Cumulative Inductive Types in Coq

Amin Timany[1] Matthieu Sozeau[2]

[1]imec-Distrinet, KU Leuven, Belgium

[2]Inria Paris & IRIF, France

November 14th, 2017

Géocal-LAC Days, Nantes, France

What are universes?

Universes are the types of *types*, e.g:

- $\texttt{nat}, \texttt{bool} : \texttt{Type}_0$
- $\texttt{Type}_0 : \texttt{Type}_1$
- $\texttt{list} : \texttt{Type}_0 \to \texttt{Type}_0$
- $\forall \alpha : \texttt{Type}_0, \texttt{list}\ \alpha : \texttt{Type}_1$
- $\forall n : \texttt{nat}, \{n = 0\} + \{n \neq 0\} : \texttt{Type}_0$

How are they organised?

A *hierarchy* of predicative universes $\texttt{Type}_0 < \texttt{Type}_1 < \dots$

- Avoids the $\texttt{Type} : \texttt{Type}$ paradox (system U^-)
- Replicates Russell's paradox of $\{x : x \notin x\}$, the set of all sets etc....
- Think of $\texttt{Type}_0$ as sets, $\texttt{Type}_1$ as classes etc...

Universe hierarchy

- In higher order dependent type theories, we have a countably infinite hierarchy of universes (types of types):

$$\mathtt{Type}_0, \mathtt{Type}_1, \mathtt{Type}_2, \ldots$$

where:

$$\mathtt{Type}_0 : \mathtt{Type}_1, \mathtt{Type}_1 : \mathtt{Type}_2, \ldots$$

Universe hierarchy

- In higher order dependent type theories, we have a countably infinite hierarchy of universes (types of types):

$$\texttt{Type}_0, \texttt{Type}_1, \texttt{Type}_2, \ldots$$

  where:

$$\texttt{Type}_0 : \texttt{Type}_1, \texttt{Type}_1 : \texttt{Type}_2, \ldots$$

- Such a system is cumulative if for any type T and i:

$$T : \texttt{Type}_i \Rightarrow T : \texttt{Type}_{i+1}$$

- Example: Predicative Calculus of Inductive Constructions (pCIC), the logic of the proof assistant Coq

Universe cumulativity in pCIC

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash B : s \qquad A \preceq B}{\Gamma \vdash t : B}$$

$$\frac{i \leq j}{\texttt{Type}_i \preceq \texttt{Type}_j} \qquad\qquad \frac{A \simeq A' \qquad B \preceq B'}{\Pi x : A.\ B \preceq \Pi x : A'.\ B'}$$

Universe polymorphism

- pCIC has recently been extended with universe polymorphism
  - Definitions can be polymorphic in universe levels, e.g., categories:

```
Record Category@{i j} : Type@{max(i+1, j+1)} :=
  { Obj : Type@{i};
    Hom : Obj → Obj → Type@{j}; ... }.
```

Universe polymorphism

- pCIC has recently been extended with universe polymorphism
  - Definitions can be polymorphic in universe levels, e.g., categories:

```
Record Category@{i j} : Type@{max(i+1, j+1)} :=
  { Obj : Type@{i};
    Hom : Obj → Obj → Type@{j}; ... }.
```

  - To keep consistency, universe polymorphic definitions come with constraints, e.g., category of categories:

```
Definition Cat@{i j k l} :=
  {| Obj := Category@{k l};
     Hom := fun C D ⇒ Functor@{k l k l} C D; ... |}
  : Category@{i j}.
```

with constraints:

$$k < i \quad \text{and} \quad l < i$$

Justifying universe polymorphism

- For universe polymorphic inductive types and constants, e.g., `Category`, copies are considered (Sozeau & Tabareau, ITP'14).
- Direct translation from pCIC to CIC + floating universe constraints, itself translatable to CIC with fixed universes.
- With no cumulativity (subtyping), i.e., `Category@{i j}` $\preceq$ `Category@{k l}` iff `i` $=$ `k` and `j` $=$ `l`

Justifying universe polymorphism

- For universe polymorphic inductive types and constants, e.g., `Category`, copies are considered (Sozeau & Tabareau, ITP'14).
- Direct translation from pCIC to CIC + floating universe constraints, itself translatable to CIC with fixed universes.
- With no cumulativity (subtyping), i.e., `Category@{i j}` $\preceq$ `Category@{k l}` iff `i` $=$ `k` and `j` $=$ `l`
- This means `Cat@{i j k l}` is the category of all categories at {`k l`} and *not lower* [1]

[1] There are however categories isomorphic to the categories in lower levels.

