Gradualizing the Calculus of Inductive Constructions

Meven Lennon-Bertrand, Kenji Maillard, Nicolas Tabareau, and Éric Tanter

TOPLAS Journal-First @ POPL '22

A bit of context

The Calculus of Inductive Constructions

The Calculus of Inductive Constructions

Dependent types:

```
head: Π (A: Type) (n: ℕ), Vect A (1+n) → A
```

The Calculus of Inductive Constructions

Dependent types:

```
head: Π (A: Type) (n: ℕ), Vect A (1+n) → A
```

Bringing together CIC and gradual typing

The Calculus of Inductive Constructions

Dependent types:

```
head: Π (A: Type) (n: ℕ), Vect A (1+n) → A
```

Gradual typing

- Mix static and dynamic typing
- Dynamic type ? in a static system
- Optimistic typing phase & runtime checks

Bringing together CIC and gradual typing

The Calculus of Inductive Constructions

Dependent types:

```
head: Π (A: Type) (n: ℕ), Vect A (1+n) → A
```

THEOREM PROVER

Agda

Gradual typing

- Mix static and dynamic typing
- Dynamic type ? in a static system
- Optimistic typing phase & runtime checks

Time has come for Gradual CIC!

Why gradual dependent types?

Why gradual dependent types?

Incremental development!

Why gradual dependent types?

Incremental development!

```
filter (A : Type) (p : A → 𝔹) (n : ℕ) (l : Vect A n)
  : Vect A …
```

Incremental development!

```
filter (A : Type) (p : A → 𝔹) (n : ℕ) (l : Vect A n)
  : Vect A ?
```

Incremental development!

```
filter (A : Type) (p : A → 𝔹) (n : ℕ) (l : Vect A n)
  : Vect A ?
```

```
head nat ? (filter nat even 3 [1;2;4])
head nat ? (filter nat even 0 [])
```

Why gradual dependent types?

Incremental development!

```
filter (A : Type) (p : A → 𝔹) (n : ℕ) (l : Vect A n)
  : Vect A ?
```

```
head nat ? (filter nat even 3 [1;2;4]) ↦ 2
head nat ? (filter nat even 0 []) ↦ err
```

Why gradual dependent types?

Incremental development!

```
filter (A : Type) (p : A → 𝔹) (n : ℕ) (l : Vect A n)
  : Vect A (count A p n l)


count A p n l := ?
```

```
head nat ? (filter nat even 3 [1;2;4]) ↦ 2
head nat ? (filter nat even 0 []) ↦ err
```

Why gradual dependent types?

Incremental development!

```
filter (A : Type) (p : A → 𝔹) (n : ℕ) (l : Vect A n)
  : Vect A (count A p n l)
```

```
count A p n l :=
match l with
  | nil _ _ ⇒ 0
  | cons _ _ _ ⇒ ?
end.
```

```
head nat ? (filter nat even 3 [1;2;4]) ↦ 2
head nat ? (filter nat even 0 []) ↦ err
```

Incremental development!

```
filter (A : Type) (p : A → 𝔹) (n : ℕ) (l : Vect A n)
  : Vect A (count A p n l)
```

```
count A p n l :=
match l with
  | nil _ _ ⇒ 0
  | cons _ _ _ ⇒ ?
end.
```

```
head nat ? (filter nat even 3 [1;2;4]) ↦ 2
```

~~head nat ? (filter nat even 0 [])~~

What's so hard with dependent types?

What's so hard with dependent types?

Mixing everything

- No type/term distinction
- No typing/runtime separation

What's so hard with dependent types?

Mixing everything

- No type/term distinction
- No typing/runtime separation

Effects

- Errors
- Divergence

What's so hard with dependent types?

Mixing everything

- No type/term distinction
- No typing/runtime separation

Effects

- Errors
- Divergence

Indexed inductive types

- Vectors need special care
- Equality is currently out of our scope

An overview of gradual CIC

No-go: the fire triangle of graduality

No-go: the fire triangle of graduality

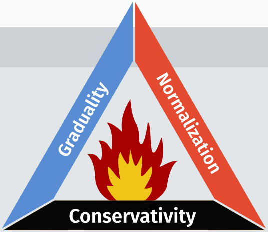
Our favorite properties

- Safety (Progress + Preservation)
- Normalization
- Conservativity (wrt. CIC)
- Graduality

