Models of CIC for Gradual Dependent Types

Kenji Maillard

Inria Nantes, team Gallinette

j.w.w.



Meven Lennon-Bertrand

Nicolas Tabareau

Éric Tanter

GdT Plume, Ens Lyon
Monday the 13th of December, 2021

Proof assistants and dependent types

Agda

Idris

LEAN
THEOREM PROVER

Foundations: dependent types (CIC)
Rich logical system & expressive programming language

Trustworthy established metatheory (canonicity, consistency)
Reliable decidable typechecking
Practical ?

The joy of indexed types


```
Inductive vec (A : □) : ℕ → □ :=
| nil  : vec A 0
| cons : A → forall n : ℕ, vec A n → vec A (S n).
```

The joy of indexed types


```
Inductive vec (A : □) : ℕ → □ :=
| nil  : vec A 0
| cons : A → forall n : ℕ, vec A n → vec A (S n).
```

```
head : forall A n, vec A (S n) → A
```

Total function

The joy of indexed types


```
Inductive vec (A : □) : ℕ → □ :=
| nil  : vec A 0
| cons : A → forall n : ℕ, vec A n → vec A (S n).
```

```
head : forall A n, vec A (S n) → A
```

Total function

```
filter : forall A n (f : A → 𝔹), vec A n → vec A ...
```

No simple way to specify the return index

Gradual Typing

Transition between static and dynamic typing.

$$\mathsf{Int} \to \mathsf{Int} \;\sqsubseteq\; \mathsf{Int} \to ? \;\sqsubseteq\; ? \to ? \;\sqsubseteq\; ?$$

- Imprecise unknown type ?
- Typing weakened to deal optimistically with precision

$$\frac{\Gamma \vdash t : T \qquad T \sim U}{\Gamma \vdash t : U}$$

$$A \sim ?$$

Gradual Typing

Transition between static and dynamic typing.

$$\mathsf{Int} \to \mathsf{Int} \;\sqsubseteq\; \mathsf{Int} \to ? \;\sqsubseteq\; ? \to ? \;\sqsubseteq\; ?$$

- Imprecise unknown type ?
- Typing weakened to deal optimistically with precision

$$\frac{\Gamma \vdash t : T \qquad T \sim U}{\Gamma \vdash t : U} \qquad A \sim ?$$

Runtime-checks to restore invariants for imprecise types.

$$(\lambda x : ?.\, x + 3)\ 5 \quad \rightsquigarrow \quad 8$$

$$(\lambda x : ?.\, x + 3)\ \texttt{true} \quad \rightsquigarrow \quad \texttt{err}_{\mathbb{N}}$$

Gradual Dependent types using examples

Fixing `filter`

$$\texttt{filter} : \texttt{forall A n (f : A} \to \mathbb{B}\texttt{), vec A n} \to \texttt{vec A ?}$$

Gradual Dependent types using examples

Fixing `filter`

```
filter : forall A n (f : A → 𝔹), vec A n → vec A ?

head ℕ ? (filter ℕ 4 even [ 0 ; 1 ; 2 ; 3 ]) (* ⇒  0  *)
head ℕ ? (filter ℕ 2 even [ 1 ; 3 ])         (* ⇒ err *)
```

Gradual Dependent types using examples

Fixing `filter`

```
filter : forall A n (f : A → 𝔹), vec A n → vec A ?
```

```
head ℕ ? (filter ℕ 4 even [ 0 ; 1 ; 2 ; 3 ]) (* ⇒  0  *)
head ℕ ? (filter ℕ 2 even [ 1 ; 3 ])          (* ⇒ err *)
```

Varying specifications:

```
Definition foo (n m : Nat) :=
  if (n > m) then m + 1 else m > 0.
```

```
foo : ?
foo : ℕ → ℕ → ?
foo : ℕ → ℕ → if ? then ℕ else ?
foo : forall (n m : ℕ), if (n > m) then ℕ else ?
foo : forall (n m : ℕ), if (n > m) then ℕ else 𝔹
```

Mixing Dependent and Gradual types

GCIC: Gradual Calculus of Inductive Construction

Features from CIC:

- Dependent types (Π, Σ);
- Hierarchy of universes $\square_i : \square_{i+1}$;
- Inductive types $\mathbb{B}, \mathbb{N}, \texttt{list}\, A$

Gradual features:

- Unknown as a term $?_A : A$ $\qquad ?_\square : \square$
- Computational content for ?

$$?_{\Pi(x:A)B} \equiv \lambda x : A.\, ?_B \qquad \texttt{if}\ ?_{\mathbb{B}}\ \texttt{then}\ 0\ \texttt{else}\ 1 \equiv ?_{\mathbb{N}}$$

Well-behavedness for Gradual types

Precision order $A \sqsubseteq B$

$$A \sqsubseteq ?$$

$$\frac{A \sqsubseteq A' \qquad B \sqsubseteq B'}{A \to B \sqsubseteq A' \to B'}$$

Induce a precision order on terms $\lambda x : A.\, t \sqsubseteq \lambda x : ?.\, t$

Well-behavedness for Gradual types

Precision order $A \sqsubseteq B$

$$A \sqsubseteq ?$$

$$\frac{A \sqsubseteq A' \qquad B \sqsubseteq B'}{A \to B \sqsubseteq A' \to B'}$$

