## First Steps Towards Cumulative Inductive Types in CIC

Amin Timany

Bart Jacobs

iMinds-Distrinet KU Leuven

ICTAC'15 - October 31, 2015

## Outline

- 1 Universe polymorphism and Inductive Types
- 2 pCIC
- 3 pCuIC
- 4 lpCuIC
- 5 Future Work Conclusion

- In higher order dependent type theories:
  - $\blacksquare$  Types are also terms and hence have a type

- In higher order dependent type theories:
  - Types are also terms and hence have a type
  - Type of all types, as it should be the type of itself, leads to paradoxes, like Russell's paradox in set theory

- In higher order dependent type theories:
  - Types are also terms and hence have a type
  - Type of all types, as it should be the type of itself, leads to paradoxes, like Russell's paradox in set theory
  - Thus, we have a countably infinite hierarchy of universes (types of types):

$$\mathsf{Type}_0, \mathsf{Type}_1, \mathsf{Type}_2, \dots$$

- In higher order dependent type theories:
  - Types are also terms and hence have a type
  - Type of all types, as it should be the type of itself, leads to paradoxes, like Russell's paradox in set theory
  - Thus, we have a countably infinite hierarchy of universes (types of types):

```
Type_0, Type_1, Type_2, \dots
```

where:

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

- In higher order dependent type theories:
  - Types are also terms and hence have a type
  - Type of all types, as it should be the type of itself, leads to paradoxes, like Russell's paradox in set theory
  - Thus, we have a countably infinite hierarchy of universes (types of types):

$$\mathsf{Type}_0, \mathsf{Type}_1, \mathsf{Type}_2, \dots$$

where:

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

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

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

■ In higher order dependent type theories:

where:

- Types are also terms and hence have a type
- Type of all types, as it should be the type of itself, leads to paradoxes, like Russell's paradox in set theory
- Thus, we have a countably infinite hierarchy of universes (types of types):

$$\label{eq:type_0} \texttt{Type}_0, \texttt{Type}_1, \texttt{Type}_2, \dots$$
 
$$\label{eq:type_0} \texttt{Type}_0 : \texttt{Type}_1, \texttt{Type}_1 : \texttt{Type}_2, \dots$$

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

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

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

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

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

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

```
 \begin{aligned} & \text{Record Category} @\{i\ j\} : \mathbf{Type} @\{\max(i+1,\ j+1)\} := \\ & \{ & \text{Obj} : \mathbf{Type} @\{i\}; \\ & \text{Hom} : \text{Obj} \to \text{Obj} \to \mathbf{Type} @\{j\}; \\ & \vdots \\ & \}. \end{aligned}
```

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

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

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

with constraints:

k < i and l < i

 $\blacksquare$  For universe polymorphic inductive types, e.g.,  ${\tt Category},$  copies are considered

- $\blacksquare$  For universe polymorphic inductive types, e.g.,  ${\tt Category},$  copies are considered
- With no cumulativity, i.e.,

- $\blacksquare$  For universe polymorphic inductive types, e.g.,  ${\tt Category},$  copies are considered
- With no cumulativity, i.e.,

 ${\tt C}: {\tt CategoryQ\{i\ j\}} \ {\rm and} \ {\tt C}: {\tt CategoryQ\{k\ l\}} \ {\rm implies} \ {\tt i} = k \ {\rm and} \ {\tt j} = l$ 

- For universe polymorphic inductive types, e.g., Category, copies are considered
- With no cumulativity, i.e.,
  - $C: Category@\{i j\} \text{ and } C: Category@\{k 1\} \text{ implies } i=k \text{ and } j=1$
- This means CatO(i j k 1) is the category of all categories at  $\{k 1\}$  and not lower

- For universe polymorphic inductive types, e.g., Category, copies are considered
- With no cumulativity, i.e.,
   C: Category@{i j} and C: Category@{k l} implies i = k and j = 1
- $\blacksquare$  This means Cat0{i j k 1} is the category of all categories at {k 1} and not lower
- Constraints on statements about universe polymorphic inductive definitions restrict to which copies they apply

- For universe polymorphic inductive types, e.g., Category, copies are considered
- With no cumulativity, i.e.,
  C: Category@{i j} and C: Category@{k l} implies i = k and j = 1
- $\blacksquare$  This means Cat0{i j k 1} is the category of all categories at {k 1} and not lower
- Constraints on statements about universe polymorphic inductive definitions restrict to which copies they apply
- For CatO(i j k 1) the fact that it has exponentials has constraints j = k = 1

- For universe polymorphic inductive types, e.g., Category, copies are considered
- With no cumulativity, i.e.,
  - C: Category@ $\{i j\}$  and C: Category@ $\{k l\}$  implies i = k and j = l
- This means Cat@{i j k 1} is the category of all categories at {k 1} and not lower
- Constraints on statements about universe polymorphic inductive definitions restrict to which copies they apply
- For Cat0(i j k 1) the fact that it has exponentials has constraints j = k = 1
- In particular:

```
Definition Type_Cat@{i j} :=
      Obj := TypeQ{j};
      Hom := fun A B \Rightarrow A \rightarrow B:
   |} : Category@{i j}.
```

with constraints: i < i

- For universe polymorphic inductive types, e.g., Category, copies are considered
- With no cumulativity, i.e.,
  - $\mathtt{C}: \texttt{CategoryQ}\{\mathtt{i}\ \mathtt{j}\} \ \mathrm{and}\ \mathtt{C}: \ \mathtt{CategoryQ}\{\mathtt{k}\ \mathtt{l}\} \ \mathrm{implies}\ \mathtt{i}=\mathtt{k} \ \ \mathrm{and} \ \ \mathtt{j}=\mathtt{l}$
- This means  $CatQ\{i j k 1\}$  is the category of all categories at  $\{k 1\}$  and not lower
- Constraints on statements about universe polymorphic inductive definitions restrict to which copies they apply
- For CatO(i j k 1) the fact that it has exponentials has constraints j = k = 1
- In particular:

```
Definition Type_Cat@{i j} := {| Obj := Type@{j}; Hom := fun A B \Rightarrow A \rightarrow B; : |} : Category@{i j}. with constraints: i < i
```

is not an object of any copy of Cat with exponentials!

