Gradualizing the Calculus of Inductive Constructions

M2 Internship Presentation

Meven Bertrand, supervised by Nicolas Tabareau

September 6 2019

Introduction

Gallinette team, LS2N, Nantes

Proof assistants (Coq), all over the place:

- Concrete implementation
- Proof assistant
- Theoretical background: type theory, effects, compilation...

Collaboration

Éric Tanter, Inria Paris/UChile

Gradual typing

- Combine static and dynamic typing (Python 3.5, C#, Ruby, JavaScript...)
- Disciplined!

Gradual typing

- Combine static and dynamic typing (Python 3.5, C#, Ruby, JavaScript...)
- Disciplined!

Calculus of (Inductive) Constructions

- Powerful and expressive type theory/logics
- Old system, new features: errors

Gradual typing

- Combine static and dynamic typing (Python 3.5, C#, Ruby, JavaScript...)
- Disciplined!

Calculus of (Inductive) Constructions

- Powerful and expressive type theory/logics
- Old system, new features: errors

But... Why?

Concretely Making programming in Coq easier/doable

Theoretically Good test for gradual typing and exceptional type theory

Plan

Gradual Typing

What is the behaviour of $(\lambda x.\mathrm{S}\ x)\ \mathrm{true}$?

What is the behaviour of $(\lambda x.\mathrm{S}\ x)\ \mathtt{true}$?

Static typing

Error at compile time

$$\nvdash (\lambda x : \mathbf{B}.\mathrm{S}\ x)\ \mathtt{true}$$

What is the behaviour of $(\lambda x.\mathrm{S}\ x)\ \mathrm{true}$?

Static typing

Error at compile time

$$\nvdash (\lambda x : \mathbf{B}.\mathrm{S}\ x)\ \mathrm{true}$$

Untyped λ-calculus

No error at compile time

$$\vdash (\lambda x.\mathrm{S}\ x)\ \mathrm{true} : ?$$

Untrapped error

$$(\lambda x.\mathrm{S}\ x)\ \mathrm{true} \mapsto \mathrm{S}\ \mathrm{true} \mapsto \lambda x\ n\ y.n\ \mathrm{true}\ x \qquad (\lambda x.\mathrm{S}\ x)\ \mathrm{true} \mapsto \mathrm{S}\ \mathrm{true} \not\mapsto$$

ENS DE LYON

LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

What is the behaviour of $(\lambda x.\mathrm{S}\ x)\ \mathrm{true}$?

Static typing

Error at compile time

$$\nvdash (\lambda x : \mathbf{B}.\mathrm{S}\ x)\ \mathrm{true}$$

Untyped λ-calculus

No error at compile time

$$\vdash (\lambda x.\mathrm{S}\ x)\ \mathrm{true} : ?$$

Untrapped error

$$(\lambda x.\mathrm{S}\ x)\ \mathrm{true} \mapsto \mathrm{S}\ \mathrm{true} \mapsto \lambda x\ n\ y.n\ \mathrm{true}\ x \qquad (\lambda x.\mathrm{S}\ x)\ \mathrm{true} \mapsto \mathrm{S}\ \mathrm{true} \not\mapsto$$

Dynamic typing

No error at compile time

$$\vdash (\lambda x.\mathrm{S}\ x)\ \mathrm{true} : ?$$

Trapped error

$$(\lambda x.\mathrm{S}\ x)\ \mathrm{true} \mapsto \mathrm{S}\ \mathrm{true} \mapsto \mathrm{raise}$$

What is the behaviour of $(\lambda x.\mathrm{S}\ x)\ \mathrm{true}$?

Static typing

Error at compile time

$$\nvdash (\lambda x : \mathbf{B}.\mathrm{S}\ x)\ \mathrm{true}$$

Untyped λ-calculus

No error at compile time

$$\vdash (\lambda x.\mathrm{S}\ x)\ \mathrm{true} : ?$$

Untrapped error

$$(\lambda x.\mathrm{S}\ x)\ \mathrm{true} \mapsto \mathrm{S}\ \mathrm{true} \mapsto \lambda x\ n\ y.n\ \mathrm{true}\ x \qquad (\lambda x.\mathrm{S}\ x)\ \mathrm{true} \mapsto \mathrm{S}\ \mathrm{true} \not\mapsto$$

Dynamic typing

No error at compile time

$$\vdash (\lambda x.\mathrm{S}\ x)\ \mathrm{true} : ?$$

Trapped error

$$(\lambda x.\mathrm{S}\ x)\ \mathrm{true} \mapsto \mathrm{S}\ \mathrm{true} \mapsto \mathrm{raise}$$

