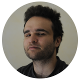
Ordered Models of CIC

Kenji Maillard

Inria Nantes, team Gallinette
j.w.w.

Meven Lennon-Bertrand | Nicolas Tabareau | Éric Tanter | Théo Laurent

GdT LHC
Friday the 5th of February, 2021

What is CIC ?

CIC: logic & programs

CIC : Calculus of (Co)Inductive Constructions
A rich logical system & an expressive programming language

- Inductive and coinductive types with pattern-matching,
- functions $(a : A) \to B$, (well-founded) fixpoints,
- **Dependent types**

$$\vdash A \text{ type} \qquad x : A \vdash B \text{ type} \qquad n : \mathbb{N} \vdash \texttt{vect}\ A\ n \text{ type}$$

CIC: logic & programs

CIC : Calculus of (Co)Inductive Constructions
A rich logical system & an expressive programming language

- Inductive and coinductive types with pattern-matching,
- functions $(a : A) \to B$, (well-founded) fixpoints,
- **Dependent types** internalized with Universes $\mathbb{U}_i$

$$\frac{\vdash A : \mathbb{U}_i}{\vdash A \text{ type}} \qquad \vdash \mathtt{vect}\, A \;:\; (A : \mathbb{U}_i)(n : \mathbb{N}) \to \mathbb{U}_i \qquad \vdash \mathbb{U}_i : \mathbb{U}_{i+1}$$

CIC: logic & programs

CIC : Calculus of (Co)Inductive Constructions
A rich logical system & an expressive programming language

- Inductive and coinductive types with pattern-matching,
- functions $(a : A) \to B$, (well-founded) fixpoints,
- **Dependent types** internalized with Universes $\mathbb{U}_i$

$$\frac{\vdash A : \mathbb{U}_i}{\vdash A \text{ type}} \qquad \vdash \mathtt{vect}\, A \ : (A : \mathbb{U}_i)(n : \mathbb{N}) \to \mathbb{U}_i \qquad \vdash \mathbb{U}_i : \mathbb{U}_{i+1}$$

Idealized metatheory of various proofs assistants:

Computations

Conversion

$$\frac{\vdash 1+2 \equiv 3 : \mathbb{N}}{\vdash \texttt{vect}\ A\ (1+2) \equiv \texttt{vect}\ A\ 3} \qquad \frac{\vdash t : A \qquad \vdash A \equiv B}{\vdash t : B}$$

Trade-offs betweeen *decidability* and *expressivity*

$$\boxed{\text{Weak TT}} \Longleftarrow \boxed{\text{CIC}} \Longrightarrow \boxed{\text{Extensional TT}}$$

Trivial conversion $\qquad \beta\delta\iota\zeta\eta \qquad$ Provable equality

Checking proofs vs Writing proofs

Models of CIC For Fun And Profit

Motivation

Add new proof principles:

- Uniqueness of identity proofs (`UIP`)
- Function extensionality (`funext`)
- Quotients
- Univalence principle
- Markov principle
- Parametricity

Account for existing programming features:

- Exceptions
- Access to a global environment
- Subtyping
- **Dynamic type**

Syntactic Models

Models of CIC in CIC:

- Defined inductively on the syntax of terms/types

$$[\![-]\!] : \mathit{Type} \to \mathit{Type} \qquad\qquad [-] : \mathit{Term} \to \mathit{Term}$$

- Preserving conversion (no coherence hell)

$$\Gamma \vdash A \equiv B \qquad \Longrightarrow \qquad [\![\Gamma]\!] \vdash [\![A]\!] \equiv [\![B]\!]$$

Main goal/theorem:

$$\Gamma \vdash t : A \qquad \Longrightarrow \qquad [\![\Gamma]\!] \vdash [t] : [\![A]\!]$$

Syntactic Models

Models of CIC in CIC:

- Defined inductively on the syntax of terms/types

$$[\![-]\!] : \mathit{Type} \to \mathit{Type} \qquad\qquad [-] : \mathit{Term} \to \mathit{Term}$$

- Preserving conversion (no coherence hell)

$$\Gamma \vdash A \equiv B \qquad\qquad \Longrightarrow \qquad\qquad [\![\Gamma]\!] \vdash [\![A]\!] \equiv [\![B]\!]$$

Why syntactic models ?