- For universe polymorphic inductive types, e.g., Category, copies are considered
- With no cumulativity, i.e.,
   C: Category@{i j} and C: Category@{k l} implies i = k and j = l
- $\blacksquare$  This means Cat0{i j k 1} is the category of all categories at {k 1} and not lower
- Constraints on statements about universe polymorphic inductive definitions restrict to which copies they apply
- For CatO(i j k 1) the fact that it has exponentials has constraints j = k = 1
- In particular:

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

■ Not restricted to categories:

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:inductive EnsO(i) : TypeO(i+1) := ensO(i) : forall (A: TypeO(i)), (A \to EnsO(i)) \to EnsO(i) .}
```

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:continuous_continuous_continuous} \begin{split} & \text{Inductive EnsO(i)}: \texttt{TypeO(i+1)} := \\ & \text{ensO(i)}: \texttt{forall (A: TypeO(i))}, (\texttt{A} \rightarrow \texttt{EnsO(i)}) \rightarrow \texttt{EnsO(i)} \\ & . \end{split}
```

```
\mathtt{empty} := \mathtt{ensQ}\{0\} \; \mathtt{Empty} \; (\mathtt{Empty\_rect} \; \mathtt{EnsQ}\{\mathtt{i}\})
```

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:continuous_continuous_continuous} \begin{split} & \text{Inductive EnsO(i)}: \texttt{TypeO(i+1)} := \\ & \text{ensO(i)}: \texttt{forall (A: TypeO(i))}, (\texttt{A} \rightarrow \texttt{EnsO(i)}) \rightarrow \texttt{EnsO(i)} \\ & . \end{split}
```

```
\begin{split} & \texttt{empty} := \texttt{ensQ}\{0\} \; \texttt{Empty} \; (\texttt{Empty\_rect} \; \texttt{EnsQ}\{i\}) \\ & \texttt{union} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} + \texttt{B}) \; (\texttt{f} + \texttt{g}) \end{split}
```

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:inductive EnsO} \begin{split} &\text{Inductive EnsO(i)}: \texttt{TypeO(i+1)} := \\ &\text{ensO(i)}: \texttt{forall (A: TypeO(i))}, (\texttt{A} \rightarrow \texttt{EnsO(i)}) \rightarrow \texttt{EnsO(i)} \\ &. \end{split}
```

```
\label{eq:empty} $$ = ens@\{0\} \; Empty \; (Empty\_rect \; Ens@\{i\})$$ union $$ (ens@\{i\} \; A \; f) \; (ens@\{i\} \; B \; g) := ens@\{i\} \; (A + B) \; (f + g)$$ intersection $$ (ens@\{i\} \; A \; f) \; (ens@\{i\} \; B \; g) := ens@\{i\} \; (A \times B) \; (f \times g)$$ }
```

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:condition} \begin{split} & \textbf{Inductive EnsO(i)}: \textbf{TypeO(i+1)} := \\ & \textbf{ensO(i)}: \textbf{forall (A: TypeO(i))}, \textbf{(A} \rightarrow \textbf{EnsO(i))} \rightarrow \textbf{EnsO(i)} \\ & . \end{split}
```

```
\begin{split} & \texttt{empty} := \texttt{ens@}\{0\} \; \texttt{Empty} \; (\texttt{Empty\_rect} \; \texttt{Ens@}\{i\}) \\ & \texttt{union} \; (\texttt{ens@}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ens@}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ens@}\{i\} \; (\texttt{A} + \texttt{B}) \; (\texttt{f} + \texttt{g}) \\ & \texttt{intersection} \; (\texttt{ens@}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ens@}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ens@}\{i\} \; (\texttt{A} \times \texttt{B}) \; (\texttt{f} \times \texttt{g}) \end{split}
```

■ Ensemble of small ensembles can't be directly formed:

```
\mathtt{ensQ}\{\mathtt{j}{+}1\} \; \mathtt{EnsQ}\{\mathtt{j}\} \; (\mathtt{fun} \; \mathtt{x} : \mathtt{EnsQ}\{\mathtt{j}\} \Rightarrow \mathtt{x})
```

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:inductive EnsO(i) : TypeO(i+1) := ensO(i) : forall (A : TypeO(i)), (A \to EnsO(i)) \to EnsO(i) .}
```

```
\begin{split} & \texttt{empty} := \texttt{ensQ}\{0\} \; \texttt{Empty} \; (\texttt{Empty\_rect} \; \texttt{EnsQ}\{i\}) \\ & \texttt{union} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} + \texttt{B}) \; (\texttt{f} + \texttt{g}) \\ & \texttt{intersection} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} \times \texttt{B}) \; (\texttt{f} \times \texttt{g}) \end{split}
```

Ensemble of small ensembles can't be directly formed: ens@{j+1} Ens@{j} (fun x : Ens@{j} ⇒ x)

■ Can be solved using liftings, e.g.,

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:continuous_continuous_continuous} \begin{split} & \text{Inductive EnsO}\{i\}: \texttt{TypeO}\{i+1\} := \\ & \text{ensO}\{i\}: \texttt{forall (A: TypeO}\{i\}), (A \rightarrow \texttt{EnsO}\{i\}) \rightarrow \texttt{EnsO}\{i\} \\ . \end{split}
```

```
\begin{split} & \texttt{empty} := \texttt{ensQ}\{0\} \; \texttt{Empty} \; (\texttt{Empty\_rect} \; \texttt{EnsQ}\{i\}) \\ & \texttt{union} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} + \texttt{B}) \; (\texttt{f} + \texttt{g}) \\ & \texttt{intersection} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} \times \texttt{B}) \; (\texttt{f} \times \texttt{g}) \end{split}
```

■ Ensemble of small ensembles can't be directly formed:

$$\mathtt{ensQ}\{\mathtt{j}+1\}\;\mathtt{EnsQ}\{\mathtt{j}\}\;(\mathtt{fun}\;\mathtt{x}:\mathtt{EnsQ}\{\mathtt{j}\}\Rightarrow\mathtt{x})$$

Can be solved using liftings, e.g., ens\_lift@{i k} : Ens@{i} → Ens@{k}

with the side condition:  $i \leq k$ .

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:inductive EnsO} \begin{split} &\text{Inductive EnsO(i)}: \texttt{TypeO(i+1)}:=\\ &\text{ensO(i)}: \texttt{forall (A: TypeO(i))}, \ (\texttt{A} \to \texttt{EnsO(i)}) \to \texttt{EnsO(i)}\\ &. \end{split}
```

```
\begin{split} & \texttt{empty} := \texttt{ensQ}\{0\} \; \texttt{Empty} \; (\texttt{Empty\_rect} \; \texttt{EnsQ}\{i\}) \\ & \texttt{union} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} + \texttt{B}) \; (\texttt{f} + \texttt{g}) \\ & \texttt{intersection} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} \times \texttt{B}) \; (\texttt{f} \times \texttt{g}) \end{split}
```

Ensemble of small ensembles can't be directly formed: ens@{j+1} Ens@{j} (fun x : Ens@{j} ⇒ x)

■ Problem: e and ens\_lift e are not necessarily the same

- Not restricted to categories:
  - Consider inductive representation of ensembles:

```
\label{eq:condition} \begin{split} & \textbf{Inductive EnsO(i)}: \textbf{TypeO(i+1)} := \\ & \textbf{ensO(i)}: \textbf{forall (A: TypeO(i))}, \textbf{(A} \rightarrow \textbf{EnsO(i))} \rightarrow \textbf{EnsO(i)} \end{split}
```

```
\begin{split} & \texttt{empty} := \texttt{ensQ}\{0\} \; \texttt{Empty} \; (\texttt{Empty\_rect} \; \texttt{EnsQ}\{i\}) \\ & \texttt{union} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} + \texttt{B}) \; (\texttt{f} + \texttt{g}) \\ & \texttt{intersection} \; (\texttt{ensQ}\{i\} \; \texttt{A} \; \texttt{f}) \; (\texttt{ensQ}\{i\} \; \texttt{B} \; \texttt{g}) := \texttt{ensQ}\{i\} \; (\texttt{A} \times \texttt{B}) \; (\texttt{f} \times \texttt{g}) \end{split}
```