Mixing static & dynamic disciplines

With a safe head function:

$v : \mathrm{Vect}\ 2 \vdash \mathrm{head}\ v$	typechecks
$v : \mathrm{Vect}\ \ ? \vdash \mathrm{head}\ v$	dynamic check of the actual type of v
$v : \mathbf{N} \nvdash \mathrm{head}\ v$	type error

Mixing static & dynamic disciplines

With a safe head function:

$v : \mathrm{Vect}\ 2 \vdash \mathrm{head}\ v$	typechecks
$v : \mathrm{Vect}\ \ ? \vdash \mathrm{head}\ v$	dynamic check of the actual type of v
$v : \mathbf{N} \nvdash \mathrm{head}\ v$	type error

Evolving types

$$\begin{tikzcd} & (\lambda x : ?\,.\, x\, 1)\ \mathrm{S} \arrow[dl, no head] \arrow[dr, no head] & \\ (\lambda x : \mathbf{N} \to ?\,.\, x\, 1)\ \mathrm{S} \arrow[d, no head] & & (\lambda x : ? \to \mathbf{N}.\, x\, 1)\ \mathrm{S} \arrow[d, no head] \arrow[dll, no head] \\ (\lambda x : \mathbf{N} \to \mathbf{N}.\, x\, 1)\ \mathrm{S} & & (\lambda x : \mathbf{N} \to (? \to ?).\, x\, 1)\ \mathrm{S} \end{tikzcd}$$

safe $\uparrow$

unsafe $\downarrow$

Syntax and typing

$$T := \mathbf{B} \mid \mathbf{N} \mid T \to T$$

$$t := x \mid \underline{n} \mid \underline{b} \mid \lambda\, x : T.t \mid t\, t \mid t + t \mid \text{if } t \,\text{then } t \,\text{else } t$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda\, x : T_1.t : T_1 \to T_2}$$

$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : T_2 \qquad \Gamma \vdash t_3 : T_3 \qquad T_1 = \mathbf{B} \qquad T_2 = T \qquad T_3 = T}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T}$$

$$\ldots$$

ENS DE LYON

LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Updated system

$$T := \mathbf{B} \mid \mathbf{N} \mid T \to T \mid ?$$

$$t := x \mid \underline{n} \mid \underline{b} \mid \lambda x : T.t \mid t\,t \mid t + t \mid \text{if } t \text{ then } t \text{ else } t$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda x : T_1.t : T_1 \to T_2}$$

$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : T_2 \qquad \Gamma \vdash t_3 : T_3 \qquad T_1 \sim \mathbf{B} \qquad T_2 \sim T \qquad T_3 \sim T}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T}$$

$$\ldots$$

ENS DE LYON

LS2N LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Updated system

$$T := \mathbf{B} \mid \mathbf{N} \mid T \to T \mid ?$$

$$t := x \mid \underline{n} \mid \underline{b} \mid \lambda\, x :\, T.t \mid t\; t \mid t + t \mid \text{if } t \text{ then } t \text{ else } t$$

$$\frac{\Gamma, x :\, T_1 \vdash t :\, T_2}{\Gamma \vdash \lambda\, x :\, T_1.t :\, T_1 \to T_2}$$

$$\frac{\Gamma \vdash t_1 \,:\, T_1 \qquad \Gamma \vdash t_2 \,:\, T_2 \qquad \Gamma \vdash t_3 \,:\, T_3 \qquad T_1 \sim \mathbf{B} \qquad T_2 \sim T \qquad T_3 \sim T}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \,:\, T}$$

$$\ldots$$

Consistency

$$? \sim (\mathbf{N} \to ?) \qquad \mathbf{N} \not\sim \mathbf{B} \qquad (\mathbf{N} \to ?) \sim (? \to \mathbf{B}) \qquad ((? \to ?) \to ?) \not\sim (\mathbf{N} \to ?)$$

ENS DE LYON

LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Subject reduction is broken

$$\vdash (\lambda x : ?\, . x + 1)\ \mathrm{true} : \mathbf{N}$$

$$(\lambda x : ?\, . x + 1)\ \mathrm{true} \mapsto \mathrm{true} + 1$$

$$\nvdash \mathrm{true} + 1$$

ENS DE LYON

LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Subject reduction is broken

$$\vdash (\lambda x : ?\,.\,x + 1)\ \mathrm{true} : \mathbf{N}$$

$$(\lambda x : ?\,.\,x + 1)\ \mathrm{true} \mapsto \mathrm{true} + 1$$

$$\nvdash \mathrm{true} + 1$$

Where are the dynamic checks?