Induce a precision order on terms $\lambda x : A.\, t \sqsubseteq \lambda x : ?.\, t$

Gradual garantees [Siek et al. 2015]

- Static gradual garantee: typing is monotone wrt $\sqsubseteq$
- Dynamic gradual garantee: $t \sqsubseteq u \Rightarrow t \preccurlyeq^{obs} u$

Well-behavedness for Gradual types

Precision order $A \sqsubseteq B$

$$A \sqsubseteq ?$$

$$\frac{A \sqsubseteq A' \qquad B \sqsubseteq B'}{A \to B \sqsubseteq A' \to B'}$$

Induce a precision order on terms $\lambda x : A.\, t \sqsubseteq \lambda x : ?.\, t$

Gradual garantees [Siek et al. 2015]

- Static gradual garantee: typing is monotone wrt $\sqsubseteq$
- Dynamic gradual garantee: $t \sqsubseteq u \Rightarrow t \preccurlyeq^{obs} u$

Graduality [New and Ahmed 2018]

$A \sqsubseteq B \Rightarrow A \lhd B$ A retract of B

Well-behavedness for Gradual types

Precision order $A \sqsubseteq B$

$$A \sqsubseteq ? \qquad \frac{A \sqsubseteq A' \qquad B \sqsubseteq B'}{A \to B \sqsubseteq A' \to B'}$$

Induce a precision order on terms $\lambda x : A.\, t \sqsubseteq \lambda x : ?.\, t$

Gradual garantees [Siek et al. 2015]

- Static gradual garantee: typing is monotone wrt $\sqsubseteq$
- Dynamic gradual garantee: $t \sqsubseteq u \Rightarrow t \preccurlyeq^{obs} u$

Graduality [New and Ahmed 2018]

$A \sqsubseteq B \Rightarrow A \lhd B$ $\qquad$ A retract of B

$\forall a : A, b : B, \qquad \uparrow a \sqsubseteq b \iff a \sqsubseteq \downarrow b \qquad a \sqsupseteq\sqsubseteq \downarrow\uparrow a$

$?_\Box$ and untyped λ-calculus

Assuming a GCIC conservative over CIC with graduality

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

$?_\Box$ and untyped λ-calculus

Assuming a GCIC conservative over CIC with graduality

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

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

$?_\square$ and untyped λ-calculus

Assuming a GCIC conservative over CIC with graduality

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

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

$$?_\square \to ?_\square \quad \sqsubseteq \quad ?_\square \qquad (?_\square \text{ maximal for } \sqsubseteq)$$

$?_\Box$ and untyped λ-calculus

Assuming a GCIC conservative over CIC with graduality

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

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

$$?_\Box \to ?_\Box \quad \sqsubseteq \quad ?_\Box \qquad (?_\Box \text{ maximal for } \sqsubseteq)$$

$$?_\Box \to ?_\Box \quad \lhd \quad ?_\Box \qquad \text{(by graduality)}$$

$?_\Box$ and untyped λ-calculus

Assuming a GCIC conservative over CIC with graduality

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

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

$$?_\Box \to ?_\Box \quad \sqsubseteq \quad ?_\Box \qquad (?_\Box \text{ maximal for } \sqsubseteq)$$

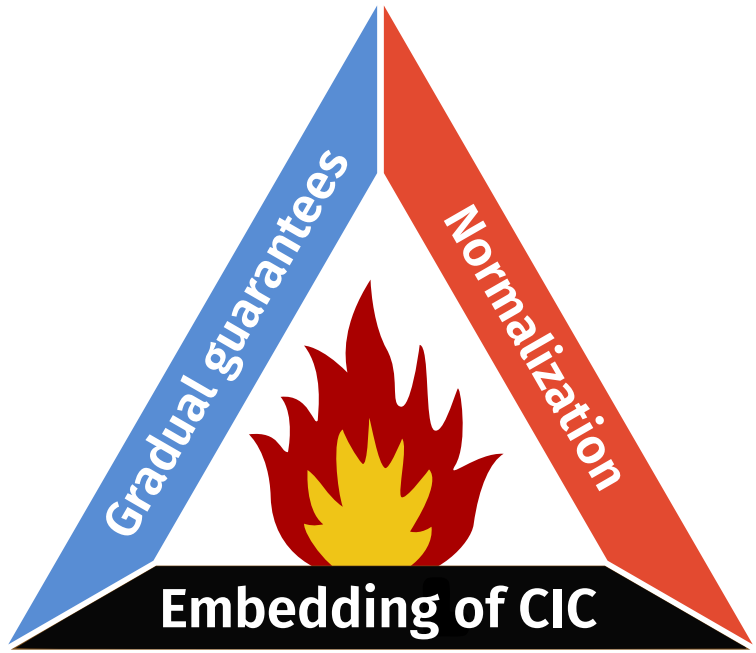
$$?_\Box \to ?_\Box \quad \lhd \quad ?_\Box \qquad \text{(by graduality)}$$

$?_\Box$ hosts a model of pure λ-calculus

$$\Omega \stackrel{def}{:=} (\lambda x : ?_\Box \to ?_\Box . \, x \, x) \; (\lambda x : ?_\Box \to ?_\Box . \, x \, x) : ?_\Box$$