■ Ensemble of small ensembles can't be directly formed:

$$\mathtt{ensQ}\{\mathtt{j}+1\}\ \mathtt{EnsQ}\{\mathtt{j}\}\ (\mathtt{fun}\ \mathtt{x}:\mathtt{EnsQ}\{\mathtt{j}\}\Rightarrow\mathtt{x})$$

• Can be solved using liftings, e.g.,

```
\begin{split} & \texttt{ens\_lift@\{i\ k\}} : \texttt{Ens@\{i\}} \to \texttt{Ens@\{k\}} \\ & \text{with the side condition: } i \leq k. \end{split}
```

- Problem: e and ens\_lift e are not necessarily the same
- Any statement about e is not usable with ens\_lift e and needs to be proven separately

## Outline

- 1 Universe polymorphism and Inductive Types
- 2 pCIC
- 3 pCuIC
- 4 lpCuIC
- 5 Future Work Conclusion

$$\frac{\Gamma \vdash A : \mathtt{Type}_i \quad \Gamma, x : A \vdash B : \mathtt{Type}_j}{\Gamma \vdash \Pi x : A. \ B : \mathtt{Type}_{max(i,j)}} \quad \text{(Prod)}$$

$$\frac{\Gamma \vdash A : \mathtt{Type}_i \quad \Gamma, x : A \vdash B : \mathtt{Type}_j}{\Gamma \vdash \Pi x : A. \ B : \mathtt{Type}_{max(i,j)}} \quad \text{(Prod)}$$
 
$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash (\lambda x : A. \ t) : (\Pi x : A. \ B)} \quad \text{(Lam)}$$

$$\frac{\Gamma \vdash A : \mathsf{Type}_i \quad \Gamma, x : A \vdash B : \mathsf{Type}_j}{\Gamma \vdash \Pi x : A . \ B : \mathsf{Type}_{max(i,j)}} \quad \text{(Prod)}$$
 
$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash (\lambda x : A . \ t) : (\Pi x : A . \ B)} \quad \text{(Lam)}$$
 
$$\frac{\Gamma \vdash t : (\Pi x : A . B) \quad \Gamma \vdash t' : A}{\Gamma \vdash (t \ t') : B[t'/x]} \quad \text{(APP)}$$

■ Some (*simplified*) typing rules of pCIC:

$$\begin{split} &\frac{\Gamma \vdash A : \mathsf{Type}_i \quad \Gamma, x : A \vdash B : \mathsf{Type}_j}{\Gamma \vdash \Pi x : A. \ B : \mathsf{Type}_{max(i,j)}} & \qquad (\mathsf{PROD}) \\ &\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash (\lambda x : A. \ t) : (\Pi x : A. \ B)} & \qquad (\mathsf{LAM}) \\ &\frac{\Gamma \vdash t : (\Pi x : A.B) \quad \Gamma \vdash t' : A}{\Gamma \vdash (t \ t') : B[t'/x]} & \qquad (\mathsf{APP}) \\ &\frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s \quad A \preceq B}{\Gamma \vdash t : B} & \qquad (\mathsf{CONV}) \end{split}$$

■ Some (*simplified*) typing rules of pCIC:

$$\frac{\Gamma \vdash A : \mathsf{Type}_i \quad \Gamma, x : A \vdash B : \mathsf{Type}_j}{\Gamma \vdash \Pi x : A. \ B : \mathsf{Type}_{max(i,j)}} \quad \text{(Prod)}$$
 
$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash (\lambda x : A. \ t) : (\Pi x : A. \ B)} \quad \text{(Lam)}$$
 
$$\frac{\Gamma \vdash t : (\Pi x : A. B) \quad \Gamma \vdash t' : A}{\Gamma \vdash (t \ t') : B[t'/x]} \quad \text{(App)}$$
 
$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s \quad A \preceq B}{\Gamma \vdash t : B} \quad \text{(Conv)}$$
 
$$\frac{A \in Ar(s) \quad \Gamma \vdash A : s' \quad (\Gamma, X : A \vdash C_i : s \quad C_i \in Co(X) \ \forall 1 \le i \le n)}{\Gamma \vdash \mathsf{Ind}(X : A)\{C_1, \dots, C_n\} : A} \quad \text{(Ind)}$$
 
$$Ar(s) \quad \text{is the set of types of the form:} \quad \Pi\overrightarrow{x} : \overrightarrow{M}. \ s$$

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

A. Timany and B. Jacobs

- Examples:
  - Prod:

$$\frac{A: \texttt{Type}_i, n: nat \vdash \mathbb{Vect}_{A,n}: \texttt{Type}_i \quad A: \texttt{Type}_i \vdash nat: \texttt{Type}_0}{A: \texttt{Type}_i \vdash (\Pi n: nat. \, \mathbb{Vect}_{A,n}): \texttt{Type}_i}$$

- Examples:
  - Prod:

$$\frac{A: \texttt{Type}_i, n: nat \vdash \mathbb{V}\texttt{ect}_{A,n}: \texttt{Type}_i \quad A: \texttt{Type}_i \vdash nat: \texttt{Type}_0}{A: \texttt{Type}_i \vdash (\Pi n: nat. \, \mathbb{V}\texttt{ect}_{A,n}): \texttt{Type}_i}$$

LAM:

$$\frac{A: \texttt{Type}_i, n: nat \vdash t: \mathbb{Vect}_{A,n}}{A: \texttt{Type}_i \vdash (\lambda n: nat. \ t): (\Pi n: nat. \ \mathbb{Vect}_{A,n})}$$

- Examples:
  - Prod:

$$\frac{A: \texttt{Type}_i, n: nat \vdash \mathbb{V}\texttt{ect}_{A,n}: \texttt{Type}_i \quad A: \texttt{Type}_i \vdash nat: \texttt{Type}_0}{A: \texttt{Type}_i \vdash (\Pi n: nat. \ \mathbb{V}\texttt{ect}_{A,n}): \texttt{Type}_i}$$

LAM:

$$\frac{A: \texttt{Type}_i, n: nat \vdash t: \mathbb{V}\texttt{ect}_{A,n}}{A: \texttt{Type}_i \vdash (\lambda n: nat. \ t): (\Pi n: nat. \ \mathbb{V}\texttt{ect}_{A,n})}$$

APP:

$$\frac{A: \texttt{Type}_i \vdash f: (\Pi n: nat. \ \mathbb{Vect}_{A,n}) \quad A: \texttt{Type}_i \vdash x: nat}{A: \texttt{Type}_i \vdash f \ x: \mathbb{Vect}_{A,x}}$$