- Useful to prototype extensions of CIC
- Proposes extensions more amenable to implementations
- Help designing reductions/conversion rules

Examples from the literature

Reflexive graphs model: external parametricity [Atkey et al.]
  Types equipped with a reflexive relation

Setoid model: `UIP`, `funext` [Altenkirch et al.]
  Types equipped with an irrelevant equivalence relation

Exceptional model: Exceptions [Pédrot et al.]
  Pointed types

Reader model: Reading and setting a global cell [Boulier et al.]
  Presheaves on a set of states

Syntactic Models: A Recipe

$$\mathrm{CIC} \xrightarrow{\quad\quad\quad [\![-]\!] \quad\quad\quad} \mathrm{CIC}$$

Crucial steps

1. Give the structure of types, type families and terms

Syntactic Models: A Recipe

$$\mathrm{CIC} \xrightarrow{[\![-]\!]} \mathrm{CIC}$$

Crucial steps

1. Give the structure of types, type families and terms
2. Translate type constructors $(\mathbb{N}, \Pi)$ & universes $[\mathbb{U}_i] : [\![\mathbb{U}_{i+1}]\!]$

Syntactic Models: A Recipe

$$\mathrm{CIC} \xrightarrow{\quad [\![-]\!] \quad} \mathrm{CIC}$$

Crucial steps

1. Give the structure of types, type families and terms
2. Translate type constructors $(\mathbb{N}, \Pi)$ & universes $[\mathbb{U}_i] : [\![\mathbb{U}_{i+1}]\!]$
3. Check that conversion is preserved $(\beta\delta\iota\zeta\eta\ldots)$

Syntactic Models: A Recipe

$$
\begin{tikzcd}
\mathrm{CIC} \arrow[rr, "[\![-]\!]"] \arrow[dr, bend right, "\textit{embeds}"'] & & \mathrm{CIC} \\
& \mathcal{T} \arrow[ur, "[\![-]\!]'"'] &
\end{tikzcd}
$$

Crucial steps

1. Give the structure of types, type families and terms
2. Translate type constructors $(\mathbb{N}, \Pi)$ & universes $[\mathbb{U}_i] : [\![\mathbb{U}_{i+1}]\!]$
3. Check that conversion is preserved $(\beta\delta\iota\zeta\eta\ldots)$
4. Extend the source CIC to a richer theory $\mathcal{T}$
   adding new constants and conversion rules

Ordered models of CIC

Types as (Pre)Orders

Step 1: Equip the translation of a type A with a relation

$$\leq^A \quad : \quad A \to A \to Type$$

$$\begin{aligned}
\texttt{reflexive} &: (a : A) \to a \leq^A a \\
\texttt{transitive} &: (a_0\, a_1\, a_2 : A) \to a_0 \leq^A a_1 \to a_1 \leq^A a_2 \to a_0 \leq^A a_2 \\
\texttt{irrelevant} &: (a_0\, a_1 : A)(h\, h' : a_0 \leq^A a_1) \to h = h' \\
\texttt{antisymmetric} &: (a_0\, a_1 : A) \to a_0 \leq^A a_1 \to a_1 \leq^A a_0 \to a_0 = a_1
\end{aligned}$$

Middle point between the reflexive graph and setoid models.

Type families ?

Translation of a type family $x : A \vdash B$ type

$$\begin{array}{rcccc} B & : & A & \to & \mathrm{Preorder} \\ B^{\leq}_{(a_0\ a_1 : A)} & : & a_0 \leq^{A} a_1 & \to & B\,a_0 \leadsto B\,a_1 \end{array}$$

indexed variants of $\mathtt{reflexive}, \mathtt{transitive} \ldots$

Multiple choices for $(\leadsto)$:

- Relations respecting the order
- Monotone maps
- Galois connections
- Embedding-projection pairs $X \lhd Y$

$$\begin{array}{l} \uparrow : X \to Y \\ \downarrow : Y \to X \end{array} \quad \text{such that} \quad \begin{cases} \uparrow x \leq^{Y} y \Leftrightarrow x \leq^{X} \downarrow y \\ \downarrow\uparrow x = x \end{cases}$$

Interpretation of type constructors

Natural numbers

$$\vdash 0 : \mathbb{N} \qquad\qquad \vdash \mathtt{S} : \mathbb{N} \to \mathbb{N}$$

$$\vdash \mathtt{0le0} : 0 \leq^{\mathbb{N}} 0 \qquad\qquad \frac{\vdash \mathit{pf} : p \leq^{\mathbb{N}} q}{\vdash \mathtt{CongrS}\,\mathit{pf} : \mathtt{S}\ p \leq^{\mathbb{N}} \mathtt{S}\ q}$$

Order relation $\leq^{\mathbb{N}}$ induced by parametricity [Bernardy-Lasson]