Relative size constraints

- Constraints on statements about universe polymorphic inductive definitions restrict to which copies they apply.
- For the category of categories `Cat@{i j k l}` the fact that it has exponentials has constraints $\mathtt{j} = \mathtt{k} = \mathtt{l}$:
  universe of objects $\mathtt{k}$ = universe of morphisms $\mathtt{l}$ = universe of functors between `k l` categories.

Relative size constraints

- Constraints on statements about universe polymorphic inductive definitions restrict to which copies they apply.
- For the category of categories `Cat@{i j k l}` the fact that it has exponentials has constraints `j` = `k` = `l`: universe of objects `k` = universe of morphisms `l` = universe of functors between `k` `l` categories.
- In particular:

```
Definition Type_Cat@{i j} :=
  {| Obj := Type@{j};
     Hom := fun A B ⇒ A → B; ... |} : Category@{i j}.
```

with constraints: $j < i$ is not an object of any copy of `Cat` with exponentials!

Relative size constraints

- Constraints on statements about universe polymorphic inductive definitions restrict to which copies they apply.
- For the category of categories `Cat@{i j k l}` the fact that it has exponentials has constraints `j` = `k` = `l`:
  universe of objects `k` = universe of morphisms `l` = universe of functors between `k` `l` categories.
- In particular:

```
Definition Type_Cat@{i j} :=
  {| Obj := Type@{j};
     Hom := fun A B ⇒ A → B; ... |} : Category@{i j}.
```

  with constraints: $j < i$ is not an object of any copy of `Cat` with exponentials!
- Yoneda embedding can't be simply defined as the exponential transpose of the *hom* functor

Inductive types in pCIC

$$\frac{\begin{array}{l}\textsc{Ind}\\ A \in Ar(s) \qquad \Gamma \vdash A : s' \qquad \Gamma, X : A \vdash C_i : s \qquad C_i \in Co(X)\end{array}}{\Gamma \vdash \mathsf{Ind}(X : A)\{C_1, \ldots, C_n\} : A}$$

$Ar(s)$ is the set of types of the form: $\Pi \vec{x} : \vec{M}.\ s$

$Co(X)$ is the set of types of the form: $\Pi \vec{x} : \vec{M}.\ X\ \vec{m}$

Inductive types in pCIC

$$\textsc{Ind}\ \frac{A \in Ar(s) \qquad \Gamma \vdash A : s' \qquad \Gamma, X : A \vdash C_i : s \qquad C_i \in Co(X)}{\Gamma \vdash \mathrm{Ind}(X : A)\{C_1, \ldots, C_n\} : A}$$

$$Ar(s) \text{ is the set of types of the form: } \Pi \vec{x} : \vec{M}.\ s$$

$$Co(X) \text{ is the set of types of the form: } \Pi \vec{x} : \vec{M}.\ X\ \vec{m}$$

No Parameters (`T` in `vec T n`) are considered in this rule.

```
Inductive vec (T : Type) : nat → Type := nil : vec T 0
  | cons : forall n, T → vec T n → vec T (S n).
```

Inductive types in pCIC

$$\frac{\begin{array}{cccc}A \in Ar(s) & \Gamma \vdash A : s' & \Gamma, X : A \vdash C_i : s & C_i \in Co(X)\end{array}}{\Gamma \vdash \mathrm{Ind}(X : A)\{C_1, \ldots, C_n\} : A}\ \textsc{Ind}$$

$Ar(s)$ is the set of types of the form: $\Pi\vec{x} : \vec{M}.\ s$

$Co(X)$ is the set of types of the form: $\Pi\vec{x} : \vec{M}.\ X\ \vec{m}$

No Parameters (`T` in `vec T n`) are considered in this rule.

```
Inductive vec (T : Type) : nat → Type := nil : vec T 0
  | cons : forall n, T → vec T n → vec T (S n).
```

Predicative Calculus of Cumulative Inductive Types (pCuIC)

C-Ind

$$\frac{\begin{array}{c} I \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}.\ s)\{\Pi\vec{x_1} : \vec{M}_1.\ X\ \vec{m}_1, \ldots, \Pi\vec{x_n} : \vec{M}_n.\ X\ \vec{m}_n\}) \\ I' \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N'}.\ s')\{\Pi\vec{x_1} : \vec{M'_1}.\ X\ \vec{m'_1}, \ldots, \Pi\vec{x_n} : \vec{M'_n}.\ X\ \vec{m'_n}\}) \\ \forall i.\ N_i \preceq N'_i \qquad \forall i, j.\ (M_i)_j \preceq (M'_i)_j \\ \mathit{length}(\vec{m}) = \mathit{length}(\vec{x}) \qquad \forall i.\ X\ \vec{m}_i \simeq X\ \vec{m'_i} \end{array}}{I\ \vec{m} \preceq I'\ \vec{m}}$$