- Examples:
  - Prod:

$$\frac{A: \texttt{Type}_i, n: nat \vdash \mathbb{V}\texttt{ect}_{A,n}: \texttt{Type}_i \quad A: \texttt{Type}_i \vdash nat: \texttt{Type}_0}{A: \texttt{Type}_i \vdash (\Pi n: nat. \ \mathbb{V}\texttt{ect}_{A,n}): \texttt{Type}_i}$$

LAM:

$$\frac{A: \texttt{Type}_i, n: nat \vdash t: \mathbb{V}\texttt{ect}_{A,n}}{A: \texttt{Type}_i \vdash (\lambda n: nat. \ t): (\Pi n: nat. \ \mathbb{V}\texttt{ect}_{A,n})}$$

APP:

$$\frac{A: \texttt{Type}_i \vdash f: (\Pi n: nat. \ \mathbb{Vect}_{A,n}) \quad A: \texttt{Type}_i \vdash x: nat}{A: \texttt{Type}_i \vdash f \ x: \mathbb{Vect}_{A,x}}$$

■ Ind:

$$\cdot \vdash \mathsf{Ind}(nat : \mathsf{Type}_0) \{ nat, nat \to nat \}$$

- Examples:
  - PROD:  $A: \text{Type}_i, n: nat \vdash \text{Vect}_{A:n}: \text{Type}_i \quad A: \text{Type}_i$

$$\frac{A: \texttt{Type}_i, n: nat \vdash \texttt{Vect}_{A,n}: \texttt{Type}_i \quad A: \texttt{Type}_i \vdash nat: \texttt{Type}_0}{A: \texttt{Type}_i \vdash (\Pi n: nat. \ \texttt{Vect}_{A,n}): \texttt{Type}_i}$$

LAM:  $\frac{A: \texttt{Type}_i, n: nat \vdash t: \mathbb{V}\texttt{ect}_{A,n}}{A: \texttt{Type}_i \vdash (\lambda n: nat. \ t): (\Pi n: nat. \ \mathbb{V}\texttt{ect}_{A,n})}$ 

- APP:  $\frac{A: \texttt{Type}_i \vdash f: (\Pi n: nat. \ \mathbb{Vect}_{A,n}) \quad A: \texttt{Type}_i \vdash x: nat}{A: \texttt{Type}_i \vdash f \ x: \mathbb{Vect}_{A,n}}$
- IND:  $\cdot \vdash \mathsf{Ind}(nat : \mathsf{Type}_0)\{nat, nat \to nat\}$

$$A: \texttt{Type}_i \vdash \mathsf{Ind}(\mathit{Vect}_A: \mathit{nat} \to \mathsf{Type}_i) \{ \mathit{Vect}_A\ 0, \Pi n: \mathit{nat}.\ A \to \mathit{Vect}_A\ n \to \mathit{Vect}_A\ (S\ n) \}$$

- Examples:
  - Prod:  $A: \texttt{Type}_i, n: nat \vdash \mathbb{Vect}_{A,n}: \texttt{Type}_i \quad A: \texttt{Type}_i \vdash nat: \texttt{Type}_0$ 
    - $A: \mathbf{Type}_i \vdash (\Pi n: nat. \ \mathbb{Vect}_{A,n}): \mathbf{Type}_i$
  - LAM:  $\frac{A: \texttt{Type}_i, n: nat \vdash t: \mathbb{V}\texttt{ect}_{A,n}}{A: \texttt{Type}_i \vdash (\lambda n: nat. \ t): (\Pi n: nat. \ \mathbb{V}\texttt{ect}_{A,n})}$
  - APP:  $\frac{A: \texttt{Type}_i \vdash f: (\Pi n: nat. \ \mathbb{Vect}_{A,n}) \quad A: \texttt{Type}_i \vdash x: nat}{A: \texttt{Type}_i \vdash f \ x: \mathbb{Vect}_{A,n}}$
  - IND:  $\cdot \vdash \mathsf{Ind}(\mathit{nat} : \mathsf{Type}_0) \{ \mathit{nat}, \mathit{nat} \to \mathit{nat} \}$

$$A: \texttt{Type}_i \vdash \underbrace{\texttt{Ind}(\mathit{Vect}_A : \mathit{nat} \to \texttt{Type}_i) \{\mathit{Vect}_A \ 0, \Pi n : \mathit{nat}. \ A \to \mathit{Vect}_A \ n \to \mathit{Vect}_A \ (S \ n)\}}_{I}$$

$$Vect_{A,n} \triangleq I \ n$$

■ Conversion/cumulativity rules of pCIC:

■ Conversion/cumulativity rules of pCIC:

$$\frac{i \leq j}{\texttt{Type}_i \preceq \texttt{Type}_j} \quad \text{(C-TYPE)}$$

■ Conversion/cumulativity rules of pCIC:

$$\frac{i \le j}{\mathsf{Type}_i \le \mathsf{Type}_j} \quad \text{(C-TYPE)}$$

$$\frac{A \simeq A' \quad B \preceq B'}{\Pi x : A. \ B \preceq \Pi x : A'. \ B'} \quad \text{(C-Prod)}$$

## Outline

- 1 Universe polymorphism and Inductive Types
- 2 pCIC
- 3 pCuIC
- 4 lpCuIC
- 5 Future Work Conclusion

■ Predicative Calculus of Cumulative Inductive Types (pCuIC):

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$\begin{split} I &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq N'_i \quad \forall i,j.\ (M_i)_j \preceq (M'_i)_j\\ \underline{length(\overrightarrow{m}) = length(\overrightarrow{x})} \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ \overrightarrow{m_i'}\\ I\ \overrightarrow{m} \preceq I'\ \overrightarrow{m} \end{split} \tag{C-Ind}$$

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$I \equiv (\operatorname{Ind}(X : \overrightarrow{\Pi x} : \overrightarrow{N}. s) \{ \overrightarrow{\Pi x_1} : \overrightarrow{M_1}. X \overrightarrow{m_1}, \dots, \overrightarrow{\Pi x_n} : \overrightarrow{M_n}. X \overrightarrow{m_n} \}$$

$$I' \equiv (\operatorname{Ind}(X : \overrightarrow{\Pi x} : \overrightarrow{N'}. s') \{ \overrightarrow{\Pi x_1} : \overrightarrow{M_1}. X \overrightarrow{m_1}, \dots, \overrightarrow{\Pi x_n} : \overrightarrow{M_n}. X \overrightarrow{m_n} \}$$

$$s \leq s' \quad \forall i. \ N_i \leq N'_i \quad \forall i. j. \ (M_i)_j \leq (M'_i)_j$$

$$ength(\overrightarrow{m}) = ength(\overrightarrow{x}) \quad \forall i. X \overrightarrow{m_i} \simeq X \overrightarrow{m_i'}$$

$$I \overrightarrow{m} \leq I' \overrightarrow{m}$$

$$(C-\operatorname{IND})$$

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$I \equiv (\operatorname{Ind}(X : \overrightarrow{\Pi x} : \overrightarrow{N}. s) \{ \overrightarrow{\Pi x_1} : \overrightarrow{M_1}. X \overrightarrow{m_1}, \dots, \overrightarrow{\Pi x_n} : \overrightarrow{M_n}. X \overrightarrow{m_n} \}$$