Subject reduction is broken

$$\vdash (\lambda x : ?\,.\,x + 1)\ \mathrm{true} : \mathbf{N}$$

$$(\lambda x : ?\,.\,x + 1)\ \mathrm{true} \mapsto \mathrm{true} + 1$$

$$\nvdash \mathrm{true} + 1$$

Where are the dynamic checks?

Cast calculus

$$\vdash (\lambda x : ?\,.\,x + 1)\ \mathrm{true} \rightsquigarrow (\lambda x : ?\,.(\mathrm{cast}_{?,\mathbf{N}}\ x) + 1)\ (\mathrm{cast}_{\mathbf{B},?}\ \mathrm{true})$$

Subject reduction is broken

$$\vdash (\lambda x : ?\,.x + 1)\ \mathrm{true} : \mathbf{N}$$

$$(\lambda x : ?\,.x + 1)\ \mathrm{true} \mapsto \mathrm{true} + 1$$

$$\nvdash \mathrm{true} + 1$$

Where are the dynamic checks?

Cast calculus

$$\vdash (\lambda x : ?\,.x + 1)\ \mathrm{true} \rightsquigarrow (\lambda x : ?\,.(\mathrm{cast}_{?,\mathbf{N}}\ x) + 1)\ (\mathrm{cast}_{\mathbf{B},?}\ \mathrm{true})$$

$$(\lambda x : ?\,.(\mathrm{cast}_{?,\mathbf{N}}\ x) + 1)\ (\mathrm{cast}_{\mathbf{B},?}\ \mathrm{true}) \mapsto (\mathrm{cast}_{?,\mathbf{N}}(\mathrm{cast}_{\mathbf{B},?}\ \mathrm{true})) + 1$$

$$(\mathrm{cast}_{?,\mathbf{N}}(\mathrm{cast}_{\mathbf{B},?}\ \mathrm{true})) + 1 \mapsto (\mathrm{cast}_{\mathbf{B},\mathbf{N}}\ \mathrm{true}) + 1 \mapsto \mathrm{raise}$$

Calculus of Inductive Constructions

- higher order, dependently typed extension of λ-calculus

$$\frac{\Gamma, x : A \vdash t : B \qquad \Gamma \vdash \Pi\, x : A.B : \square_i}{\Gamma \vdash \lambda\, x : A.t : \Pi\, x : A.B}$$

- with a conversion rule

$$\frac{\Gamma \vdash t : A \qquad A \equiv B \qquad \Gamma \vdash B : \square_i}{\Gamma \vdash t : B}$$

New base types, with three components:

- type (family)

$$\text{Vect} : \Pi\, A : \square, n : \mathbf{N}.\square$$

- constructors

$$\text{nil} : \Pi\, A : \square.\text{Vect } A\ 0 \qquad \text{cons} : \Pi\, A : \square, n : \mathbf{N}, a : A, v : \text{Vect } A\ n.\text{Vect } A\ (\text{S } n)$$

- recursion/induction principle

$$\text{rec}_{\text{Vect}} : \Pi\, A : \square, P : (\Pi\, n : \mathbf{N}.\text{Vect } A\ n \to \square).(P\, 0\, (\text{nil } A)) \to$$

$$(\Pi\, n : \mathbf{N}, a : A, v : \text{Vect } A\ n.P\ n\ v \to P\, (\text{S } n)\ (\text{cons } A\ n\ a\ v)) \to$$

$$\Pi\, n : \mathbf{N}, v : \text{Vect } A\ n.P\ n\ v$$

And reduction rules

$$\text{rec}_{\text{Vect}}\ A\ P\ t_{\text{nil}}\ t_{\text{cons}}\ (\text{nil } A) \equiv t_{\text{nil}}$$

$$\text{rec}_{\text{Vect}}\ A\ P\ t_{\text{nil}}\ t_{\text{cons}}\ (\text{cons } A\ n\ a\ v) \equiv t_{\text{cons}}\ n\ a\ v\, (\text{rec}_{\text{Vect}}\ A\ P\ t_{\text{nil}}\ t_{\text{cons}}\ v)$$

Gradual CIC

- in CIC, computation happens during typing : $\dfrac{\Gamma \vdash t : A \qquad A \equiv B \qquad \Gamma \vdash B : \square_i}{\Gamma \vdash t : B}$

- dependent types: ? is everywhere

- natural consistency is HO unification:
  Consistency definition $\not\longleftrightarrow$ Decidable typing

- errors: tricky in CIC

- cast definition: what about inductive types? $\mathrm{cast}_{\mathrm{Id}_A\,x\,x', \mathrm{Id}_A\,y\,y'}$?

