# 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
  - $+ \dashv \Delta \dashv \times$ and $(- \times a) \dashv a^{-}$
- external constructions:
  - comma categories
  - product category
- for **Cat**: $(\texttt{Obj} := \texttt{Category}, \texttt{Hom} := \texttt{Functor})$
  - cartesian closure
  - initial object
- for **Set**: $(\texttt{Obj} := \texttt{Type}, \texttt{Hom} := \texttt{fun A B} \Rightarrow \texttt{A} \to \texttt{B})$
  - initial object
  - sums
  - equalizers
  - coequalizers$^\dagger$
  - pullbacks
  - cartesian closure
  - local cartesian closure$^\dagger$
  - completeness
  - co-completeness$^\dagger$
  - sub-object classifier $(\texttt{Prop} : \texttt{Type})^\dagger$
  - topos$^\dagger$

  $^\dagger$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 : $F \dashv G \Rightarrow G^{op} \dashv F^{op}$
  - 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 instance of a record type are *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`

- Primitive projections for records:
  - The $\eta$ rule for records: two instance of a record type are *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 *definitional* equalities, e.g.: (similar to Coq/HoTT implementation)
    - For Categories: `(C^op)^op` $\equiv$ `C`
    - For Functors: `(F^op)^op` $\equiv$ `F`
    - For Natural Transformations: `(N^op)^op` $\equiv$ `N`

- Smallness and largeness

- Smallness and largeness
  - A category $\mathcal{C}$ 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 $hom_{\mathcal{C}}(A, B)$ forms a set but objects themselves fail to, e.g., **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 $hom_{\mathcal{C}}(A, B)$ forms a set but objects themselves fail to, e.g., **Set**
  - It is called large otherwise

- 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 $hom_{\mathcal{C}}(A, B)$ 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)

- 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 $hom_{\mathcal{C}}(A, B)$ 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

- 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 $hom_{\mathcal{C}}(A, B)$ 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, $\mathbf{Set}^{\mathbf{Set}}$ 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 $hom_{\mathcal{C}}(A, B)$ 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, $\mathbf{Set}^{\mathbf{Set}}$ 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 a set $\mathcal{U}$ such that

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

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

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

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

- 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.

- 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

- 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 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 $A \in \mathcal{U}$
- 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
  - $\forall A, B \in \mathcal{U}.\ \{A, B\} \in \mathcal{U}$
  - $\forall I \in \mathcal{U}.\ \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
  - $\forall A \in \mathcal{U}.\ 2^A \in \mathcal{U}$
  - $\forall A, B.\ A \in B \wedge B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
  - 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 $A \in \mathcal{U}$
- 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 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 $A \in \mathcal{U}$
- 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
  - $\forall A, B \in \mathcal{U}.\ \{A, B\} \in \mathcal{U}$
  - $\forall I \in \mathcal{U}.\ \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
  - $\forall A \in \mathcal{U}.\ 2^A \in \mathcal{U}$
  - $\forall A, B.\ A \in B \wedge B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
  - 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 $A \in \mathcal{U}$
- 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., $\mathbf{Set}_{\mathcal{U}}$: the category of sets in $\mathcal{U}$

- 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 $A \in \mathcal{U}$
- 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., $\mathbf{Set}_{\mathcal{U}}$: the category of sets in $\mathcal{U}$
- It is called $\mathcal{U}$-large otherwise

- 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 $A \in \mathcal{U}$
- 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
  - $\forall A, B \in \mathcal{U}.\ \{A, B\} \in \mathcal{U}$
  - $\forall I \in \mathcal{U}.\ \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
  - $\forall A \in \mathcal{U}.\ 2^A \in \mathcal{U}$
  - $\forall A, B.\ A \in B \wedge B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$
  - 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 $A \in \mathcal{U}$
- 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 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 $A \in \mathcal{U}$
- 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: $\mathtt{Type}_0 : \mathtt{Type}_1 : \mathtt{Type}_2 : \ldots$