No-go: the fire triangle of graduality

Our favorite properties

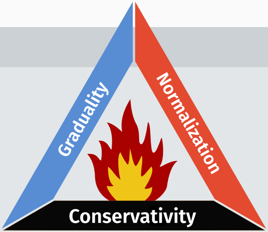
- Safety (Progress + Preservation)
- Normalization
- Conservativity (wrt. CIC)
- Graduality



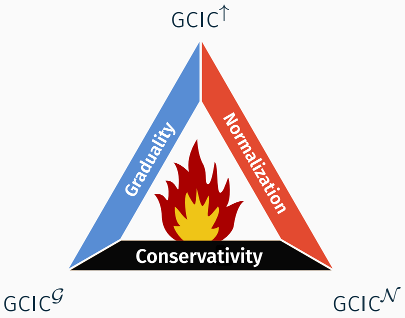
No-go: the fire triangle of graduality

Our favorite properties

- Safety (Progress + Preservation)
- Normalization
- Conservativity (wrt. cic)
- Graduality



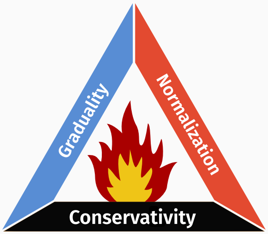
Safety + Conservativity + Graduality ⇒ Pure λ-calculus ⇒ Divergence


GCIC$^{\uparrow}$
Graduality
Normalization
Conservativity
GCIC$^{\mathcal{G}}$
GCIC$^{\mathcal{N}}$

Controlled by universe levels of Π types:

- at reduction (cast decomposition)
- at typing

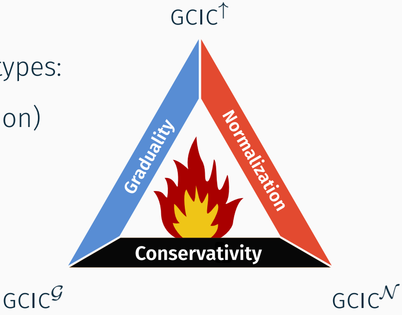
$GCIC^{\uparrow}$



$GCIC^{\mathcal{G}}$ $GCIC^{\mathcal{N}}$

Controlled by universe levels of Π types:

- at reduction (cast decomposition)
- at typing



		GCIC$^{\mathcal{G}}$	GCIC$^{\mathcal{N}}$	GCIC$^{\uparrow}$
Reduction	Pure λ-calculus	×		
	Eager failure		×	×
Typing	CIC	×	×	
	Restricted			×

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T}$$

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T}$$

Issues

- transitivity

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T}$$

Issues

- transitivity

Solutions

- bidirectional typing

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T} \qquad \frac{\Gamma \vdash t \triangleright S \qquad S \equiv T}{\Gamma \vdash t \triangleleft T}$$

Issues

- transitivity

Solutions

- bidirectional typing

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T} \qquad \frac{\Gamma \vdash t \triangleright S \qquad S \equiv T}{\Gamma \vdash t \triangleleft T}$$

Issues

- transitivity
- computation needs checks

Solutions

- bidirectional typing

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T} \qquad \frac{\Gamma \vdash t \triangleright S \qquad S \equiv T}{\Gamma \vdash t \triangleleft T}$$

Issues

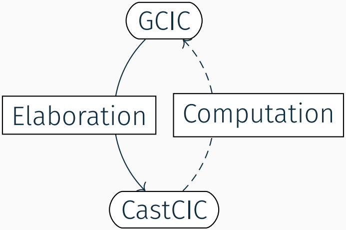
- transitivity
- computation needs checks

Solutions

- bidirectional typing
- type-based elaboration

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T} \qquad \frac{\Gamma \vdash t \triangleright S \qquad S \equiv T}{\Gamma \vdash t \triangleleft T}$$



Issues

- transitivity
- computation needs checks

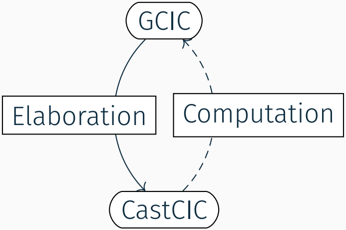
Solutions

- bidirectional typing
- type-based elaboration

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T} \qquad \frac{\Gamma \vdash t \triangleright S \qquad S \equiv T}{\Gamma \vdash t \triangleleft T}$$

$$\frac{\Gamma \vdash t \rightsquigarrow t' \triangleright S \qquad S \sim T}{\Gamma \vdash t \triangleleft T \rightsquigarrow \langle T \Leftarrow S \rangle\, t'}$$