Predicative Calculus of Cumulative Inductive Types (pCuIC)

$$
\textsc{C-Ind}\quad
\frac{\begin{array}{c}
I \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}.\ s)\{\Pi\vec{x_1} : \vec{M}_1.\ X\ \vec{m}_1, \ldots, \Pi\vec{x_n} : \vec{M}_n.\ X\ \vec{m}_n\}) \\
I' \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N'}.\ s')\{\Pi\vec{x_1} : \vec{M}'_1.\ X\ \vec{m}'_1, \ldots, \Pi\vec{x_n} : \vec{M}'_n.\ X\ \vec{m}'_n\}) \\
\boxed{\forall i.\ N_i \preceq N'_i} \qquad \forall i, j.\ (M_i)_j \preceq (M'_i)_j \\
\mathit{length}(\vec{m}) = \mathit{length}(\vec{x}) \qquad \forall i.\ X\ \vec{m}_i \simeq X\ \vec{m}'_i
\end{array}}{I\ \vec{m} \preceq I'\ \vec{m}}
$$

Predicative Calculus of Cumulative Inductive Types (pCuIC)

$$\textsc{C-Ind}\quad \frac{\begin{array}{c} I \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}.\ s)\{\Pi\vec{x_1} : \vec{M}_1.\ X\ \vec{m}_1, \ldots, \Pi\vec{x_n} : \vec{M}_n.\ X\ \vec{m}_n\}) \\ I' \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}'.\ s')\{\Pi\vec{x_1} : \vec{M}'_1.\ X\ \vec{m}'_1, \ldots, \Pi\vec{x_n} : \vec{M}'_n.\ X\ \vec{m}'_n\}) \\ \forall i.\ N_i \preceq N'_i \qquad \forall i,j.\ (M_i)_j \preceq (M'_i)_j \\ \mathit{length}(\vec{m}) = \mathit{length}(\vec{x}) \qquad \forall i.\ X\ \vec{m}_i \simeq X\ \vec{m}'_i \end{array}}{I\ \vec{m} \preceq I'\ \vec{m}}$$

Predicative Calculus of Cumulative Inductive Types (pCuIC)

C-Ind

$$\frac{\begin{array}{c} I \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}.\ s)\{\Pi\vec{x_1} : \vec{M}_1.\ X\ \vec{m}_1, \ldots, \Pi\vec{x_n} : \vec{M}_n.\ X\ \vec{m}_n\}) \\ I' \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N'}.\ s')\{\Pi\vec{x_1} : \vec{M}'_1.\ X\ \vec{m}'_1, \ldots, \Pi\vec{x_n} : \vec{M}'_n.\ X\ \vec{m}'_n\}) \\ \forall i.\ N_i \preceq N'_i \qquad \forall i, j.\ (M_i)_j \preceq (M'_i)_j \\ length(\vec{m}) = length(\vec{x}) \qquad \boxed{\forall i.\ X\ \vec{m}_i \simeq X\ \vec{m}'_i} \end{array}}{I\ \vec{m} \preceq I'\ \vec{m}}$$

Predicative Calculus of Cumulative Inductive Types (pCuIC)

$$\textsc{C-Ind}\quad \dfrac{\begin{array}{c} I \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}.\ s)\{\Pi\vec{x_1} : \vec{M}_1.\ X\ \vec{m}_1, \ldots, \Pi\vec{x_n} : \vec{M}_n.\ X\ \vec{m}_n\}) \\ I' \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}'.\ s')\{\Pi\vec{x_1} : \vec{M}'_1.\ X\ \vec{m}'_1, \ldots, \Pi\vec{x_n} : \vec{M}'_n.\ X\ \vec{m}'_n\}) \\ \forall i.\ N_i \preceq N'_i \qquad \forall i, j.\ (M_i)_j \preceq (M'_i)_j \\ \boxed{\mathit{length}(\vec{m}) = \mathit{length}(\vec{x})} \qquad \forall i.\ X\ \vec{m}_i \simeq X\ \vec{m}'_i \end{array}}{I\ \vec{m} \preceq I'\ \vec{m}}$$