Dependent products

$$\begin{aligned}(a : A) \xrightarrow{\text{mon}} B := \{\ f : (a : A) \to B\ \mid \\ (a_{01} : a_0 \leq^{A} a_1) \to B^{\leq}\ a_{01}\ (f\ a_0)\ (f\ a_1)\ \}\end{aligned}$$

$$f \leq g := (a : A) \to f\ a \leq^{B\,a} g\ a$$

Models for Gradual Types

Mixing orders and exceptions

Required ingredients for a Gradual model:

- Types X endowed with a *precision* preorder $\sqsubseteq^X$
- Universal placeholders $?_X$ such that $\forall x : X, x \sqsubseteq^X ?_X$
- Errors $\texttt{raise}_X$ such that $\forall x : X, \texttt{raise}_X \sqsubseteq^X x$
- Whenever $X \sqsubseteq^{\mathbb{U}} Y$, a pair of an upcast $\uparrow : X \to Y$ and a downcast $\downarrow : Y \to X$ forming an ep-pair $(\uparrow, \downarrow) : X \lhd Y$

Mixing orders and exceptions

Required ingredients for a Gradual model:

- Types X endowed with a *precision* preorder $\sqsubseteq^X$
- Universal placeholders $?_X$ such that $\forall x : X, x \sqsubseteq^X ?_X$
- Errors $\mathtt{raise}_X$ such that $\forall x : X, \mathtt{raise}_X \sqsubseteq^X x$
- Whenever $X \sqsubseteq^{\mathbb{U}} Y$, a pair of an upcast $\uparrow : X \to Y$ and a downcast $\downarrow : Y \to X$ forming an ep-pair $(\uparrow, \downarrow) : X \lhd Y$

Natural numbers

$$\vdash 0 : \mathbb{N} \qquad \vdash \mathsf{S} : \mathbb{N} \to \mathbb{N} \qquad \vdash ?_{\mathbb{N}} : \mathbb{N} \qquad \vdash \mathtt{raise}_{\mathbb{N}} : \mathbb{N}$$

$$0 \sqsubseteq^{\mathbb{N}} 0 \qquad \frac{p \sqsubseteq^{\mathbb{N}} q}{\mathsf{S}\ p \sqsubseteq \mathsf{S}\ q} \qquad \mathtt{raise}_{\mathbb{N}} \sqsubseteq^{\mathbb{N}} p \qquad 0, ?_{\mathbb{N}} \sqsubseteq ?_{\mathbb{N}} \qquad \frac{p \sqsubseteq ?_{\mathbb{N}}}{\mathsf{S}\ p \sqsubseteq ?_{\mathbb{N}}}$$

An Inductive-recursive hierarchy of Universes

Key ideas

1. Universes and their precision order must be defined mutually

$$\frac{\vdash A : \mathbb{U}_i \qquad \vdash B : A \xrightarrow{\text{mon}} \mathbb{U}_i}{\vdash (a : A) \xrightarrow{\texttt{mon}} B\, a : \mathbb{U}_i}$$

An Inductive-recursive hierarchy of Universes

Key ideas

1. Universes and their precision order must be defined mutually

$$\frac{\vdash A : \mathbb{U}_i \qquad \vdash B : A \xrightarrow{\texttt{mon}} \mathbb{U}_i}{\vdash (a : A) \xrightarrow{\texttt{mon}} B\, a : \mathbb{U}_i}$$

2. $X \sqsubseteq^{\mathbb{U}} Y$ irrelevant requires *intensional* data on types

An Inductive-recursive hierarchy of Universes

Key ideas

1. Universes and their precision order must be defined mutually

$$\frac{\vdash A : \mathbb{U}_i \qquad \vdash B : A \xrightarrow{\texttt{mon}} \mathbb{U}_i}{\vdash \pi\ A\ B : \mathbb{U}_i} \qquad \mathrm{El}\,(\pi\, A\, B) := (a : A) \xrightarrow{\texttt{mon}} B\, a$$

2. $X \sqsubseteq^{\mathbb{U}} Y$ irrelevant requires *intensional* data on types
   Inductive universe of codes $\mathbb{U}_i$ and
   Recursive decoding function $\mathrm{El} : \mathbb{U}_i \to \mathit{Type}$

An Inductive-recursive hierarchy of Universes

Key ideas

1. Universes and their precision order must be defined mutually

$$\frac{\vdash A : \mathbb{U}_i \qquad \vdash B : A \xrightarrow{\texttt{mon}} \mathbb{U}_i}{\vdash \pi\ A\ B : \mathbb{U}_i} \qquad \mathrm{El}\,(\pi\, A\, B) := (a : A) \xrightarrow{\texttt{mon}} B\, a$$