$$I' \equiv (\operatorname{Ind}(X : \overrightarrow{\Pi x} : \overrightarrow{N'}. s') \{ \overrightarrow{\Pi x_1} : \overrightarrow{M'_1}. X \overrightarrow{m'_1}, \dots, \overrightarrow{\Pi x_n} : \overrightarrow{M'_n}. X \overrightarrow{m'_n} \}$$

$$s \leq s' \quad \forall i. \ N_i \leq N'_i \quad \forall i. j. \ (M_i)_j \leq (M'_i)_j$$

$$length(\overrightarrow{m}) = length(\overrightarrow{x}) \quad \forall i. \ X \ \overrightarrow{m_i} \simeq X \ \overrightarrow{m'_i}$$

$$I \ \overrightarrow{m} \leq I' \ \overrightarrow{m}$$

$$(C-IND)$$

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$I \equiv (\operatorname{Ind}(X : \Pi \overrightarrow{x} : \overrightarrow{N}. s) \{ \Pi \overrightarrow{x_1} : \overrightarrow{M_1}. X \overrightarrow{m_1}, \dots, \Pi \overrightarrow{x_n} : \overrightarrow{M_n}. X \overrightarrow{m_n} \}$$

$$I' \equiv (\operatorname{Ind}(X : \Pi \overrightarrow{x} : \overrightarrow{N'}. s') \{ \Pi \overrightarrow{x_1} : \overrightarrow{M_1}. X \overrightarrow{m_1}, \dots, \Pi \overrightarrow{x_n} : \overrightarrow{M_n}. X \overrightarrow{m_n} \}$$

$$s \preceq s' \quad \forall i. \ N_i \preceq N'_i \quad \forall i, j. \ (M_i)_j \preceq (M'_i)_j$$

$$length(\overrightarrow{m}) = length(\overrightarrow{x}) \quad \forall i. X \overrightarrow{m_i} \simeq X \overrightarrow{m_i'}$$

$$I \overrightarrow{m} \preceq I' \overrightarrow{m} \qquad (C-\operatorname{IND})$$

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$I \equiv (\operatorname{Ind}(X : \Pi \overrightarrow{x} : \overrightarrow{N}. s) \{ \Pi \overrightarrow{x_1} : \overrightarrow{M_1}. X \overrightarrow{m_1}, \dots, \Pi \overrightarrow{x_n} : \overrightarrow{M_n}. X \overrightarrow{m_n} \}$$

$$I' \equiv (\operatorname{Ind}(X : \Pi \overrightarrow{x} : \overrightarrow{N'}. s') \{ \Pi \overrightarrow{x_1} : \overrightarrow{M'_1}. X \overrightarrow{m'_1}, \dots, \Pi \overrightarrow{x_n} : \overrightarrow{M'_n}. X \overrightarrow{m'_n} \}$$

$$s \preceq s' \quad \forall i. \ N_i \preceq N'_i \quad \forall i. j. \ (M_i)_j \preceq (M'_i)_j$$

$$length(\overrightarrow{m}) = length(\overrightarrow{x}) \quad \forall i. X \overrightarrow{m_i} \simeq X \overrightarrow{m'_i}$$

$$I \overrightarrow{m} \preceq I' \overrightarrow{m}$$

$$(C-\operatorname{IND})$$

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$\begin{split} I &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq N'_i \quad \forall i,j.\ (M_i)_j \preceq (M'_i)_j\\ & \underline{length(\overrightarrow{m}) = length(\overrightarrow{x})} \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ \overrightarrow{m_i'}\\ & \underline{I\ \overrightarrow{m} \preceq I'\ \overrightarrow{m}} \end{split} \tag{C-Ind}$$

$$\texttt{Category@\{i\ j\}} \equiv \mathsf{Ind}(X: \texttt{Type}_{max(i+1,j+1)}) \{\Pi o: \texttt{Type}_i.\Pi h: o \rightarrow o \rightarrow \texttt{Type}_j.N \}$$
 where  $i$  and  $j$  don't appear in term  $N$ 

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$\begin{split} I &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1'}.\ X\ \overrightarrow{m_1'},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n'}.\ X\ \overrightarrow{m_n'}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq N'_i \quad \forall i,j.\ (M_i)_j \preceq (M'_i)_j\\ & \underline{length(\overrightarrow{m}) = length(\overrightarrow{x})} \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ \overrightarrow{m_i'}\\ & \underline{I\ \overrightarrow{m} \preceq I'\ \overrightarrow{m}} \end{split} \tag{C-Ind}$$

$$\texttt{Category@\{i\ j\}} \equiv \texttt{Ind}(X: \texttt{Type}_{max(i+1,j+1)}) \{\Pi o: \texttt{Type}_{i}.\Pi h: o \rightarrow o \rightarrow \texttt{Type}_{j}.N \}$$
 where  $i$  and  $j$  don't appear in term  $N$ 

■ By C-Ind:

$$\texttt{Type}_i \preceq \texttt{Type}_k \text{ and } \texttt{Type}_j \preceq \texttt{Type}_l \Rightarrow \texttt{CategoryQ}\{\texttt{i} \texttt{ j}\} \preceq \texttt{CategoryQ}\{\texttt{k} \texttt{ l}\}$$

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$\begin{split} I &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq N'_i \quad \forall i,j.\ (M_i)_j \preceq (M'_i)_j\\ \underline{length(\overrightarrow{m}) = length(\overrightarrow{x})} \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ \overrightarrow{m_i'}\\ I\ \overrightarrow{m} \preceq I'\ \overrightarrow{m} \end{split} \tag{C-Ind}$$

$$\texttt{Category@\{i\ j\}} \equiv \mathsf{Ind}(X: \texttt{Type}_{max(i+1,j+1)}) \{ \Pi o: \texttt{Type}_i. \Pi h: o \to o \to \texttt{Type}_j. N \}$$
 where  $i$  and  $j$  don't appear in term  $N$ 

■ By C-Ind:

$$\mathsf{Type}_i \preceq \mathsf{Type}_k \text{ and } \mathsf{Type}_j \preceq \mathsf{Type}_l \Rightarrow \mathsf{Category} \mathsf{Q} \{ \mathsf{i} \ \mathsf{j} \} \preceq \mathsf{Category} \mathsf{Q} \{ \mathsf{k} \ \mathsf{1} \}$$

Also:

$$\mathtt{EnsQ}\{\mathtt{i}\} \equiv \mathsf{Ind}(X : \mathtt{Type}_{i+1})\{\Pi A : \mathtt{Type}_i.(A \to X) \to X\}$$

- Predicative Calculus of Cumulative Inductive Types (pCuIC):
- pCuIC is pCIC + C-IND rule:

$$\begin{split} I &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq N'_i \quad \forall i,j.\ (M_i)_j \preceq (M'_i)_j\\ \underline{length(\overrightarrow{m}) = length(\overrightarrow{x})} \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ \overrightarrow{m_i'}\\ I\ \overrightarrow{m} \preceq I'\ \overrightarrow{m} \end{split} \tag{C-Ind}$$

$$\texttt{Category@\{i\ j\}} \equiv \texttt{Ind}(X: \texttt{Type}_{max(i+1,j+1)}) \{\Pi o: \texttt{Type}_{i}.\Pi h: o \rightarrow o \rightarrow \texttt{Type}_{j}.N \}$$
 where  $i$  and  $j$  don't appear in term  $N$ 