Predicative Calculus of Cumulative Inductive Types (pCuIC)

$$
\dfrac{\begin{array}{c} I \equiv (\mathsf{Ind}(X : \Pi \vec{x} : \vec{N}.\ s)\{\Pi \vec{x_1} : \vec{M}_1.\ X\ \vec{m}_1, \ldots, \Pi \vec{x_n} : \vec{M}_n.\ X\ \vec{m}_n\}) \\ I' \equiv (\mathsf{Ind}(X : \Pi \vec{x} : \vec{N'}.\ s')\{\Pi \vec{x_1} : \vec{M}'_1.\ X\ \vec{m}'_1, \ldots, \Pi \vec{x_n} : \vec{M}'_n.\ X\ \vec{m}'_n\}) \\ \forall i.\ N_i \preceq N'_i \qquad \forall i, j.\ (M_i)_j \preceq (M'_i)_j \\ \mathit{length}(\vec{m}) = \mathit{length}(\vec{x}) \qquad \forall i.\ X\ \vec{m}_i \simeq X\ \vec{m}'_i \end{array}}{I\ \vec{m} \preceq I'\ \vec{m}}\ \textsc{C-Ind}
$$

Predicative Calculus of Cumulative Inductive Types (pCuIC)

$$\frac{\begin{array}{c} I \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}.\ s)\{\Pi\vec{x_1} : \vec{M}_1.\ X\ \vec{m}_1, \ldots, \Pi\vec{x_n} : \vec{M}_n.\ X\ \vec{m}_n\}) \\ I' \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N'}.\ s')\{\Pi\vec{x_1} : \vec{M'_1}.\ X\ \vec{m'_1}, \ldots, \Pi\vec{x_n} : \vec{M'_n}.\ X\ \vec{m'_n}\}) \\ \forall i.\ N_i \preceq N'_i \qquad \forall i, j.\ (M_i)_j \preceq (M'_i)_j \\ length(\vec{m}) = length(\vec{x}) \qquad \forall i.\ X\ \vec{m}_i \simeq X\ \vec{m'_i} \end{array}}{I\ \vec{m} \preceq I'\ \vec{m}}\ \textsc{C-Ind}$$

- Example:

  $$\texttt{Category@}\{\texttt{i j}\} \equiv \mathsf{Ind}(X : \texttt{Type}_{max(i+1,j+1)})\{\Pi o : \texttt{Type}_i.\Pi h : o \to o \to \texttt{Type}_j. \cdots\}$$

- By C-Ind:

  $$\texttt{i} \leq \texttt{k} \text{ and } \texttt{j} \leq \texttt{l} \Rightarrow \texttt{Category@}\{\texttt{i j}\} \preceq \texttt{Category@}\{\texttt{k l}\}$$

Predicative Calculus of Cumulative Inductive Types (pCuIC)

$$\textsc{C-Ind}\quad \dfrac{\begin{array}{c} I \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N}.\ s)\{\Pi\vec{x_1} : \vec{M}_1.\ X\ \vec{m}_1, \ldots, \Pi\vec{x_n} : \vec{M}_n.\ X\ \vec{m}_n\}) \\ I' \equiv (\mathsf{Ind}(X : \Pi\vec{x} : \vec{N'}.\ s')\{\Pi\vec{x_1} : \vec{M}'_1.\ X\ \vec{m}'_1, \ldots, \Pi\vec{x_n} : \vec{M}'_n.\ X\ \vec{m}'_n\}) \\ \forall i.\ N_i \preceq {N'}_i \qquad \forall i,j.\ (M_i)_j \preceq (M'_i)_j \\ length(\vec{m}) = length(\vec{x}) \qquad \forall i.\ X\ \vec{m}_i \simeq X\ \vec{m}'_i \end{array}}{I\ \vec{m} \preceq I'\ \vec{m}}$$

- Example:

$$\texttt{Category@\{i j\}} \equiv \mathsf{Ind}(X : \texttt{Type}_{max(i+1,j+1)})\{\Pi o : \texttt{Type}_i.\Pi h : o \to o \to \texttt{Type}_j. \cdots\}$$

- By C-Ind:

$$\texttt{i} \leq \texttt{k} \text{ and } \texttt{j} \leq \texttt{l} \Rightarrow \texttt{Category@\{i j\}} \preceq \texttt{Category@\{k l\}}$$

- Notice C-Ind does not consider parameters or sort of the inductive type

Back to categories

- For `Cat@{i j k l}` with exponentials we had the constraints: $\mathtt{j} = \mathtt{k} = \mathtt{l}$
- `Type_Cat@{i' j'}` we had the constraint: $j' < i'$
- Now `Type_Cat@{i' j'}` : `Obj Cat@{i j k l}` just imposes the constraint: $\mathtt{i}' \leq \mathtt{k} \wedge \mathtt{j}' \leq \mathtt{l}$, which is consistent.