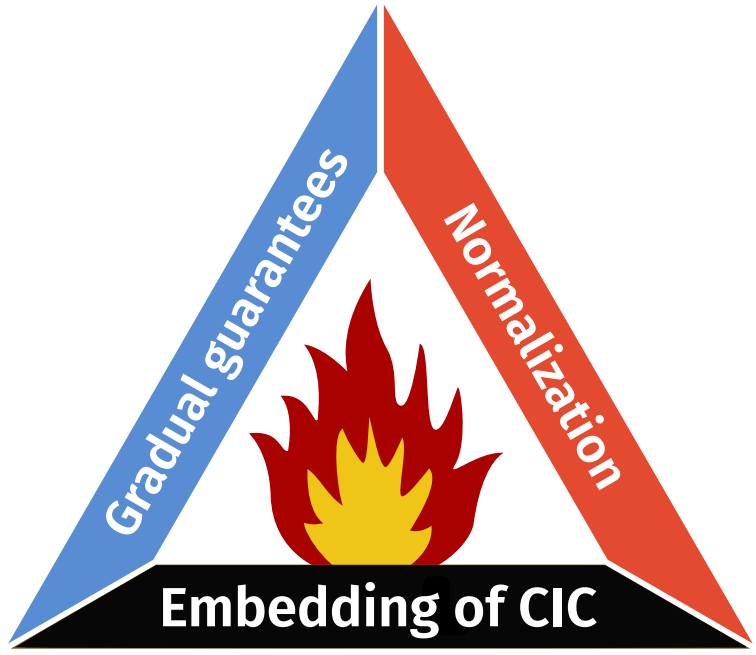
Fire triangle of Graduality

Assuming Safety: No close term can get stuck.



Fire triangle of Graduality

Assuming Safety: No close term can get stuck.



Can we find partial resolutions ?

3 solutions in 1

$\mathsf{GCIC}^{\mathcal{G}}$ not normalizing

$\mathsf{GCIC}^{\mathcal{N}}$ no gradual guarantees

$\mathsf{GCIC}^{\uparrow}$ universes are not closed under $\rightarrow$

3 solutions in 1

$\text{GCIC}^{\mathcal{G}}$ not normalizing

$\text{GCIC}^{\mathcal{N}}$ no gradual guarantees

$\text{GCIC}^{\uparrow}$ universes are not closed under $\to$

Parametrization by universe levels

- $s_\Pi(i,j)$ static level for Π
- $c_\Pi(i)$ dynamic level for Π used by casts

$$\frac{\Gamma \vdash A : \Box_i \qquad \Gamma, x : A \vdash B : \Box_j}{\Gamma \vdash \Pi(x : A)\, B : \Box_{s_\Pi(i,j)}}$$

$$?\Box_{c_\Pi(i)} \to ?\Box_{c_\Pi(i)} \quad \sqsubseteq \quad ?\Box_i$$

3 solutions in 1

$\text{GCIC}^{\mathcal{G}}$ not normalizing
$s_\Pi(i,j) := \max(i,j) \qquad c_\Pi(i) := i$

$\text{GCIC}^{\mathcal{N}}$ no gradual guarantees
$s_\Pi(i,j) := \max(i,j) \qquad c_\Pi(i) := i - 1$

$\text{GCIC}^{\uparrow}$ universes are not closed under $\to$
$s_\Pi(i,j) := \max(i,j) + 1 \qquad c_\Pi(i) := i - 1$