■ By C-Ind:

$$Type_i \leq Type_k \text{ and } Type_j \leq Type_l \Rightarrow Category@\{i j\} \leq Category@\{k 1\}$$

Also:

$$\begin{split} \operatorname{EnsQ}\{\mathbf{i}\} & \equiv \operatorname{Ind}(X: \operatorname{Type}_{i+1})\{\Pi A: \operatorname{Type}_i.(A \to X) \to X\} \\ & \operatorname{Type}_i \preceq \operatorname{Type}_k \Rightarrow \operatorname{EnsQ}\{\mathbf{i}\} \preceq \operatorname{EnsQ}\{\mathbf{k}\} \end{split}$$

# ${\bf Conjecture}$

## pCuIC has the following properties:

- Church-Rosser property
- 2 Strong normalization
- 3 Context Validity
- 4 Typing Validity
- 5 Subject Reduction

#### Conjecture

pCuIC has the following properties:

- Church-Rosser property
- 2 Strong normalization
- 3 Context Validity
- 4 Typing Validity
- 5 Subject Reduction

#### Conjecture

Let  $\Gamma \vdash_{\mathsf{pCIC}} T : s$  be a pCIC type such that  $\Gamma \vdash_{\mathsf{pCulC}} t : T$ . Then there exists a term t' such that  $\Gamma \vdash_{\mathsf{pCIC}} t' : T$ .

#### Conjecture

pCuIC has the following properties:

- Church-Rosser property
- 2 Strong normalization
- 3 Context Validity
- 4 Typing Validity
- 5 Subject Reduction

## Conjecture

Let  $\Gamma \vdash_{\mathsf{pCIIC}} T : s$  be a pCIC type such that  $\Gamma \vdash_{\mathsf{pCuIC}} t : T$ . Then there exists a term t' such that  $\Gamma \vdash_{\mathsf{pCIIC}} t' : T$ .

■ The latter reduces the soundness of pCuIC to the soundness of pCIC:

$$\cdot \dashv_{\mathsf{pCuIC}} t : False \Rightarrow \exists t'. \ \cdot \dashv_{\mathsf{pCIC}} t' : False$$

#### Conjecture

pCuIC has the following properties:

- Church-Rosser property
- 2 Strong normalization
- 3 Context Validity
- 4 Typing Validity
- 5 Subject Reduction

#### Conjecture

Let  $\Gamma \vdash_{\mathsf{pCIIC}} T : s$  be a pCIC type such that  $\Gamma \vdash_{\mathsf{pCuIC}} t : T$ . Then there exists a term t' such that  $\Gamma \vdash_{\mathsf{pCIC}} t' : T$ .

■ The latter reduces the soundness of pCuIC to the soundness of pCIC:

$$\cdot \dashv_{\mathsf{pCuIC}} t : False \Rightarrow \exists t'. \cdot \dashv_{\mathsf{pCIC}} t' : False$$

■ We prove this conjecture for the lesser pCuIC (lpCuIC), a fragment of pCuIC

# Outline

- 1 Universe polymorphism and Inductive Types
- 2 pCIC
- 3 pCuIC
- 4 lpCuIC
- 5 Future Work Conclusion

■ The lesser pCuIC (lpCuIC) is a fragment of pCuIC:

- $\blacksquare$  The lesser pCuIC (lpCuIC) is a fragment of pCuIC:
- $\blacksquare$  For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : s$

- The lesser pCuIC (lpCuIC) is a fragment of pCuIC:
- $\blacksquare$  For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : s$
- $\blacksquare \text{ For any sub-derivation } \Gamma \vdash_{\mathsf{lpCulC}} T : \overrightarrow{\Pi x} : \overset{\rightarrow}{A}.s \text{ we have } \Gamma \vdash_{\mathsf{pCIC}} T : \overrightarrow{\Pi x} : \overset{\rightarrow}{A}.s$

- The lesser pCuIC (lpCuIC) is a fragment of pCuIC:
- For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : s$
- $\blacksquare \text{ For any sub-derivation } \Gamma \vdash_{\mathsf{lpCulC}} T : \overrightarrow{\Pi x} : \overrightarrow{A}.s \text{ we have } \Gamma \vdash_{\mathsf{pCIC}} T : \overrightarrow{\Pi x} : \overrightarrow{A}.s$
- C-Ind is replaced by C-Ind'

$$\begin{split} I &\equiv (\mathsf{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi(\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\} \\ I' &\equiv (\mathsf{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\} \\ s &\preceq s' \quad \forall i.\ N_i \preceq_{\mathsf{PCIC}} N'_i \quad \forall i,j.\ (M_i)_j \preceq_{\mathsf{PCIC}} (M'_i)_j \\ &\qquad \qquad \underbrace{length(\overrightarrow{m}) = length(\overrightarrow{x})}_{I\ \overrightarrow{m}} \ \forall i.\ X\ \overrightarrow{m_i} \simeq X\ \overrightarrow{m_i'}}_{I\ \overrightarrow{m}} \end{split} \tag{C-Ind'}$$

- The lesser pCuIC (lpCuIC) is a fragment of pCuIC:
- For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : s$
- For any sub-derivation  $\Gamma \vdash_{\mathsf{lpCulC}} T : \Pi \overrightarrow{x} : \overrightarrow{A}.s$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : \Pi \overrightarrow{x} : \overrightarrow{A}.s$
- C-Ind is replaced by C-Ind'

$$\begin{split} I &\equiv (\mathsf{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi(\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\mathsf{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M'_1}.\ X\ \overrightarrow{m'_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M'_n}.\ X\ \overrightarrow{m'_n}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq_{\text{pCIC}} N'_i \quad \forall i,j.\ (M_i)_j \preceq_{\text{pCIC}} (M'_i)_j \\ &\qquad \qquad \qquad \underbrace{length(\overrightarrow{m}) = length(\overrightarrow{x}) \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ m'_i}_{I\ \overrightarrow{m} \preceq I'\ \overrightarrow{m}} \end{split} \tag{C-Ind'}$$

- The lesser pCuIC (lpCuIC) is a fragment of pCuIC:
- For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : s$
- For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCulC}} T : \Pi \overrightarrow{x} : \overrightarrow{A}.s$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : \Pi \overrightarrow{x} : \overrightarrow{A}.s$
- C-Ind is replaced by C-Ind'

$$\begin{split} I &\equiv (\mathsf{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi(\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\mathsf{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq_{\mathsf{pCIC}} N'_i \quad \forall i,j.\ (M_i)_j \preceq_{\mathsf{pCIC}} (M'_i)_j\\ &\qquad \qquad \qquad \underbrace{length(\overrightarrow{m}) = length(\overrightarrow{x}) \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ m'_i}_{I\ \overrightarrow{m}} \quad (\text{C-Ind}') \end{split}$$