- A Grothendieck universe a set $\mathcal{U}$ such that
  - $\forall A, B \in \mathcal{U}.\ \{A, B\} \in \mathcal{U}$
  - $\forall I \in \mathcal{U}.\ \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
  - $\forall A \in \mathcal{U}.\ 2^A \in \mathcal{U}$
  - $\forall A, B.\ A \in B \wedge B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$ (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: $\texttt{Type}_0 : \texttt{Type}_1 : \texttt{Type}_2 : \ldots$
- Cumulativity: $\texttt{T} : \texttt{Type}_i$ and $i \leq j$ then $\texttt{T} : \texttt{Type}_j$

- A Grothendieck universe a set $\mathcal{U}$ such that
  - $\forall A, B \in \mathcal{U}.\ \{A, B\} \in \mathcal{U}$
  - $\forall I \in \mathcal{U}.\ \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
  - $\forall A \in \mathcal{U}.\ 2^A \in \mathcal{U}$
  - $\forall A, B.\ A \in B \wedge B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$ (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: $\mathtt{Type}_0 : \mathtt{Type}_1 : \mathtt{Type}_2 : \ldots$
- Cumulativity: $\mathtt{T} : \mathtt{Type}_i$ and $i \leq j$ then $\mathtt{T} : \mathtt{Type}_j$
- Function types: $\mathtt{A} : \mathtt{Type}_i$ and $\mathtt{B} : \mathtt{Type}_i$, $\mathtt{A} \rightarrow \mathtt{B} : \mathtt{Type}_i$

  we can derive: $A, B \in \mathcal{U} \Rightarrow B^A \in \mathcal{U}$

- A Grothendieck universe a set $\mathcal{U}$ such that
  - $\forall A, B \in \mathcal{U}.\ \{A, B\} \in \mathcal{U}$
  - $\forall I \in \mathcal{U}.\ \forall i \in I, A_i \in \mathcal{U} \Rightarrow \bigcup_{i \in I} A_i \in \mathcal{U}$
  - $\forall A \in \mathcal{U}.\ 2^A \in \mathcal{U}$
  - $\forall A, B.\ A \in B \wedge B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$ (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: $\mathtt{Type}_0 : \mathtt{Type}_1 : \mathtt{Type}_2 : \ldots$
- Cumulativity: $\mathtt{T} : \mathtt{Type}_i$ and $i \leq j$ then $\mathtt{T} : \mathtt{Type}_j$
- Function types: $\mathtt{A} : \mathtt{Type}_i$ and $\mathtt{B} : \mathtt{Type}_i$, $\mathtt{A} \rightarrow \mathtt{B} : \mathtt{Type}_i$

  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**

```
Definition Tp := Type
```

- In Coq **without universe polymorphism**

```
Definition Tp := Type
```

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

```
Definition Tp := Type@{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`

- 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.

- 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 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`

- 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 $\mathtt{i} < \mathtt{i}$

- **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: $\mathtt{j} < \mathtt{i}$

- Universe levels in definitions and theorems are inferred by Coq and **never** appear in our source code

- 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 (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 $(i, j)$, `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 **Cat** directly as:
- `Instance` **Cat** : `Category` := {`Obj` := `Category`; `Hom` := `Functor`; ...}

- 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 : (Obj C), Hom x z ≃ ((Arrow C) → Hom x y)
```

- 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 : (Obj C), Hom x z ≃ ((Arrow C) → Hom x y)
```

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

- 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 : (Obj C), Hom x z ≃ ((Arrow C) → Hom x y)
```

- 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 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 : (Obj C), Hom x z ≃ ((Arrow C) → Hom x y)
```

- 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 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 : (Obj C), Hom x z ≃ ((Arrow C) → Hom x y)
```

- 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

- 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 : (Obj C), Hom x z ≃ ((Arrow C) → Hom x y)
```

- 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
  - For `C : Category@{k, l}` we get the restriction that `k` $\leq$ `l`
  - This is in contradiction with the fact that **Set** : `Category@{m, n}` with `m` $>$ `n`

- 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 : (Obj C), Hom x z ≃ ((Arrow C) → Hom x y)
```