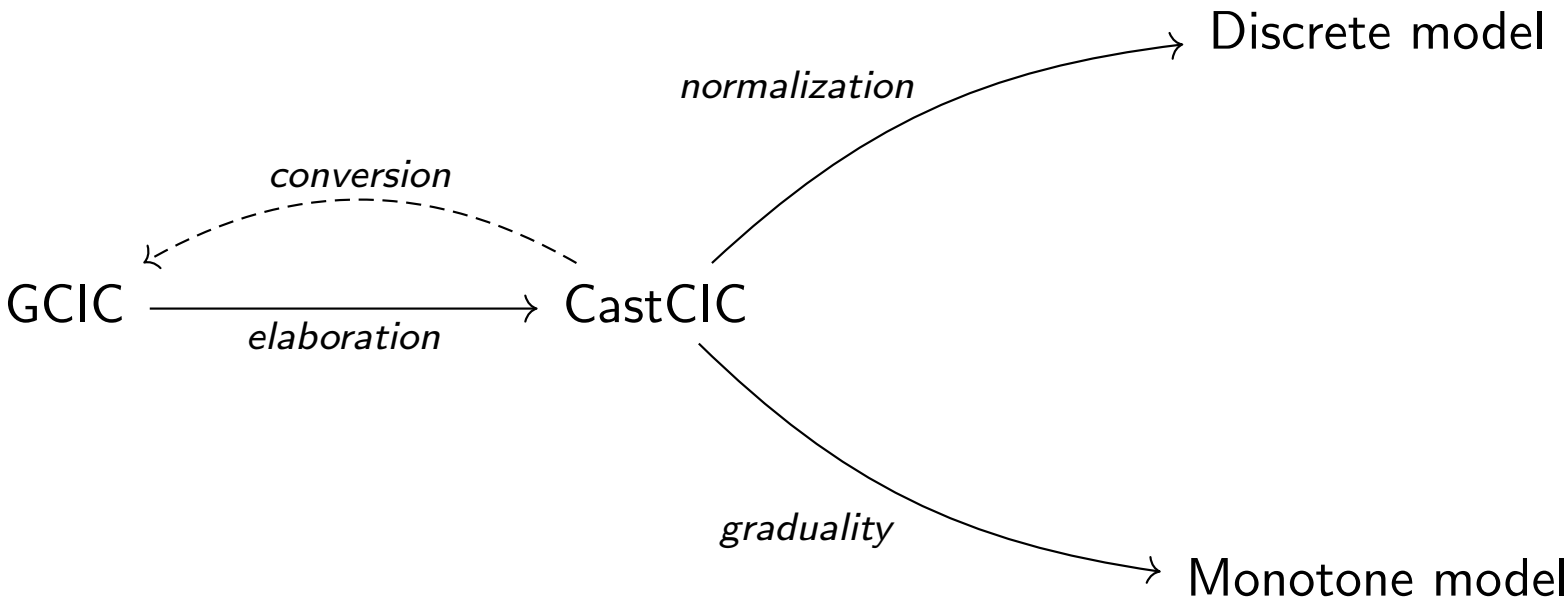
Parametrization by universe levels

- $s_\Pi(i,j)$ static level for Π
- $c_\Pi(i)$ dynamic level for Π used by casts

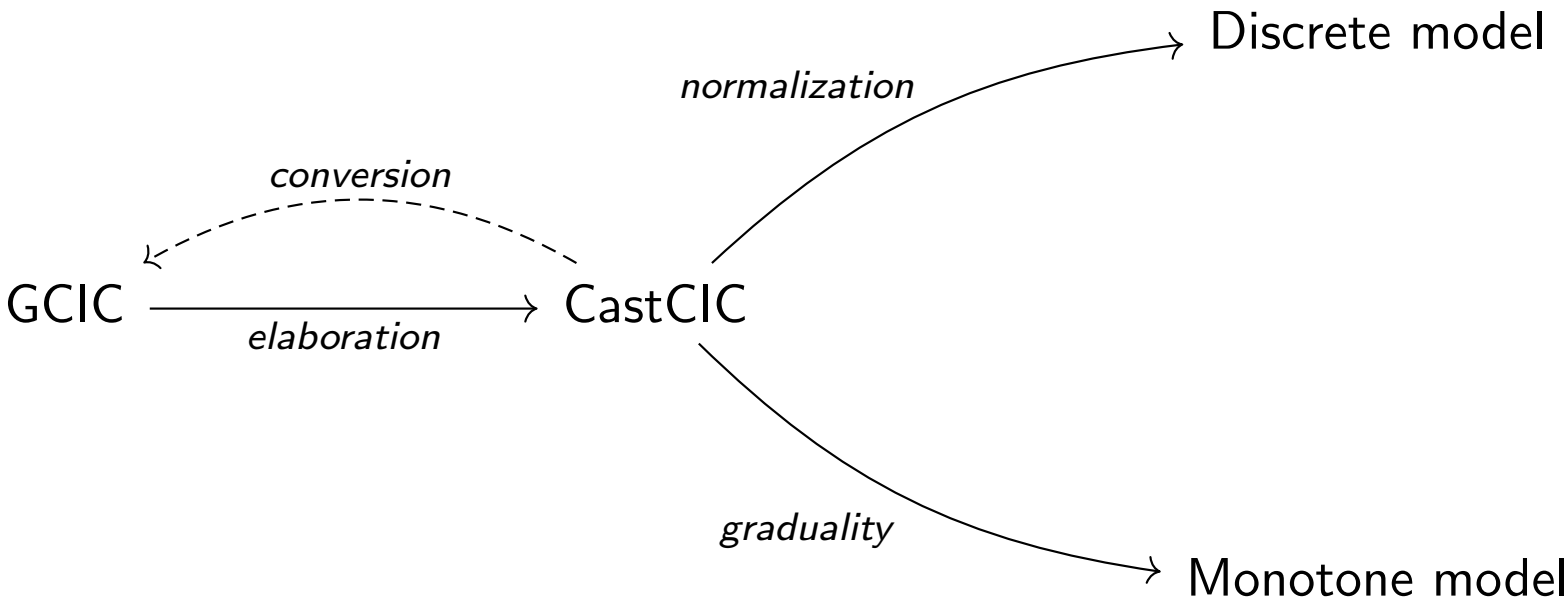
$$\frac{\Gamma \vdash A : \square_i \qquad \Gamma, x : A \vdash B : \square_j}{\Gamma \vdash \Pi(x : A)\, B : \square_{s_\Pi(i,j)}}$$

$$?\square_{c_\Pi(i)} \to ?\square_{c_\Pi(i)} \quad \sqsubseteq \quad ?\square_i$$

From GCIC to CIC



From GCIC to CIC



- GCIC surface language, with implicit type ascriptions
- CastCIC explicit cast calculus
- Models for design and metatheory (variants of CIC)

CastCIC

CIC extended with:

- Unknown terms $?_A : A$
- Error terms $\mathtt{err}_A : A$
- Arbitrary casts $\langle B \Leftarrow A \rangle\, t : B$ whenever $t : A$

CastCIC

CIC extended with:

- Unknown terms $?_A : A$
- Error terms $\mathtt{err}_A : A$
- Arbitrary casts $\langle B \Leftarrow A \rangle\, t : B$ whenever $t : A$