App is replaced by App'

$$(\mathsf{APP'}) \ \frac{\Gamma \vdash t : (\Pi x : A.\ B) \quad \Gamma \vdash t' : A}{\Gamma \vdash (t\ t') : B[t'/x]} \quad (\Gamma \vdash_{\mathsf{pCIC}} t' : A \text{ or } x \not\in FV(B))$$

- The lesser pCuIC (lpCuIC) is a fragment of pCuIC:
- $\blacksquare$  For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : s$
- For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCulC}} T : \Pi \overrightarrow{x} : \overrightarrow{A}.s$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : \Pi \overrightarrow{x} : \overrightarrow{A}.s$
- C-Ind is replaced by C-Ind'

$$\begin{split} I &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi(\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq \underset{\mathsf{pCIC}}{\operatorname{pCIC}} N'_i \quad \forall i,j.\ (M_i)_j \preceq \underset{\mathsf{pCIC}}{\operatorname{pCIC}} (M'_i)_j \\ &\qquad \qquad \qquad \underbrace{length(\overrightarrow{m}) = length(\overrightarrow{x}) \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ m'_i}_{I\ \overrightarrow{m}} \end{split} \tag{C-Ind'}$$

App is replaced by App'

$$(APP') \frac{\Gamma \vdash t : (\Pi x : A. B) \quad \Gamma \vdash t' : A}{\Gamma \vdash (t \ t') : B[t'/x]} \quad (\Gamma \vdash_{\mathsf{pCIC}} t' : A \text{ or } x \not\in FV(B))$$

In lpCuIC,

$$\mathsf{Type}_i \leq \mathsf{Type}_k \text{ and } \mathsf{Type}_j \leq \mathsf{Type}_l \Rightarrow \mathsf{Category} \emptyset \{ i j \} \leq \mathsf{Category} \emptyset \{ k 1 \}$$

- The lesser pCuIC (lpCuIC) is a fragment of pCuIC:
- $\blacksquare$  For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : s$
- For any sub-derivation  $\Gamma \vdash_{\mathsf{IpCulC}} T : \Pi \overrightarrow{x} : \overrightarrow{A}.s$  we have  $\Gamma \vdash_{\mathsf{pCIC}} T : \Pi \overrightarrow{x} : \overrightarrow{A}.s$
- C-Ind is replaced by C-Ind'

$$\begin{split} I &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N}.\ s)\{\Pi(\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ I' &\equiv (\operatorname{Ind}(X:\Pi\overrightarrow{x}:\overrightarrow{N'}.\ s')\{\Pi\overrightarrow{x_1}:\overrightarrow{M_1}.\ X\ \overrightarrow{m_1},\dots,\Pi\overrightarrow{x_n}:\overrightarrow{M_n}.\ X\ \overrightarrow{m_n}\}\\ s &\preceq s' \quad \forall i.\ N_i \preceq_{\text{pCIC}} N'_i \quad \forall i,j.\ (M_i)_j \preceq_{\text{pCIC}} (M'_i)_j \\ &\qquad \qquad \qquad \\ &\qquad \qquad \qquad \underbrace{length(\overrightarrow{m}) = length(\overrightarrow{x}) \quad \forall i.\ X\ \overrightarrow{m_i} \simeq X\ \overrightarrow{m_i'}}_{I\ \overrightarrow{m}} \ (\text{C-Ind'}) \end{split}$$

App is replaced by App'

$$(\mathsf{APP'}) \ \frac{\Gamma \vdash t : (\Pi x : A.\ B) \quad \Gamma \vdash t' : A}{\Gamma \vdash (t\ t') : B[t'/x]} \quad (\Gamma \vdash_{\mathsf{pCIC}} t' : A \text{ or } x \not\in FV(B))$$

In lpCuIC,

$$\mathsf{Type}_i \preceq \mathsf{Type}_k \text{ and } \mathsf{Type}_j \preceq \mathsf{Type}_l \Rightarrow \mathsf{CategoryQ}\{\mathsf{i} \mathsf{j}\} \preceq \mathsf{CategoryQ}\{\mathsf{k} \mathsf{l}\}$$

Also:

$$Type_i \leq Type_k \Rightarrow Ens0\{i\} \leq Ens0\{k\}$$

■ We prove soundness of lpCuIC:

# Theorem (Inhabitants in lpCuIC)

Let t and T be terms such that  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$ . Then there exists t' such that  $\Gamma \vdash_{\mathsf{pCIC}} t' : T$ .

■ We prove soundness of lpCuIC:

## Theorem (Inhabitants in lpCuIC)

Let t and T be terms such that  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$ . Then there exists t' such that  $\Gamma \vdash_{\mathsf{pCIC}} t' : T$ .

## Proof sketch.

We build lifters  $\Gamma \dashv_{\mathsf{pCIC}} \Upsilon_{T \preceq_{\mathsf{lpCulC}} T'} : T \to T'$  for  $T \preceq_{\mathsf{lpCulC}} T'$ . Each sub-term t : T for which we have used Conv to derive t : T' is replaced with  $(\Upsilon_{T \preceq_{\mathsf{lpCulC}} T'} \ t)$ .

■ We prove soundness of lpCuIC:

## Theorem (Inhabitants in lpCuIC)

Let t and T be terms such that  $\Gamma \vdash_{\mathsf{IpCuIC}} t : T$ . Then there exists t' such that  $\Gamma \vdash_{\mathsf{pCIC}} t' : T$ .

## Proof sketch.

We build lifters  $\Gamma \dashv_{\mathsf{pCIC}} \Upsilon_{T \preceq_{\mathsf{lpCulC}} T'} : T \to T'$  for  $T \preceq_{\mathsf{lpCulC}} T'$ . Each sub-term t : T for which we have used Conv to derive t : T' is replaced with  $(\Upsilon_{T \preceq_{\mathsf{lpCulC}} T'} \ t)$ .

# Corollary (Soundness of lpCuIC)

 $\cdot \vdash_{\mathsf{lpCulC}} t : False \ implies \ that \ there \ exists \ t' \ such \ that \ \cdot \vdash_{\mathsf{pClC}} t' : False.$ 

- Future work:
  - $\blacksquare$  Proof of conjectures about pCuIC

- Future work:
  - Proof of conjectures about pCuIC
  - Implementation

- Proof of conjectures about pCuIC
- Implementation
- Considering parameters of inductive types:

```
\begin{split} & \textbf{Inductive List0}\{i\} \ (A: \begin{tabular}{ll} \textbf{Type0}\{i\}) := \\ & | \begin{tabular}{ll} Ni1 : List0\{i\} \ A \\ & | \begin{tabular}{ll} Cons : A $\rightarrow$ List0\{i\} \ A $\rightarrow$ List0\{i\} \ A \\ \end{split}
```

- Proof of conjectures about pCuIC
- Implementation
- Considering parameters of inductive types:

```
\begin{array}{l} \textbf{Inductive List0}\{i\} \; (A: \begin{tabular}{l} \textbf{Type0}\{i\}) := \\ | \begin{tabular}{l} | \begin{tabular}{l} \textbf{Nil} : List0\{i\} \; A \\ | \begin{tabular}{l} | \begin{tabular}{l} \textbf{Cons} : \; A \to List0\{i\} \; A \\ \end{tabular} \end{array}
```

We can have ListQ{i} A  $\leq$  ListQ{i'} B when A  $\leq$  B.

