A Reasonably Gradual Dependent Type Theory

Kenji Maillard

Inria Nantes, team Gallinette
j.w.w.

Meven Lennon-Bertrand | Nicolas Tabareau | Éric Tanter

ICFP'22, Ljubljana
Monday, 12th of September, 2022

Prototyping with dependent types


```
Inductive vect (A : Type) : ℕ → Type :=
| nil : vect A 0
| _::_ {n : ℕ} (head : A) (tail : vect A n) : vect A (1+n)
```

Prototyping with dependent types


```
Inductive vect (A : Type) : ℕ → Type :=
| nil : vect A 0
| _::_ {n : ℕ} (head : A) (tail : vect A n) : vect A (1+n)
```

```
Equations filter {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len) : vect A ? :=
```

Prototyping with dependent types


```
Inductive vect (A : Type) : ℕ → Type :=
| nil : vect A 0
| _::_ {n : ℕ} (head : A) (tail : vect A n) : vect A (1+n)
```

```
Equations filter {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len) : vect A ? :=
filter P nil := nil
filter P (head :: tail) when not (P head) := filter tail
filter P (head :: tail) := head :: (filter tail) : vect A ?
```

Prototyping with dependent types


```
Inductive vect (A : Type) : ℕ → Type :=
| nil : vect A 0
| _::_ {n : ℕ} (head : A) (tail : vect A n) : vect A (1+n)
```

```
Equations filter {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len) : vect A ? :=
filter P nil := nil
filter P (head :: tail) when not (P head) := filter tail
filter P (head :: tail) := head :: (filter tail) : vect A ?
```

$$\underbrace{\texttt{head :: (filter tail)}}_{\texttt{vect A (1+}\boxed{?}\texttt{)}} : \texttt{vect A } \boxed{?}$$

Prototyping with dependent types


```
Inductive vect (A : Type) : ℕ → Type :=
| nil : vect A 0
| _::_ {n : ℕ} (head : A) (tail : vect A n) : vect A (1+n)
```

```
Equations filter {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len) : vect A ? :=
filter P nil := nil
filter P (head :: tail) when not (P head) := filter tail
```

$$\texttt{filter P (head :: tail)} := \underbrace{\texttt{head :: (filter tail)}}_{\texttt{vect A } (1+\boxed{?})} : \texttt{vect A } \boxed{?}$$

$$1 + \boxed{?} \sim \boxed{?}$$

Prototyping with dependent types

1

```
head: forall (A: Type) (n: ℕ), vect A (1+n) → A

head ℕ ? (filter even (1 :: 2 :: 4 :: nil))
head ℕ ? (filter even nil)
```

```
Equations filter {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len) : vect A ? :=
filter P nil := nil
filter P (head :: tail) when not (P head) := filter tail
```

$$\texttt{filter P (head :: tail)} := \underbrace{\texttt{head :: (filter tail)}}_{\texttt{vect A (1+?)}} : \texttt{vect A ?}$$

$$1 + ? \sim ?$$

Prototyping with dependent types

1

```
head: forall (A: Type) (n: ℕ), vect A (1+n) → A

head ℕ ? (filter even (1 :: 2 :: 4 :: nil))
head ℕ ? (filter even nil)
```

vect A (1+?)

```
Equations filter {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len) : vect A ? :=
filter P nil := nil
filter P (head :: tail) when not (P head) := filter tail
filter P (head :: tail) := head :: (filter tail) : vect A ?
```

vect A (1+?)

$1 + ? \sim ?$

Prototyping with dependent types

head: forall (A: Type) (n: ℕ), vect A (1+n) → A

head ℕ ? (filter even (1 :: 2 :: 4 :: nil)) ⇝ head ℕ ? (2 :: 4 :: nil)
head ℕ ? (filter even nil) ⇝ head ℕ ? nil

Equations filter {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len) : vect A ? :=
filter P nil := nil
filter P (head :: tail) when not (P head) := filter tail
filter P (head :: tail) := $\underbrace{\texttt{head :: (filter tail)}}_{\texttt{vect A (1+?)}}$: vect A ?