2. $X \sqsubseteq^{\mathbb{U}} Y$ irrelevant requires *intensional* data on types
   Inductive universe of codes $\mathbb{U}_i$ and
   Recursive decoding function $\mathrm{El} : \mathbb{U}_i \to \mathit{Type}$

3. Precision on codes decodes to embedding-projection pairs

$$\mathrm{El}^{\mathsf{rel}} \quad : \quad X \sqsubseteq Y \quad \to \quad X \triangleleft Y$$

$\rightsquigarrow$ induces casts $\uparrow, \downarrow$ between types

ω-cpo and the Scott model

$?_{\mathbb{U}} : \mathbb{U}$ (by def)

ω-cpo and the Scott model

$$?_{\mathbb{U}} : \mathbb{U} \qquad \text{(by def)}$$

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} : \mathbb{U} \qquad (\mathbb{U} \text{ closed under } \to)$$

ω-cpo and the Scott model

$$?_{\mathbb{U}} : \mathbb{U} \qquad \text{(by def)}$$

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} : \mathbb{U} \qquad (\mathbb{U} \text{ closed under } \to)$$

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} \sqsubseteq ?_{\mathbb{U}} \qquad (? \text{ maximal for } \sqsubseteq)$$

ω-cpo and the Scott model

$$?_{\mathbb{U}} : \mathbb{U}$$ (by def)

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} : \mathbb{U}$$ ($\mathbb{U}$ closed under $\to$)

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} \sqsubseteq ?_{\mathbb{U}}$$ ($?$ maximal for $\sqsubseteq$)

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} \lhd ?_{\mathbb{U}}$$ (by decoding)

$?_{\mathbb{U}}$ hosts a model of pure λ-calculus

ω-cpo and the Scott model

$$?_{\mathbb{U}} : \mathbb{U} \qquad \text{(by def)}$$

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} : \mathbb{U} \qquad (\mathbb{U} \text{ closed under } \to)$$

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} \sqsubseteq ?_{\mathbb{U}} \qquad (? \text{ maximal for } \sqsubseteq)$$

$$?_{\mathbb{U}} \to ?_{\mathbb{U}} \lhd ?_{\mathbb{U}} \qquad \text{(by decoding)}$$

$?_{\mathbb{U}}$ hosts a model of pure λ-calculus

Let's go back to Scott's domain theory

Add an ω-cpo structure on a type A:

$$\mathsf{sup}^A \quad : \quad (\omega \xrightarrow{\mathsf{mon}} A) \quad \longrightarrow \quad A$$

Dynamic type $?_{\mathbb{U}}$ as a sequential colimit

$$\begin{tikzcd}
\perp \arrow[r, "\lhd"] \arrow[drrrrr, bend right] & F\perp \arrow[r, "\lhd"] \arrow[drrrr, bend right] & \cdots \arrow[r, "\lhd"] & F^n\perp \arrow[rr] \arrow[drr] & & \ldots \\
& & & & & \operatorname{colim}_{n\in\omega} F^n\perp = ?_{\mathbb{U}}
\end{tikzcd}$$

where

$$F\,X \quad \cong \quad \mathbb{N} \quad + \quad X \to X \quad + \quad \ldots$$

Dynamic type $?_{\mathbb{U}}$ as a sequential colimit

$$\begin{tikzcd}
\bot \arrow[r, "\lhd"] \arrow[drrrr, bend right] & F\bot \arrow[r, "\lhd", "\cdots"'] \arrow[drrr, bend right] & F^n\bot \arrow[r] \arrow[drr] & \ldots \\
 & & & & \operatorname{colim}_{n\in\omega} F^n\bot = ?_{\mathbb{U}}
\end{tikzcd}$$

where

$$F\,X \quad \cong \quad \mathbb{N} \quad + \quad X \to X \quad + \quad \ldots$$

What's a typical element of $?_{\mathbb{U}}$

- a tag corresponding to a summand of F, e.g. $\to$
- and an element of the corresponding type, e.g. $?_{\mathbb{U}} \to ?_{\mathbb{U}}$

$A \sqsubseteq ?_{\mathbb{U}}$ decomposes elements along the structure of A !

Conclusion

Recap

- CIC is a subtle equilbrium
- …and I passed over many important details
  (impredicativity, indexed types, induction-recursion)
- Syntactic models can help prototyping extensions
- Even simple objects (orders) give rise to a whole spectrum

Further directions

- Study these models systematically
- As well as how they relate !
- Design full-fledge type theories (hard !)