Issues

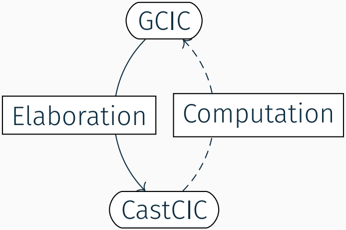
- transitivity
- computation needs checks

Solutions

- bidirectional typing
- type-based elaboration

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T} \qquad \frac{\Gamma \vdash t \triangleright S \qquad S \equiv T}{\Gamma \vdash t \triangleleft T}$$

$$\frac{\Gamma \vdash t \rightsquigarrow t' \triangleright S \qquad S \sim T}{\Gamma \vdash t \triangleleft T \rightsquigarrow \langle T \Leftarrow S \rangle\, t'}$$



Issues

- transitivity
- computation needs checks
- decidability

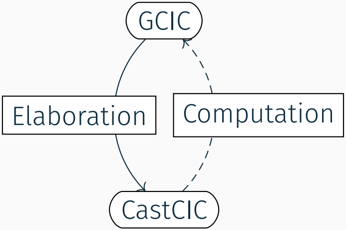
Solutions

- bidirectional typing
- type-based elaboration

Elaboration and consistent conversion

$$\frac{\Gamma \vdash t : S \qquad S \equiv T}{\Gamma \vdash t : T} \qquad \frac{\Gamma \vdash t \triangleright S \qquad S \equiv T}{\Gamma \vdash t \triangleleft T}$$

$$\frac{\Gamma \vdash t \rightsquigarrow t' \triangleright S \qquad S \sim T}{\Gamma \vdash t \triangleleft T \rightsquigarrow \langle T \Leftarrow S \rangle\, t'}$$



Issues

- transitivity
- computation needs checks
- decidability

Solutions

- bidirectional typing
- type-based elaboration
- over approximation

New terms

New terms...

GCIC: $\cdots \mid ?$

castCIC: $\cdots \mid ?_{\mathrm{T}} \mid \mathtt{err}_{\mathrm{T}} \mid \langle B \Leftarrow A \rangle\, t$

New terms

New terms...

GCIC: $\cdots \mid ?$

castCIC: $\cdots \mid ?_{\mathrm{T}} \mid \mathtt{err}_{\mathrm{T}} \mid \langle B \Leftarrow A \rangle\, t$

... and their semantics

New terms...

GCIC: $\cdots \mid ?$

castCIC: $\cdots \mid ?_{\mathrm{T}} \mid \mathtt{err}_{\mathrm{T}} \mid \langle B \Leftarrow A \rangle\, t$

... and their semantics

- $?$ elaborates to the least precise term: $\overline{\Gamma \vdash ? \rightsquigarrow ?_{?_\square} \triangleright ?_\square}$

New terms

New terms...

GCIC: $\cdots \mid ?$

CastCIC: $\cdots \mid ?_{\mathrm{T}} \mid \mathtt{err}_{\mathrm{T}} \mid \langle B \Leftarrow A \rangle\, t$

... and their semantics

- $?$ elaborates to the least precise term: $\dfrac{}{\Gamma \vdash ? \rightsquigarrow ?_{?_\square} \triangleright ?_\square}$
- $?$ and $\mathtt{err}$ as errors: $\mathtt{if}\ ?_{\mathbf{B}}\ \mathtt{return}\ T\ \mathtt{then}\ t\ \mathtt{else}\ t' \mapsto ?_T$

New terms

New terms...

GCIC: $\cdots \mid ?$

castCIC: $\cdots \mid ?_{\mathrm{T}} \mid \mathtt{err}_{\mathrm{T}} \mid \langle B \Leftarrow A \rangle\, t$

... and their semantics

- $?$ elaborates to the least precise term: $\dfrac{}{\Gamma \vdash\, ? \rightsquigarrow ?_{?_\square} \triangleright ?_\square}$
- $?$ and $\mathtt{err}$ as errors: $\mathtt{if}\ ?_{\mathbf{B}}\ \mathtt{return}\ T\ \mathtt{then}\ t\ \mathtt{else}\ t' \mapsto ?_T$
- casts compute on the type: $\langle \mathbf{N} \Leftarrow \mathbf{B} \rangle\, b \mapsto \mathtt{err}_{\mathbf{N}}$

New terms

New terms...