$1 + ? \sim ?$

Prototyping with dependent types

1

$\mathtt{head}$: $\mathtt{forall}$ $(\mathtt{A}\colon \mathtt{Type})$ $(\mathtt{n}\colon \mathbb{N})$, $\mathtt{vect}\ \mathtt{A}\ (1{+}\mathtt{n}) \to \mathtt{A}$

$\mathtt{head}\ \mathbb{N}\ \boxed{?}\ (\mathtt{filter}\ \mathtt{even}\ (1 :: 2 :: 4 :: \mathtt{nil})) \leadsto \mathtt{head}\ \mathbb{N}\ \boxed{?}\ (2 :: 4 :: \mathtt{nil}) \leadsto 2$

$\mathtt{head}\ \mathbb{N}\ \boxed{?}\ (\mathtt{filter}\ \mathtt{even}\ \mathtt{nil}) \leadsto \mathtt{head}\ \mathbb{N}\ \boxed{?}\ \mathtt{nil}$

$\mathtt{Equations}\ \mathtt{filter}\ \{\mathtt{A} : \mathtt{Type}\}\ (\mathtt{P} : \mathtt{A} \to \mathbb{B})\ \{\mathtt{len} : \mathbb{N}\}\ (\mathtt{l} : \mathtt{vect}\ \mathtt{A}\ \mathtt{len}) : \mathtt{vect}\ \mathtt{A}\ \boxed{?} :=$

$\mathtt{filter}\ \mathtt{P}\ \mathtt{nil} := \mathtt{nil}$

$\mathtt{filter}\ \mathtt{P}\ (\mathtt{head} :: \mathtt{tail})\ \mathtt{when}\ \mathtt{not}\ (\mathtt{P}\ \mathtt{head}) := \mathtt{filter}\ \mathtt{tail}$

$\mathtt{filter}\ \mathtt{P}\ (\mathtt{head} :: \mathtt{tail}) := \underbrace{\mathtt{head} :: (\mathtt{filter}\ \mathtt{tail})}_{\mathtt{vect}\ \mathtt{A}\ (1+\boxed{?})} : \mathtt{vect}\ \mathtt{A}\ \boxed{?}$

$1 + \boxed{?} \sim \boxed{?}$

Prototyping with dependent types

1

```
head: forall (A: Type) (n: ℕ), vect A (1+n) → A
```

```
head ℕ ? (filter even (1 :: 2 :: 4 :: nil))   ⇝ head ℕ ? (2 :: 4 :: nil)  ⇝ 2
head ℕ ? (filter even nil)                     ⇝ head ℕ ? nil              ⇝ err
```

```
Equations filter {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len) : vect A ? :=
filter P nil := nil
filter P (head :: tail) when not (P head) := filter tail
```

$\texttt{filter P (head :: tail) :=} \underbrace{\texttt{head :: (filter tail)}}_{\texttt{vect A (1+?)}} \texttt{ : vect A ?}$

$1 + ? \sim ?$

Prototyping with dependent types

$\texttt{head}: \texttt{forall}\ (\texttt{A}: \texttt{Type})\ (\texttt{n}: \mathbb{N}),\ \texttt{vect A}\ (1+\texttt{n}) \to \texttt{A}$

$\texttt{head}\ \mathbb{N}\ \boxed{?}\ (\texttt{filter even}\ (1 :: 2 :: 4 :: \texttt{nil})) \rightsquigarrow \texttt{head}\ \mathbb{N}\ \boxed{?}\ (2 :: 4 :: \texttt{nil}) \rightsquigarrow 2$

$\cancel{\texttt{head}\ \mathbb{N}\ \boxed{?}\ (\texttt{filter even nil})} \rightsquigarrow \cancel{\texttt{head}\ \mathbb{N}\ \boxed{?}\ \texttt{nil}} \rightsquigarrow \cancel{\boxed{\texttt{err}}}$