Data structures and "template polymorphism"

$$\texttt{list@\{i\}}\ (\texttt{A} : \texttt{Type}_i) \equiv \mathsf{Ind}(X : \texttt{Type}_i)\{X, A \to X \to X\}$$

- By C-IND:

  $\texttt{list@\{i\} A} \preceq \texttt{list@\{j\} A}$

Data structures and "template polymorphism"

$$\texttt{list@}\{\texttt{i}\}\ (\texttt{A} : \texttt{Type}_i) \equiv \mathsf{Ind}(X : \texttt{Type}_i)\{X, A \to X \to X\}$$

- By C-Ind:

  $\texttt{list@}\{\texttt{i}\}\ \texttt{A} \preceq \texttt{list@}\{\texttt{j}\}\ \texttt{A}$ (**regardless of** `i` **and** `j`)

Data structures and "template polymorphism"

$$\texttt{list@\{i\}}\ (\texttt{A} : \texttt{Type}_i) \equiv \mathsf{Ind}(X : \texttt{Type}_i)\{X, A \to X \to X\}$$

- By C-Ind:

  $$\texttt{list@\{i\} A} \preceq \texttt{list@\{j\} A} \qquad \textbf{(regardless of } \texttt{i} \textbf{ and } \texttt{j}\textbf{)}$$

- In pCuIC we consider *fully applied* inductive types $I\ \vec{m}$ and $I'\ \vec{m}$ convertible if they are mutually subtypes

$$\textsc{Conv-Ind}\ \frac{I\ \vec{m} \preceq I'\ \vec{m} \qquad I'\ \vec{m} \preceq I\ \vec{m}}{I\ \vec{m} \simeq I'\ \vec{m}}$$

Data structures and "template polymorphism"

$$\texttt{list@\{i\}}\ (\texttt{A} : \texttt{Type}_i) \equiv \mathsf{Ind}(X : \texttt{Type}_i)\{X, A \to X \to X\}$$

- By C-IND:

  $\texttt{list@\{i\} A} \preceq \texttt{list@\{j\} A}$ (**regardless of** `i` **and** `j`)
- In pCuIC we consider *fully applied* inductive types $I\ \vec{m}$ and $I'\ \vec{m}$ convertible if they are mutually subtypes

$$\textsc{Conv-Ind}\ \frac{I\ \vec{m} \preceq I'\ \vec{m} \qquad I'\ \vec{m} \preceq I\ \vec{m}}{I\ \vec{m} \simeq I'\ \vec{m}}$$

- $\texttt{i} = \texttt{k}$ and $\texttt{j} = \texttt{l} \Rightarrow \texttt{Category@\{i j\}} \simeq \texttt{Category@\{k l\}}$
- $\texttt{list@\{i\} A} \simeq \texttt{list@\{j\} A}$ (**regardless of** `i` **and** `j`)
- $\texttt{nil@\{i\} A} \simeq \texttt{nil@\{j\} A}$ (**regardless of** `i` **and** `j` as well)
- Properly models "template-polymorphism", which allows "transparent" copies of inductives (e.g. $\texttt{list nat} : \texttt{Set}$ while $\texttt{list Type}_i : \texttt{Type}_{i+1}$).

Demo

This is implemented in Coq 8.7!

Thanks to Pierre-Marie Pédrot and Gaëtan Gilbert for many improvements on the universe system as well.

Theoretical justification

We construct a set theoretic model for pCuIC
$[\![\cdot]\!] : \mathit{Terms}_{\mathit{pCuIC}} \to \mathit{ZFC}$: ZFC with suitable axioms, e.g., inaccessible cardinals, to model pCuIC universes.

- Based on a modification of Werner & Lee's model which assumed Strong Normalization.
- For subtyping $A \preceq B$ we have $[\![A]\!] \subseteq [\![B]\!]$
- Inductive types interpreted using least fixpoints of **monotone**[2] functions, eliminators instead of match+fix+guard condition.
- This justifies both C-IND and CONV-IND and even conversion of constructors applied to parameters.

[2] Due to strict positivity condition

The End

Thanks