Reduction rules for unknown and errors:

$$
\begin{array}{lcl}
?_{\Pi(x:A)B} & \rightsquigarrow & \lambda x : A.\, ?_B \\
\mathtt{if}\ ?_{\mathbb{B}}\ \mathtt{return}\ T\ \mathtt{then}\ t\ \mathtt{else}\ u & \rightsquigarrow & ?_T \\
\langle T \Leftarrow ?_{\square} \rangle\, ?_{?_{\square}} & \rightsquigarrow & ?_T
\end{array}
$$

CastCIC

CIC extended with:

- Unknown terms $?_A : A$
- Error terms $\mathtt{err}_A : A$
- Arbitrary casts $\langle B \Leftarrow A \rangle\, t : B$ whenever $t : A$

Reduction rules for casts:

$$\langle \mathbb{N} \Leftarrow \mathbb{B} \rangle\, b \rightsquigarrow \mathtt{err}_{\mathbb{N}}$$

$$\langle A_2 \to B_2 \Leftarrow A_1 \to B_1 \rangle\, t \rightsquigarrow \lambda y : A_2.\, \langle B_2 \Leftarrow B_1 \rangle\, (t\ (\langle A_1 \Leftarrow A_2 \rangle\, y))$$

$$\langle ?_{\square_i} \Leftarrow A_1 \to B_1 \rangle\, t \rightsquigarrow \langle ?_i \Leftarrow ?_{\square_{c_\Pi(i)}} \to ?_{\square_{c_\Pi(i)}} \rangle\, \langle ?_{\square_{c_\Pi(i)}} \to ?_{\square_{c_\Pi(i)}} \Leftarrow A_1 \to B_1 \rangle\, t$$

Models of CastCIC

Syntactic Models

Models of CIC in CIC:

- Defined inductively on the syntax of terms/types

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

- 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)

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

Why syntactic models ?

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

Discrete model of CastCIC

Ingredients

- $\mathtt{err}_A, ?_A$ translated to exceptions [Pedrot-Tabareau 2018]
- Pattern matching on universe (ad-hoc polymorphism)

Discrete model of CastCIC

Ingredients

- $\mathtt{err}_A, ?_A$ translated to exceptions [Pedrot-Tabareau 2018]
- Pattern matching on universe (ad-hoc polymorphism)

Realize $?_\square$ as pairs (h, t)

- h code describing a *head constructor* of a type;
- t an element of the decoding of h, e.g. for $h = \Pi$, $t : ?_\square \to ?_\square$

Casts defined by case analysis on the types.

Discrete model of CastCIC

Ingredients

- $\mathtt{err}_A, ?_A$ translated to exceptions [Pedrot-Tabareau 2018]
- Pattern matching on universe (ad-hoc polymorphism)

Realize $?_\square$ as pairs (h, t)

- h code describing a *head constructor* of a type;
- t an element of the decoding of h, e.g. for $h = \Pi$, $t : ?_\square \to ?_\square$

Casts defined by case analysis on the types.

Theorem

The discrete model is a syntactic model. Moreover, if $t \rightsquigarrow u$ *in* CastCIC, $[t] \to_{wh}^{+} [u]$ *in* CIC $+$ *induction-recursion.*

Discrete model of CastCIC

Ingredients

- $\mathtt{err}_A, ?_A$ translated to exceptions [Pedrot-Tabareau 2018]
- Pattern matching on universe (ad-hoc polymorphism)

Realize $?_\square$ as pairs (h, t)

- h code describing a *head constructor* of a type;
- t an element of the decoding of h, e.g. for $h = \Pi$, $t : ?_\square \to ?_\square$

Casts defined by case analysis on the types.

Theorem

The discrete model is a syntactic model. Moreover, if $t \rightsquigarrow u$ *in* CastCIC, $[t] \to_{wh}^{+} [u]$ *in* CIC $+$ *induction-recursion.*

Says nothing about precision/success of casts

Monotone model

Translate each type as a poset $(A, \sqsubseteq^A)$:

$$\sqsubseteq^A \quad : \quad A \to A \to \mathit{Type}$$

$$\texttt{reflexive} : (a : A) \to a \sqsubseteq^A a$$

$$\texttt{transitive} : (a_0\ a_1\ a_2 : A) \to a_0 \sqsubseteq^A a_1 \to a_1 \sqsubseteq^A a_2 \to a_0 \sqsubseteq^A a_2$$

$$\texttt{irrelevant} : (a_0\ a_1 : A)(h\ h' : a_0 \sqsubseteq^A a_1) \to h = h'$$

$$\texttt{antisymmetric} : (a_0\ a_1 : A) \to a_0 \sqsubseteq^A a_1 \to a_1 \sqsubseteq^A a_0 \to a_0 = a_1$$

Monotone model

Translate each type as a poset $(A, \sqsubseteq^A)$:

$$\sqsubseteq^A \quad : \quad A \to A \to \mathit{Type}$$

$$\texttt{reflexive} : (a : A) \to a \sqsubseteq^A a$$

$$\texttt{transitive} : (a_0\ a_1\ a_2 : A) \to a_0 \sqsubseteq^A a_1 \to a_1 \sqsubseteq^A a_2 \to a_0 \sqsubseteq^A a_2$$

$$\texttt{irrelevant} : (a_0\ a_1 : A)(h\ h' : a_0 \sqsubseteq^A a_1) \to h = h'$$

$$\texttt{antisymmetric} : (a_0\ a_1 : A) \to a_0 \sqsubseteq^A a_1 \to a_1 \sqsubseteq^A a_0 \to a_0 = a_1$$