$\texttt{Equations filter}\ \{\texttt{A} : \texttt{Type}\}\ (\texttt{P} : \texttt{A} \to \mathbb{B})\ \{\texttt{len} : \mathbb{N}\}\ (\texttt{l} : \texttt{vect A len}) : \texttt{vect A}\ \underbrace{\cancel{\boxed{?}}}_{\texttt{if l = nil then 0 else}\ \boxed{?}} :=$

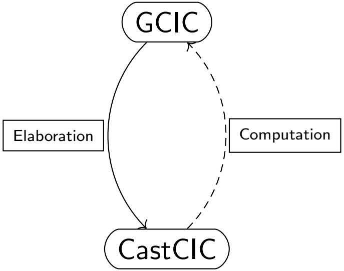
$\texttt{filter P nil} := \texttt{nil}$

$\texttt{filter P}\ (\texttt{head} :: \texttt{tail})\ \texttt{when not}\ (\texttt{P head}) := \texttt{filter tail}$

$\texttt{filter P}\ (\texttt{head} :: \texttt{tail}) := \underbrace{\texttt{head} :: (\texttt{filter tail})}_{\texttt{vect A}\ (1+\boxed{?})} : \texttt{vect A}\ \boxed{?}$

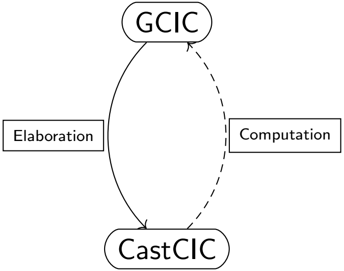
$1 + \boxed{?} \sim \boxed{?}$

GCIC & CastCIC: a marriage of Gradual & Dependent Types



- Unknown terms and types $?_A : A$
- Errors $\mathtt{err}_A : A$
- Casts $t : A \Longrightarrow \langle B \Leftarrow A \rangle\, t : B$

GCIC & CastCIC: a marriage of Gradual & Dependent Types

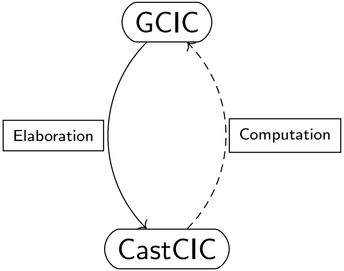


Precision between types

$$\mathbb{N} \to \mathbb{N} \sqsubseteq \mathbb{N} \to ? \sqsubseteq ? \to ? \sqsubseteq ?$$

- Unknown terms and types $?_A : A$
- Errors $\mathtt{err}_A : A$
- Casts $t : A \implies \langle B \Leftarrow A \rangle\, t : B$

GCIC & CastCIC: a marriage of Gradual & Dependent Types



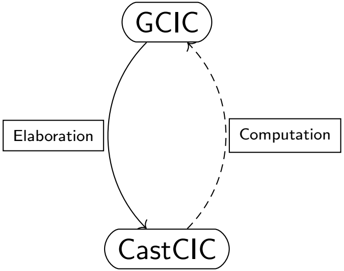
Precision between types

maximal

$$\mathbb{N}\to\mathbb{N} \sqsubseteq \mathbb{N}\to ? \sqsubseteq ?\to ? \sqsubseteq ?$$

- Unknown terms and types $?_A : A$
- Errors $\texttt{err}_A : A$
- Casts $t : A \implies \langle B \Leftarrow A \rangle\, t : B$

GCIC & CastCIC: a marriage of Gradual & Dependent Types



- Unknown terms and types $?_A : A$
- Errors $\mathtt{err}_A : A$
- Casts $t : A \implies \langle B \Leftarrow A \rangle\, t : B$

Precision between types

maximal

$$\mathbb{N}\to\mathbb{N} \;\sqsubseteq\; \mathbb{N}\to ? \;\sqsubseteq\; ?\to ? \;\sqsubseteq\; ?$$