- Proof of conjectures about pCuIC
- Implementation
- Considering parameters of inductive types:

```
\begin{split} & \text{Inductive ListQ[i] (A: TypeQ[i])} := \\ & | \text{Nil: ListQ[i] A} \\ & | \text{Cons: A} \rightarrow \text{ListQ[i] A} \rightarrow \text{ListQ[i] A} \\ & \cdot \\ & \text{We can have ListQ[i] A} \preceq \text{ListQ[i'] B when A} \preceq \text{B.} \\ & \text{Inductive Img_inhQ[i j] (F: TypeQ[i] } \rightarrow \text{TypeQ[j]) (A: TypeQ[i])} := \\ & \text{fa: (F A)} \rightarrow \text{Img_inh F A} \\ & \cdot \\ \end{split}
```

- Proof of conjectures about pCuIC
- Implementation
- Considering parameters of inductive types:

```
\begin{array}{l} \textbf{Inductive List0}\{i\} \; (A: \begin{tabular}{l} \textbf{Type0}\{i\}) := \\ | \begin{tabular}{l} | \begin{tabular}{l} \textbf{Ni1} : List0\{i\} \; A \\ | \begin{tabular}{l} | \begin{tabular}{l} \textbf{Cons} : \; A \to List0\{i\} \; A \\ \end{tabular} \end{array}
```

We can have ListQ{i} A  $\leq$  ListQ{i'} B when A  $\leq$  B.

```
\label{eq:continuity} \begin{array}{l} \textbf{Inductive Img\_inh@\{i\ j\}\ (F:Type@\{i\}) \to Type@\{j\})\ (A:Type@\{i\}) := fa:\ (F\ A)\ \to Img\_inh\ F\ A } \end{array}
```

Whether  $(Img_inhQ\{i\ j\}\ F\ A) \leq (Img_inhQ\{i'\ j'\}\ F\ B)$  when  $A \leq B$  depends on variance of F.

- Proof of conjectures about pCuIC
- Implementation
- Considering parameters of inductive types:

```
\begin{array}{l} \textbf{Inductive List@\{i\} (A: Type@\{i\}) :=} \\ | \texttt{Nil} : \texttt{List@\{i\} A} \\ | \texttt{Cons} : \texttt{A} \rightarrow \texttt{List@\{i\} A} \rightarrow \texttt{List@\{i\} A} \\ \end{array}
```

We can have  $ListQ\{i\}$  A  $\leq ListQ\{i'\}$  B when A  $\leq$  B.

```
\label{eq:continuous_index} \begin{split} & \textbf{Inductive } \ \textbf{Img\_inh0} \\ & \textbf{i} \ \textbf{j} \ \textbf{j} \ \textbf{fi} : \ \textbf{Type0} \\ & \textbf{i} \ \textbf{j} \\ & \textbf{inductive } \ \textbf{Img\_inh } \ \textbf{F} \ \textbf{A} \\ & \textbf{j} \end{split}
```

Whether (Img\_inh0{i j} F A)  $\leq$  (Img\_inh0{i' j'} F B) when A  $\leq$  B depends on variance of F.

- Conclusion:
  - Presented pCuIC

- Proof of conjectures about pCuIC
- Implementation
- Considering parameters of inductive types:

```
\begin{split} & \textbf{Inductive List0}\{i\} \; (A: \  \, \textbf{Type0}\{i\}) := \\ & | \  \, \textbf{Ni1} : \  \, \textbf{List0}\{i\} \; \textbf{A} \\ & | \  \, \textbf{Cons} : \; \textbf{A} \to \textbf{List0}\{i\} \; \textbf{A} \to \textbf{List0}\{i\} \; \textbf{A} \end{split}
```

We can have ListQ{i} A  $\leq$  ListQ{i'} B when A  $\leq$  B.

```
\label{eq:continuous} \begin{split} &\text{Inductive Img\_inh0\{i\ j\}\ (F: Type0\{i\} \to Type0\{j\})\ (A: Type0\{i\}) := \\ &\text{fa: } (F\ A) \to Img\_inh\ F\ A \end{split}
```

Whether (Img\_inh0{i j} F A)  $\leq$  (Img\_inh0{i' j'} F B) when A  $\leq$  B depends on variance of F.

#### ■ Conclusion:

- Presented pCuIC
- Discussed how it makes working with structures such as categories and ensembles easier

- Proof of conjectures about pCuIC
- Implementation
- Considering parameters of inductive types:

```
\begin{array}{l} \textbf{Inductive ListQ\{i\} (A: TypeQ\{i\}) :=} \\ | \texttt{Nil} : \texttt{ListQ\{i\} A} \\ | \texttt{Cons} : \texttt{A} \rightarrow \texttt{ListQ\{i\} A} \rightarrow \texttt{ListQ\{i\} A} \\ . \end{array}
```

We can have  $ListQ\{i\}$  A  $\leq ListQ\{i'\}$  B when A  $\leq$  B.

```
\label{eq:continuous_inductive_Img_inh0} $$\operatorname{Inductive\ Img\_inh0(i\ j)}$ (F: Type0(i) \to Type0(j)) (A: Type0(i)) := fa: (FA) \to Img\_inh\ FA
```

Whether (Img\_inh0{i j} F A)  $\leq$  (Img\_inh0{i' j'} F B) when A  $\leq$  B depends on variance of F.

#### Conclusion:

- Presented pCuIC
- Discussed how it makes working with structures such as categories and ensembles easier
- Presented lpCuIC

- Proof of conjectures about pCuIC
- Implementation
- Considering parameters of inductive types:

```
\begin{array}{l} \textbf{Inductive ListQ\{i\} (A: TypeQ\{i\}) :=} \\ | \texttt{Nil} : \texttt{ListQ\{i\} A} \\ | \texttt{Cons} : \texttt{A} \rightarrow \texttt{ListQ\{i\} A} \rightarrow \texttt{ListQ\{i\} A} \\ . \end{array}
```

We can have ListQ{i} A  $\leq$  ListQ{i'} B when A  $\leq$  B.

```
\label{eq:continuous} \begin{split} &\text{Inductive Img\_inh0\{i\ j\}\ (F: Type0\{i\} \to Type0\{j\})\ (A: Type0\{i\}) := \\ &\text{fa: } (F\ A) \to Img\_inh\ F\ A \end{split}
```

Whether (Img\_inh0{i j} F A)  $\leq$  (Img\_inh0{i' j'} F B) when A  $\leq$  B depends on variance of F.

#### ■ Conclusion:

- Presented pCuIC
- Discussed how it makes working with structures such as categories and ensembles easier
- Presented lpCuIC
  - As an intuitive reason why we believe pCuIC is sound