Natural numbers

$$\vdash 0 : \mathbb{N} \qquad \vdash \texttt{suc} : \mathbb{N} \to \mathbb{N} \qquad \vdash ?_{\mathbb{N}} : \mathbb{N} \qquad \vdash \texttt{err}_{\mathbb{N}} : \mathbb{N}$$

Monotone model

Translate each type as a poset $(A, \sqsubseteq^A)$:

$$\sqsubseteq^A \quad : \quad A \to A \to \mathit{Type}$$

$$\texttt{reflexive} : (a : A) \to a \sqsubseteq^A a$$

$$\texttt{transitive} : (a_0\, a_1\, a_2 : A) \to a_0 \sqsubseteq^A a_1 \to a_1 \sqsubseteq^A a_2 \to a_0 \sqsubseteq^A a_2$$

$$\texttt{irrelevant} : (a_0\, a_1 : A)(h\, h' : a_0 \sqsubseteq^A a_1) \to h = h'$$

$$\texttt{antisymmetric} : (a_0\, a_1 : A) \to a_0 \sqsubseteq^A a_1 \to a_1 \sqsubseteq^A a_0 \to a_0 = a_1$$

Natural numbers

$$\vdash 0 : \mathbb{N} \qquad \vdash \texttt{suc} : \mathbb{N} \to \mathbb{N} \qquad \vdash ?_{\mathbb{N}} : \mathbb{N} \qquad \vdash \texttt{err}_{\mathbb{N}} : \mathbb{N}$$

$$0 \sqsubseteq^{\mathbb{N}} 0 \qquad \texttt{err}_{\mathbb{N}} \sqsubseteq^{\mathbb{N}} p \qquad 0, ?_{\mathbb{N}} \sqsubseteq ?_{\mathbb{N}}$$

$$\frac{p \sqsubseteq^{\mathbb{N}} q}{\texttt{suc}\ p \sqsubseteq \texttt{suc}\ q} \qquad \frac{p \sqsubseteq ?_{\mathbb{N}}}{\texttt{suc}\, p \sqsubseteq ?_{\mathbb{N}}}$$

Posetal families

Translate a type family $x : A \vdash B$ type as

$$
\begin{array}{rcccc}
B & : & A & \to & \mathsf{Poset} \\
B^{\sqsubseteq}_{(a_0\ a_1 : A)} & : & a_0 \sqsubseteq^A a_1 & \to & B\ a_0 \lhd B\ a_1
\end{array}
$$

indexed variants of `reflexive`, `transitive` ...

Posetal families

Translate a type family $x : A \vdash B$ type as

$$\begin{array}{rcccc} B & : & A & \to & \mathsf{Poset} \\ B^{\sqsubseteq}_{(a_0\ a_1 : A)} & : & a_0 \sqsubseteq^A a_1 & \to & B\ a_0 \lhd B\ a_1 \end{array}$$

indexed variants of `reflexive`, `transitive` ...

Dependent products

$$(a : A) \xrightarrow{\mathsf{mon}} B := \{\ f : (a : A) \to B \ \mid\ (a_{01} : a_0 \sqsubseteq^A a_1) \to B^{\sqsubseteq}\ a_{01}\ (f\ a_0)\ (f\ a_1)\ \}$$

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

An Inductive-recursive hierarchy of Universes

Key ideas $\boxed{\square}_i = [\![\square_i]\!]$

1. Universes and their precision order must be defined mutually

$$\frac{\vdash A : \boxed{\square}_i \qquad \vdash B : A \xrightarrow{\text{mon}} \boxed{\square}_i}{\vdash (a : A) \xrightarrow{\text{mon}} B\, a : \boxed{\square}_i}$$

An Inductive-recursive hierarchy of Universes

Key ideas $\boxed{\square}_i = [\![\square_i]\!]$

1. Universes and their precision order must be defined mutually

$$\frac{\vdash A : \boxed{\square}_i \qquad \vdash B : A \xrightarrow{\text{mon}} \boxed{\square}_i}{\vdash (a : A) \xrightarrow{\text{mon}} B\, a : \boxed{\square}_i}$$

2. $X \sqsubseteq^{\boxed{\square}} Y$ irrelevance requires *intensional* data on types

An Inductive-recursive hierarchy of Universes

Key ideas $\boxbox_i = [\![\Box_i]\!]$

1. Universes and their precision order must be defined mutually

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

2. $X \sqsubseteq^{\Box} Y$ irrelevance requires *intensional* data on types
   Inductive universe of codes $\boxbox_i$ and
   Recursive decoding function $\mathrm{El} : \boxbox_i \to \text{Poset}$

An Inductive-recursive hierarchy of Universes

Key ideas $\boxbox_i = [\![\Box_i]\!]$

1. Universes and their precision order must be defined mutually

$$\frac{\vdash A : \boxbox_i \qquad \vdash B : A \xrightarrow{\text{mon}} \boxbox_i}{\vdash \pi\ A\ B : \boxbox_i} \qquad \mathrm{El}\,(\pi\, A\, B) := (a : A) \xrightarrow{\text{mon}} B\, a$$