Precision enforces cast validity:

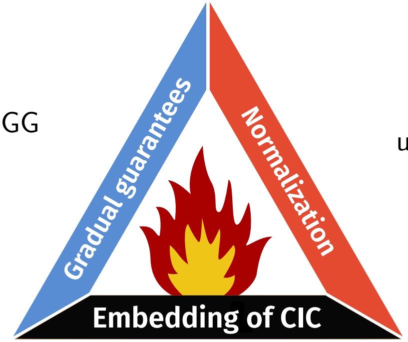
$$A \sqsubseteq B \implies \begin{tikzcd} A \arrow[r, bend left, "\langle B \Leftarrow A\rangle"] & B \arrow[l, bend left, "\langle A \Leftarrow B\rangle"] \end{tikzcd}$$

Graduality [New & Ahmed 2018]

The Fire Triangle of Graduality

From Gradualizing the CIC [TOPLAS 2022]

Graduality
Static & Dynamic GG



$\mathbf{?}_{\square} \to \mathbf{?}_{\square} \not\sqsubseteq \mathbf{?}_{\square}$
No embedding of
untyped λ-calculus

Conservativity: t, A $?$-free

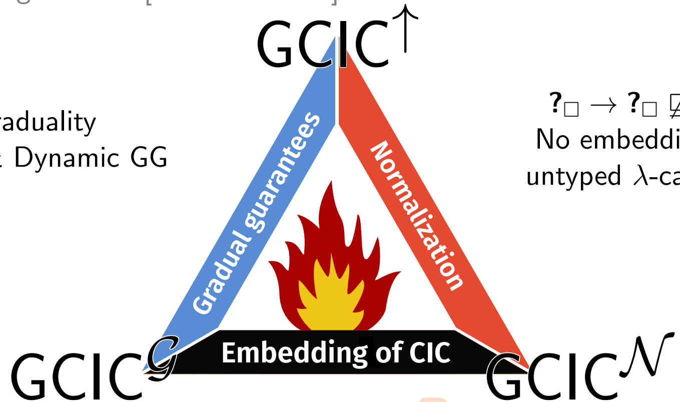
t has type A in GCIC iff t has type A in CIC

$\vdash_{\mathbf{?}} t : A \iff \vdash_{\mathrm{CIC}} t : A$

The Fire Triangle of Graduality

From Gradualizing the CIC [TOPLAS 2022]

Graduality
Static & Dynamic GG



$\mathbf{?}_{\square} \to \mathbf{?}_{\square} \not\sqsubseteq \mathbf{?}_{\square}$
No embedding of
untyped λ-calculus

Conservativity: t, A $\boxed{?}$-free

t has type A in GCIC iff t has type A in CIC

$\vdash_{\mathbf{?}} t : A \iff \vdash_{\mathsf{CIC}} t : A$

Internal precision in GRIP

$\mathsf{CastCIC}^{\mathcal{N}}$: Normalizing, conservative over CIC, not gradual

How far is $\mathsf{CastCIC}^{\mathcal{N}}$ from actually being gradual ?

Internal precision in GRIP

CastCIC$^{\mathcal{N}}$: Normalizing, conservative over CIC, not gradual

How far is CastCIC$^{\mathcal{N}}$ from actually being gradual ?

Make precision internal so that we can reason on it!

$$\text{GRIP} = \text{CastCIC}^{\mathcal{N}} + \sqsubseteq + \mathbb{P}$$

Internal precision in GRIP

$\text{CastCIC}^{\mathcal{N}}$: Normalizing, conservative over CIC, not gradual

$$\text{GRIP} = \text{CastCIC}^{\mathcal{N}} + \sqsubseteq + \mathbb{P}$$

$$\frac{\Gamma \vdash A, B : \square_i}{\Gamma \vdash A \sqsubseteq_i B : \mathbb{P}} \qquad \frac{\Gamma \vdash A, B : \square_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash a \ {}_{A}{\sqsubseteq}_{B}\ b : \mathbb{P}}$$

