \text{Cat} \) in which we have exponentials.

That means we can’t define Yoneda embedding as exponential transpose (currying) of the hom functor.

Defining Yoneda separately, it still can only be applied in a category \( C : \text{Category}^{\{i, j\}} \) if \( i = j \).

We can use Yoneda to prove that in any cartesian closed category:
\[
(a^b)^c \cong a^{b \times c}
\]

but this lemma can’t be applied to \( \text{Cat} \) or \( \text{Set} \).
Conclusion:
- We presented an implementation of category theory covering some of the basic category theory.

You can find the Coq development at:
https://github.com/amintimany/Categories

Thanks for your attention!

Amin Timany
Bart Jacobs
Conclusion:

- We presented an implementation of category theory covering some of the basic category theory.
- We use features of Coq 8.5: primitive projections and universe polymorphism.

You can find the Coq development at:

https://github.com/amintimany/Categories

Thanks for your attention!

Amin Timany
Bart Jacobs
Conclusion:

- We presented an implementation of category theory covering some of the basic category theory.
- We use features of Coq 8.5: primitive projections and universe polymorphism.
- Universe polymorphism to represent smallness/largeness.

As is usually done with Grothendieck universes, this works so well that we don’t need to mention any universe levels manually. And can prove things like:

\[ \text{Cat} \quad \text{and} \quad \text{Complete_Preorder} \]

It also has shortcomings: e.g., can’t use Yoneda in \[ \text{Cat} \quad \text{and} \quad \text{Set} \] You can find the Coq development at:

https://github.com/amintimany/Categories

Thanks for your attention!

Amin Timany

Bart Jacobs

Category Theory in Coq 8.5
Conclusion:

- We presented an implementation of category theory covering some of the basic category theory
- We use features of Coq 8.5: primitive projections and universe polymorphism
- Universe polymorphism to represent smallness/largeness
  - As is usually done with Grothendieck universes

You can find the Coq development at: 
https://github.com/amintimany/Categories

Thanks for your attention!
Conclusion:

- We presented an implementation of category theory covering some of the basic category theory.
- We use features of Coq 8.5: primitive projections and universe polymorphism.
- Universe polymorphism to represent smallness/largeness.
  - As is usually done with Grothendieck universes.
  - This works so well that we don’t need to mention any universe levels manually.

You can find the Coq development at: https://github.com/amintimany/Categories.

Thanks for your attention!

Amin Timany
Bart Jacobs
Conclusion:

- We presented an implementation of category theory covering some of the basic category theory
- We use features of Coq 8.5: primitive projections and universe polymorphism
- Universe polymorphism to represent smallness/largeness
  - As is usually done with Grothendieck universes
  - This works so well that we don’t need to mention any universe levels manually
  - And can prove things like: \texttt{Cat} and \texttt{Complete_Preorder}
Conclusion:

- We presented an implementation of category theory covering some of the basic category theory
- We use features of Coq 8.5: primitive projections and universe polymorphism
- Universe polymorphism to represent smallness/largeness
  - As is usually done with Grothendieck universes
  - This works so well that we don’t need to mention any universe levels manually
  - And can prove things like: \texttt{Cat} and \texttt{Complete_Preorder}
  - It also has shortcomings: e.g., can’t use Yoneda in \texttt{Cat} and \texttt{Set}

You can find the Coq development at:

https://github.com/amintimany/Categories

Thanks for your attention!