- 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
  - For `C : Category@{k, l}` we get the restriction that `k` $\leq$ `l`
  - This is in contradiction with the fact that **Set** : `Category@{m, n}` with `m` $>$ `n`

```
Set@{m, n} :=
{|
  Obj := Type@{n} : Type@{m};
  Hom := fun A B ⇒ A → B : Obj → Obj → Type@{n}; ...
|} : Category@{m, n}
```

- There are also cases where universe polymorphism of Coq is not flexible enough

- 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 **Cat** in Coq:

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

- 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'`

- 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`)

- 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 $(\mathtt{k}, \mathtt{l})$ or lower but *only* at level $(\mathtt{k}, \mathtt{l})$
- We can lift category:

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

  for $\mathtt{k} < \mathtt{k}'$ and $\mathtt{l} < \mathtt{l}'$

- 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 **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 $(\mathtt{k}, \mathtt{l})$ or lower but *only* at level $(\mathtt{k}, \mathtt{l})$
- We can lift category:

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

  for $\mathtt{k} < \mathtt{k}'$ and $\mathtt{l} < \mathtt{l}'$
- But we can't prove or even specify (universe inconsistency) `forall (C : Category), C = lift C`

- 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 $(\mathtt{k}, \mathtt{l})$ or lower but *only* at level $(\mathtt{k}, \mathtt{l})$
- We can lift category:

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

  for $\mathtt{k} < \mathtt{k}'$ and $\mathtt{l} < \mathtt{l}'$

- But we can't prove or even specify (universe inconsistency)
  `forall (C : Category), C = lift C`
- 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 **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 $(\mathtt{k}, \mathtt{l})$ or lower but *only* at level $(\mathtt{k}, \mathtt{l})$
- We can lift category:

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

  for $\mathtt{k} < \mathtt{k}'$ and $\mathtt{l} < \mathtt{l}'$
- 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

- 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 $(\mathtt{k}, \mathtt{l})$ or lower but *only* at level $(\mathtt{k}, \mathtt{l})$
- We can lift category:

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

  for $\mathtt{k} < \mathtt{k}'$ and $\mathtt{l} < \mathtt{l}'$
- 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 $\mathcal{C}$ : `Category@{i j}` we also have $\mathcal{C}$ : `Category@{k l}` if $\mathtt{i} \leq \mathtt{k}$ and $\mathtt{j} \leq \mathtt{l}$

- If we show that **Cat**`@{i, j, k, l}` has exponentials, we get the constraints that $j = k = l$

- If we show that **Cat**@{`i`, `j`, `k`, `l`} has exponentials, we get the constraints that $j = k = l$
- Therefore, no copy of **Set** is in a copy of **Cat** in which we have exponentials

- If we show that **Cat**`@{i, j, k, l}` has exponentials, we get the constraints that $j = k = l$
- Therefore, no copy of **Set** is in a copy of **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 **Cat**@{i, j, k, l} has exponentials, we get the constraints that $j = k = l$
- Therefore, no copy of **Set** is in a copy of **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 : Category@{i, j}` if $i = j$.

- If we show that **Cat**`@{i, j, k, l}` has exponentials, we get the constraints that $j = k = l$
- Therefore, no copy of **Set** is in a copy of **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 : Category@{i, j}` if $i = j$.
- We can use Yoneda to prove that in any cartesian closed category:

$$\left(a^b\right)^c \simeq a^{b \times c}$$

but this lemma can't be applied to **Cat** or **Set**

- Conclusion:
  - We presented an implementation of category theory covering some of the basic category theory

- 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

- 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

- 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

- 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

- 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: **Cat** and `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: **Cat** and `Complete_Preorder`
    - It also has shortcomings: e.g., can't use Yoneda in **Cat** and **Set**

- You can find the Coq development at:

`https://github.com/amintimany/Categories`

Thanks for your attention!