$\mathbb{P}$: Pure universe of propositions

Internal precision in GRIP

CastCIC$^{\mathcal{N}}$: Normalizing, conservative over CIC, not gradual

$$\mathrm{GRIP} = \mathrm{CastCIC}^{\mathcal{N}} + \sqsubseteq + \mathbb{P}$$

$$\frac{\Gamma \vdash A, B : \square_i}{\Gamma \vdash A \sqsubseteq_i B : \mathbb{P}} \qquad \frac{\Gamma \vdash A, B : \square_i \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash a \;{}_{A}{\sqsubseteq}_{B}\; b : \mathbb{P}}$$

$\mathbb{P}$: $\underbrace{\text{Pure}}_{\substack{\text{no error,}\\ \text{no casts,}\\ \text{no } \boxed{?}}}$ universe of $\underbrace{\text{propositions}}_{\substack{\text{definitionally}\\ \text{proof-irrelevant}}}$ for sound reasoning

Proofs of precision – External meaning

Dynamic Gradual Guarantee

$$\mathtt{DGG} : \forall (A : \square)(C : A \to \mathbb{B})(x\ y : A), x\ {}_{A}{\sqsubseteq}_{A}\ y \to C\ x\ {}_{\mathbb{B}}{\sqsubseteq}_{\mathbb{B}}\ C\ y.$$

$$\mathtt{DGG}\ A\ C \text{ holds} \quad \iff \quad C\ {}_{A\to\mathbb{B}}{\sqsubseteq}_{A\to\mathbb{B}}\ C \quad \iff \quad C \text{ monotone wrt. } \sqsubseteq$$

Proofs of precision – External meaning

Dynamic Gradual Guarantee

$$\mathtt{DGG} : \forall (A : \square)(C : A \to \mathbb{B})(x\ y : A), x\ {}_{A}{\sqsubseteq}_{A}\ y \to C\ x\ {}_{\mathbb{B}}{\sqsubseteq}_{\mathbb{B}}\ C\ y.$$

$$\mathtt{DGG}\ A\ C \text{ holds} \quad \Longleftrightarrow \quad C\ {}_{A\to\mathbb{B}}{\sqsubseteq}_{A\to\mathbb{B}}\ C \quad \Longleftrightarrow \quad C \text{ monotone wrt. } \sqsubseteq$$

$\mathtt{DGG}$ correspond to error-approximation:

$$\begin{tikzcd}
 & \mathtt{true} \arrow[dr, phantom, sloped, "\sqsubseteq"] & \\
\mathtt{err}_{\mathbb{B}} \arrow[ur, phantom, sloped, "\sqsubseteq"] \arrow[dr, phantom, sloped, "\sqsubseteq"] & & ?_{\mathbb{B}} \\
 & \mathtt{false} \arrow[ur, phantom, sloped, "\sqsubseteq"] &
\end{tikzcd}$$

Proofs of precision – An example

$$\texttt{add1} : \mathbb{N} \to \mathbb{N} := \lambda x : \mathbb{N}. x + 1$$

$$\texttt{add?} : \mathbf{?}_\square \to \mathbb{N} := \lambda x : \mathbf{?}_\square . (\langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle \, x) + 1$$

$\texttt{add1} \;{}_{\mathbb{N}\to\mathbb{N}}\!\sqsubseteq_{\mathbf{?}_\square\to\mathbb{N}}\; \texttt{add?} \quad \Longrightarrow \quad$ add? does not error on good input

Proofs of precision – An example

$$\texttt{add1} : \mathbb{N} \to \mathbb{N} := \lambda x : \mathbb{N}.x + 1$$

$$\texttt{add?} : \mathbf{?}_\square \to \mathbb{N} := \lambda x : \mathbf{?}_\square.(\langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle\, x) + 1$$

$$\texttt{add1} \;{}_{\mathbb{N}\to\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square\to\mathbb{N}}\; \texttt{add?}$$