Computation at typing

Separate conversions:

typing time simple and good enough: ? is just a constant

runtime fancy: errors, casting, etc.

Computation at typing

Separate conversions:

typing time simple and good enough: ? is just a constant

runtime fancy: errors, casting, etc.

? at any type

Let's just deal with it...

Computation at typing

Separate conversions:

typing time simple and good enough: ? is just a constant

runtime fancy: errors, casting, etc.

? at any type

Let's just deal with it...

Undecidability of ideal consistency

- approximation
- design choice, with axiomatic approach
- open for further investigation

Still handleable though

Computation at typing

Separate conversions:

typing time simple and good enough: ? is just a constant

runtime fancy: errors, casting, etc.

? at any type

Let's just deal with it...

Undecidability of ideal consistency

- approximation
- design choice, with axiomatic approach
- open for further investigation

Errors, type recursion

Syntactical models, brand new or heavy tweaking of existing ideas

ENS DE LYON

LS2N LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

$$\begin{array}{c}
\text{GCIC} \\
\downarrow \text{1: replacing consistency with casting} \\
\text{CIC} + \mathbf{cast} + ? \\
\downarrow \text{2: encoding of inductives with equality} \\
\text{CIdC} + \mathbf{cast} + ? \\
\downarrow \text{3: realizing } \mathbf{cast} \text{ and } ? \\
\text{Exceptional Type Theory} + \mathbf{quote} \\
\downarrow \text{4: syntactical model} \\
\text{CIC} + \text{Induction-Recursion}
\end{array}$$

ENS DE LYON

LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Perspectives

Playing around

Aim: sanity check, testing...

- reduction of Ω and symptomatic terms
- handling of indexes: $\mathrm{Vect}\ A\ n$
- compare consistency choices
- ideally, implementation

Ongoing work we could benefit from

- errors & effects in CIC
- quoting mechanism for Coq

ENS DE LYON

LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Thank you!

Questions?

Reduction of Ω

We write $?_i$ for $?\ \square_i$. We first get

$$\vdash \Omega : ?_{i+1} \rightsquigarrow (\lambda\, x : ?_{i+1} \,.(\mathrm{cast}_{?_{i+1}, ?_{i+1} \to ?_{i+1}}\ x)\ x)\ (\mathrm{cast}_{?_i \to ?_i, ?_{i+1}} (\lambda\, x : ?_i \,.\, \mathrm{cast}_{?_i, ?_i \to ?_i}\ x\, x))$$

For readability, we set

$$\delta_i := \lambda\, x : ?_i \,.(\mathrm{cast}_{?_i, ?_i \to ?_i}\ x)\ x$$

and

$$\Omega_i := \delta_{i+1}\ (\mathrm{cast}_{?_i \to ?_i, ?_{i+1}}\ \delta_i)$$

The reduction now gives

$$\begin{aligned}
\Omega_i \quad & \mapsto^* \quad (\mathrm{cast}_{?_{i+1}, ?_{i+1} \to ?_{i+1}}\ \mathrm{cast}_{?_i \to ?_i, ?_{i+1}}\ \delta_i)\ (\mathrm{cast}_{?_i \to ?_i, ?_{i+1}}\ \delta_i) \\
& \mapsto^* \quad (\mathrm{cast}_{?_i \to ?_i, ?_{i+1} \to ?_{i+1}}\ \delta_i)\ (\mathrm{cast}_{?_i \to ?_i, ?_{i+1}}\ \delta_i) \\
& \mapsto^* \quad (\mathrm{cast}_{?_i \to ?_i, ?_{i+1} \to ?_{i+1}} (\lambda\, x : ?_i \,.(\mathrm{cast}_{?_i, ?_i \to ?_i}\ x)\ x))\ (\mathrm{cast}_{?_i \to ?_i, ?_{i+1}}\ \delta_i) \\
& \mapsto^* \quad (\lambda\, x : ?_{i+1} \,.\, \mathrm{cast}_{?_i, ?_{i+1}}\ ((\mathrm{cast}_{?_i, ?_i \to ?_i}\ \mathrm{cast}_{?_{i+1}, ?_i}\ x)\ \mathrm{cast}_{?_{i+1}, ?_i}\ x))(\dots) \\
& \mapsto^* \quad (\lambda\, x : ?_{i+1} \,.\, \mathrm{cast}_{?_i, ?_{i+1}} ((\mathrm{cast}_{?_i, ?_i \to ?_i} (?\ \ ?_i))\ (?\ \ ?_i)))(\mathrm{cast}_{?_i \to ?_i, ?_{i+1}}\ \delta_i) \\
& \mapsto^* \quad ?\ \ (?_{i+1})
\end{aligned}$$