2. $X \sqsubseteq^{\Box} Y$ irrelevance requires *intensional* data on types
   Inductive universe of codes $\boxbox_i$ and
   Recursive decoding function $\mathrm{El} : \boxbox_i \to \text{Poset}$

3. Precision on codes decodes to embedding-projection pairs

$$\mathrm{El}^{\text{rel}} \quad : \quad X \sqsubseteq Y \quad \to \quad X \lhd Y$$

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

Precision order on the universe

Precision order $\sqsubseteq$ on the universes (where $i \leq j$)

$$\frac{A : \square_j}{\widehat{\mathtt{err}}_i \sqsubseteq A}\ \widehat{\mathtt{err}}\text{-}\sqsubseteq \qquad \frac{}{\widehat{\mathbb{N}} \sqsubseteq \widehat{\mathbb{N}}}\ \widehat{\mathbb{N}}\text{-}\sqsubseteq \qquad \frac{}{\widehat{\square}_i \sqsubseteq \widehat{\square}_i}\ \widehat{\square}\text{-}\sqsubseteq \qquad \frac{}{\widehat{?}_i \sqsubseteq \widehat{?}_j}\ \widehat{?}\text{-}\sqsubseteq$$

$$\frac{A \sqsubseteq A' \qquad a : \mathrm{El}\, A, a' : \mathrm{El}\, A', a_\epsilon : a\ {}_A\sqsubseteq_{A'}\ a' \vdash B\, a \sqsubseteq B'\, a'}{\widehat{\Pi}\, A\, B \sqsubseteq \widehat{\Pi}\, A'\, B'}\ \widehat{\Pi}\text{-}\sqsubseteq$$

$$\frac{h = \mathrm{head}\, A \in \mathrm{Head}_i \qquad A \sqsubseteq \widehat{\mathrm{germ}}_j\, h}{A \sqsubseteq \widehat{?}_j}\ \mathrm{Head}\text{-}\sqsubseteq$$

Precision on terms ${}_A\sqsubseteq_B := \mathrm{El}_\varepsilon(A \sqsubseteq B)$

$$\frac{a : \mathrm{El}\, A}{() \ {}_{\widehat{\mathtt{err}}}\sqsubseteq_A\ a}\ \mathrm{El}_\varepsilon(\widehat{\mathtt{err}}\text{-}\sqsubseteq) \qquad \frac{A = \widehat{\mathbb{N}}, \widehat{\square}_i \qquad x \sqsubseteq^{\mathrm{El}\, A} y}{x\ {}_A\sqsubseteq_A\ y}\ \mathrm{El}_\varepsilon(\widehat{\mathbb{N}}\text{-}\sqsubseteq), \mathrm{El}_\varepsilon(\widehat{\square}\text{-}\sqsubseteq)$$

$$\frac{z : \ddot{?}_j}{\mathtt{err}_{\widehat{?}_i}\ {}_{\widehat{?}_i}\sqsubseteq_{\widehat{?}_j}\ z}\ \mathrm{El}_\varepsilon(\widehat{?}\text{-}\sqsubseteq) \qquad \frac{z : \ddot{?}_i}{z\ {}_{\widehat{?}_i}\sqsubseteq_{\widehat{?}_j}\ ?_{\widehat{?}_j}} \qquad \frac{x\ {}_{\widehat{\mathrm{germ}}_i\, h}\sqsubseteq_{\widehat{\mathrm{germ}}_j\, h}\ x'}{[h; x]\ {}_{\widehat{?}_i}\sqsubseteq_{\widehat{?}_j}\ [h; x']}$$

$$\frac{a : \mathrm{El}\, A, a' : \mathrm{El}\, A', a_\epsilon : a\ {}_A\sqsubseteq_{A'}\ a' \vdash f\ a\ {}_{B\, a}\sqsubseteq_{B'\, a'}\ f'\ a'}{f\ {}_{\widehat{\Pi}\, A\, B}\sqsubseteq_{\widehat{\Pi}\, A'\, B'}\ f'}\ \mathrm{El}_\varepsilon(\widehat{\Pi}\text{-}\sqsubseteq)$$

$$\frac{a\ {}_A\sqsubseteq_{\widehat{\mathrm{germ}}_j\ (\mathrm{head}\, A)}\ x}{a\ {}_A\sqsubseteq_{\widehat{?}_j}\ [\mathrm{head}\, A; x]}\ \mathrm{El}_\varepsilon(\mathrm{Head}\text{-}\sqsubseteq) \qquad \frac{a : \mathrm{El}\, A}{a\ {}_A\sqsubseteq_{\widehat{?}_j}\ ?_{\widehat{?}_j}}$$

$$[\langle U \Leftarrow T \rangle\, t] \quad := \quad \downarrow^{?\square}_{[\![U]\!]} \uparrow^{?\square}_{[\![T]\!]} [t]$$

Graduality

Define a semantic notion of precision:

$$\Gamma \vdash_{\mathtt{cast}} t \ {}_T\sqsubseteq_U \ u \qquad := \qquad \exists e, \quad [\![\Gamma]\!] \vdash e : [t] \ {}_{[\![T]\!]}\sqsubseteq_{[\![U]\!]} \ [u].$$

Theorem

- *Syntactic precision implies semantic precision:*

$$t \sqsubseteq u \qquad \Longrightarrow \qquad \Gamma \vdash t \ {}_T\sqsubseteq_U \ u$$

- *(DGG) If* $\Gamma \vdash_{\mathtt{cast}} t \ {}_T\sqsubseteq_T \ t'$ *then* $t \preccurlyeq^{obs} t'$
- *(Graduality) If* $\Gamma \vdash_{\mathtt{cast}} T \sqsubseteq U$ *then*

$$(\langle U \Leftarrow T \rangle -, \langle T \Leftarrow U \rangle -) \quad : T \lhd U$$

Beyond gradual dependent types

Are we ready for gradual dependent types ?