Proofs of precision – An example

$$\texttt{add1} : \mathbb{N} \to \mathbb{N} := \lambda x : \mathbb{N}. x + 1$$

$$\texttt{add?} : \mathbf{?}_{\Box} \to \mathbb{N} := \lambda x : \mathbf{?}_{\Box}. (\langle \mathbb{N} \Leftarrow \mathbf{?}_{\Box} \rangle \, x) + 1$$

$$\frac{\forall (x : \mathbb{N})(z : \mathbf{?}_{\Box}), x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_{\Box}}\; z \quad \to \quad \texttt{add1}\; x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; \texttt{add?}\; z}{\texttt{add1} \;{}_{\mathbb{N}\to\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_{\Box}\to\mathbb{N}}\; \texttt{add?}} \quad \sqsubseteq \text{ on } \to$$

Proofs of precision – An example

$$\texttt{add1} : \mathbb{N} \to \mathbb{N} := \lambda x : \mathbb{N}. x + 1$$

$$\texttt{add?} : \mathbf{?}_\square \to \mathbb{N} := \lambda x : \mathbf{?}_\square . (\langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle\, x) + 1$$

$$\dfrac{\dfrac{x : \mathbb{N}, z : \mathbf{?}_\square, xz : x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad\vdash\quad x+1 \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; (\langle \mathbb{N} \Leftarrow \mathbf{?}_\square\rangle\, z)+1}{\forall (x:\mathbb{N})(z:\mathbf{?}_\square), x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad\to\quad \texttt{add1}\ x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; \texttt{add?}\ z}\ \text{Intros, compute}}{\texttt{add1} \;{}_{\mathbb{N}\to\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square\to\mathbb{N}}\; \texttt{add?}}\ \sqsubseteq \text{ on } \to$$

Proofs of precision – An example

$$\texttt{add1} : \mathbb{N} \to \mathbb{N} := \lambda x : \mathbb{N}. x + 1$$
$$\texttt{add?} : \mathbf{?}_\square \to \mathbb{N} := \lambda x : \mathbf{?}_\square. (\langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle\, x) + 1$$

$$\dfrac{\dfrac{\dfrac{x : \mathbb{N}, z : \mathbf{?}_\square, xz : x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad \vdash \quad x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; \langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle\, z}{x : \mathbb{N}, z : \mathbf{?}_\square, xz : x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad \vdash \quad x + 1 \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; (\langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle\, z) + 1} \; +1 \text{ monotone}}{\forall (x : \mathbb{N})(z : \mathbf{?}_\square), x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad \to \quad \texttt{add1}\ x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; \texttt{add?}\ z} \; \text{Intros, compute}}{\texttt{add1} \;{}_{\mathbb{N}\to\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square \to \mathbb{N}}\; \texttt{add?}} \; \sqsubseteq \text{ on } \to$$

Proofs of precision – An example

$$\mathtt{add1} : \mathbb{N} \to \mathbb{N} := \lambda x : \mathbb{N}. x + 1$$

$$\mathtt{add?} : \mathbf{?}_\square \to \mathbb{N} := \lambda x : \mathbf{?}_\square. (\langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle \; x) + 1$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{x : \mathbb{N}, z : \mathbf{?}_\square, xz : x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad \vdash \quad xz : x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z}}{x : \mathbb{N}, z : \mathbf{?}_\square, xz : x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad \vdash \quad x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; \langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle \; z} \; \langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle \text{ right adjoint}}{x : \mathbb{N}, z : \mathbf{?}_\square, xz : x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad \vdash \quad x + 1 \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; (\langle \mathbb{N} \Leftarrow \mathbf{?}_\square \rangle \; z) + 1} \; +1 \text{ monotone}}{\forall (x : \mathbb{N})(z : \mathbf{?}_\square), x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square}\; z \quad \to \quad \mathtt{add1}\; x \;{}_{\mathbb{N}}{\sqsubseteq}_{\mathbb{N}}\; \mathtt{add?}\; z} \; \text{Intros, compute}}{\mathtt{add1} \;{}_{\mathbb{N}\to\mathbb{N}}{\sqsubseteq}_{\mathbf{?}_\square \to \mathbb{N}}\; \mathtt{add?}} \; \sqsubseteq \text{ on } \to$$