GCIC: $\cdots \mid ?$

castCIC: $\cdots \mid ?_{\mathrm{T}} \mid \mathtt{err}_{\mathrm{T}} \mid \langle B \Leftarrow A \rangle\, t$

... and their semantics

- $?$ elaborates to the least precise term: $\overline{\Gamma \vdash ? \rightsquigarrow ?_{?_\square} \triangleright ?_\square}$
- $?$ and $\mathtt{err}$ as errors: $\mathtt{if}\ ?_{\mathbf{B}}\ \mathtt{return}\ T\ \mathtt{then}\ t\ \mathtt{else}\ t' \mapsto ?_T$
- casts compute on the type: $\langle \mathbf{N} \Leftarrow \mathbf{B} \rangle\, b \mapsto \mathtt{err}_{\mathbf{N}}$
- $?_\square$ with casts as constructors/destructors:
  $\langle X \Leftarrow ?_\square \rangle\, \langle ?_\square \Leftarrow \mathbf{N} \rangle\, n \mapsto \langle X \Leftarrow \mathbf{N} \rangle\, n$

Theorems!

Type theory

Progress, preservation, normalization for castCIC.

Type theory

Progress, preservation, normalization for castCIC.

Proof: extension of *Sozeau et al., 2020* + translation back to CIC.

Type theory

Progress, preservation, normalization for castcic.

Proof: extension of *Sozeau et al., 2020* + translation back to cic.

Static gradual guarantee

Elaboration is monotone with respect to loss of precision.
$\Rightarrow$ *Syntactic* precision.

Type theory

Progress, preservation, normalization for castCIC.

Proof: extension of *Sozeau et al., 2020* + translation back to CIC.

Static gradual guarantee

Elaboration is monotone with respect to loss of precision.
$\Rightarrow$ *Syntactic* precision.

Key property: precision is a simulation for reduction (*Siek et al., 2015*)
$\Rightarrow$ monotony of consistency.

Type theory

Progress, preservation, normalization for castCIC.

Proof: extension of *Sozeau et al., 2020* + translation back to CIC.

Static gradual guarantee

Elaboration is monotone with respect to loss of precision.
⇒ *Syntactic* precision.

Key property: precision is a simulation for reduction (*Siek et al., 2015*)
⇒ monotony of consistency.

Conservativity

CIC embeds faithfully into GCIC.

Type theory

Progress, preservation, normalization for castCIC.

Proof: extension of *Sozeau et al., 2020* + translation back to CIC.

Static gradual guarantee

Elaboration is monotone with respect to loss of precision.
$\Rightarrow$ *Syntactic* precision.

Key property: precision is a simulation for reduction (*Siek et al., 2015*)
$\Rightarrow$ monotony of consistency.

Conservativity

CIC embeds faithfully into GCIC.

Proof: on CIC consistency = conversion, using simulation again.

The semantic theorems

Dynamic gradual guarantee

Reduction is monotone with respect to loss of precision.

Dynamic gradual guarantee

Reduction is monotone with respect to loss of precision.

Allows for degenerate computation $t \mapsto$? (*Eremondi et al., 2019*).

Dynamic gradual guarantee

Reduction is monotone with respect to loss of precision.

Allows for degenerate computation $t \mapsto$? (*Eremondi et al., 2019*).

Graduality (*New and Ahmed, 2018*)

If $A \sqsubseteq B$ then $\langle A \Leftarrow B \rangle$ and $\langle B \Leftarrow A \rangle$ form an embedding-projection pair.
$\Rightarrow$ *Semantic* precision.

Dynamic gradual guarantee

Reduction is monotone with respect to loss of precision.

Allows for degenerate computation $t \mapsto$? (*Eremondi et al., 2019*).

Graduality (*New and Ahmed, 2018*)

If $A \sqsubseteq B$ then $\langle A \Leftarrow B \rangle$ and $\langle B \Leftarrow A \rangle$ form an embedding-projection pair.
$\Rightarrow$ *Semantic* precision.

Proof: build order models to interpret precision.

Dynamic gradual guarantee

Reduction is monotone with respect to loss of precision.

Allows for degenerate computation $t \mapsto ?$ (*Eremondi et al., 2019*).

Graduality (*New and Ahmed, 2018*)

If $A \sqsubseteq B$ then $\langle A \Leftarrow B \rangle$ and $\langle B \Leftarrow A \rangle$ form an embedding-projection pair.
$\Rightarrow$ *Semantic* precision.

Proof: build order models to interpret precision. 

That's it for today

Take-away

A quick summary

- No go: fire triangle of graduality
- GCIC: one system, three variants
- Type-based elaboration to CastCIC
- Safety and normalization
- Conservativity, static gradual guarantee and graduality
- Equality: still in progress

Thank you!