Pros:

- Faster prototyping in proof assistant
- Accessible dependently typed programming ?

Cons:

- No full story for indexed inductives
- Precision reasoning is external

Are we ready for gradual dependent types ?

Pros:

- Faster prototyping in proof assistant
- Accessible dependently typed programming ?

Cons:

- No full story for indexed inductives
- Precision reasoning is external
- Controversial tradeoffs (normalization, universe level of Π)

Are we ready for gradual dependent types ?

Pros:

- Faster prototyping in proof assistant
- Accessible dependently typed programming ?

Cons:

- No full story for indexed inductives
- Precision reasoning is external
- Controversial tradeoffs (normalization, universe level of Π)

Could we implement gradual dependent types as opt-ins ?

Domain Specific Logics

Domains Specific Languages for specifications and proofs:
Extend proof-assistants on a per-domain basis.

Future challenges

- Implementing DSL (rewrite rules [Winterhalter et al. 2020])
- Methodology for DSL's metatheory
- Controlling DSL interactions (multiverse)
- Proof-engineering with DSL (sort-polymorphism)

Domain Specific Logics

Domains Specific Languages for specifications and proofs:
Extend proof-assistants on a per-domain basis.

Future challenges

- Implementing DSL (rewrite rules [Winterhalter et al. 2020])
- Methodology for DSL's metatheory
- Controlling DSL interactions (multiverse)
- Proof-engineering with DSL (sort-polymorphism)

Thank you ! Any questions ?

Additional Material

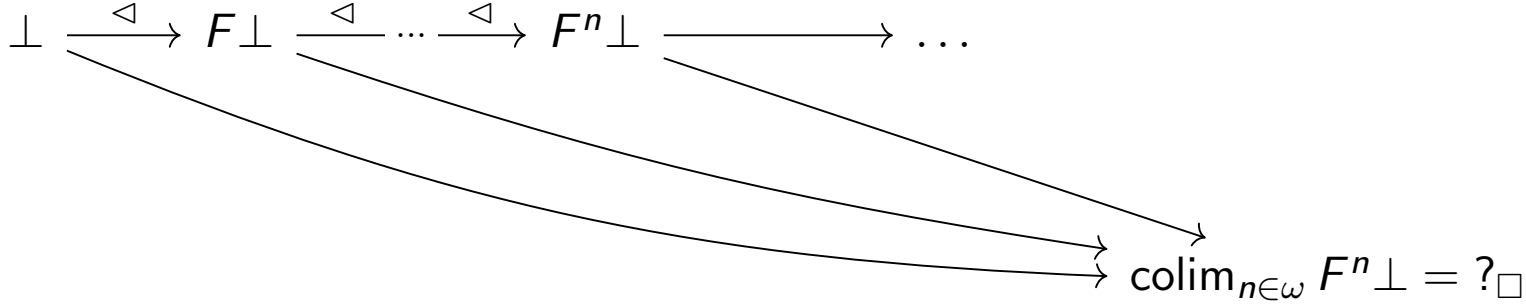
CastCIC$^{\mathcal{G}}$ and ω-cpo

Let's go back to Scott's domain theory

Add an ω-cpo structure on each type A:

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

Dynamic type $?_\square$ as a sequential colimit



where

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

Syntactic Models: A Recipe

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

Crucial steps

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

Syntactic Models: A Recipe

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

Crucial steps

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

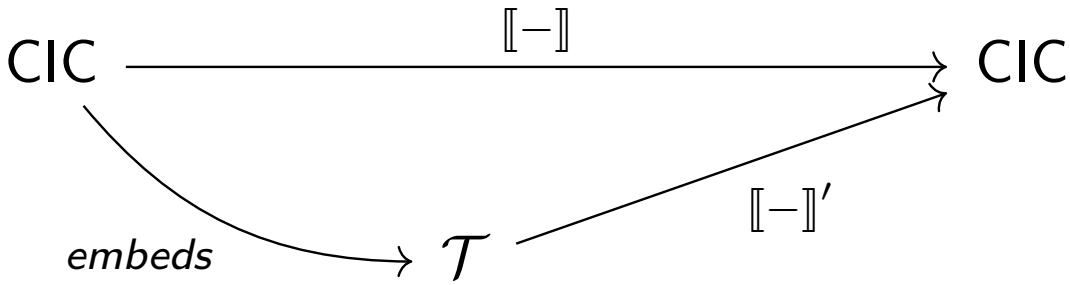
Syntactic Models: A Recipe

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

Crucial steps

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

Syntactic Models: A Recipe



Crucial steps

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

Motivation for models of CIC

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**

Examples of models 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