Proofs of precision – Principles

Reasoning principles for $\sqsubseteq$

- heterogeneous transitivity

$$a \;{}_{A}{\sqsubseteq}_{B}\; b \wedge b \;{}_{B}{\sqsubseteq}_{C}\; c \rightarrow a \;{}_{A}{\sqsubseteq}_{C}\; c,$$

- quasi-reflexivity

$$a \;{}_{A}{\sqsubseteq}_{B}\; b \rightarrow a \;{}_{A}{\sqsubseteq}_{A}\; a \wedge b \;{}_{B}{\sqsubseteq}_{B}\; b,$$

- adjunction properties

$$A \sqsubseteq_i B \rightarrow \langle B \Leftarrow A \rangle\, a \;{}_{B}{\sqsubseteq}_{B}\; b \Longleftrightarrow a \;{}_{A}{\sqsubseteq}_{B}\; b \Longleftrightarrow a \;{}_{A}{\sqsubseteq}_{A}\; \langle A \Leftarrow B \rangle\, b,$$

- decomposition through upper bounds

$$A, B \sqsubseteq_i X \rightarrow a \;{}_{A}{\sqsubseteq}_{B}\; b \Longleftrightarrow \langle X \Leftarrow A \rangle\, a \;{}_{X}{\sqsubseteq}_{X}\; \langle X \Leftarrow B \rangle\, b.$$

Proofs of precision – Principles

Reasoning principles for $\sqsubseteq$

- heterogeneous transitivity

$$a \;{}_{A}{\sqsubseteq}_{B}\; b \wedge b \;{}_{B}{\sqsubseteq}_{C}\; c \to a \;{}_{A}{\sqsubseteq}_{C}\; c,$$

- quasi-reflexivity

$$a \;{}_{A}{\sqsubseteq}_{B}\; b \to a \;{}_{A}{\sqsubseteq}_{A}\; a \wedge b \;{}_{B}{\sqsubseteq}_{B}\; b,$$

- adjunction properties

$$A \sqsubseteq_i B \to \langle B \Leftarrow A \rangle\, a \;{}_{B}{\sqsubseteq}_{B}\; b \Longleftrightarrow a \;{}_{A}{\sqsubseteq}_{B}\; b \Longleftrightarrow a \;{}_{A}{\sqsubseteq}_{A}\; \langle A \Leftarrow B \rangle\, b,$$

- decomposition through upper bounds

$$A, B \sqsubseteq_i X \to a \;{}_{A}{\sqsubseteq}_{B}\; b \Longleftrightarrow \langle X \Leftarrow A \rangle\, a \;{}_{X}{\sqsubseteq}_{X}\; \langle X \Leftarrow B \rangle\, b.$$

However

- $A \;{}_{\square_i}{\sqsubseteq}_{\square_i}\; B \iff A \sqsubseteq_i B \wedge B \sqsubseteq_i \mathbf{?}_{\square_i}$

Proofs of precision – Principles

Reasoning principles for $\sqsubseteq$

- heterogeneous transitivity

$$a \;{}_{A}{\sqsubseteq}_{B}\; b \wedge b \;{}_{B}{\sqsubseteq}_{C}\; c \rightarrow a \;{}_{A}{\sqsubseteq}_{C}\; c,$$

- quasi-reflexivity

$$a \;{}_{A}{\sqsubseteq}_{B}\; b \rightarrow a \;{}_{A}{\sqsubseteq}_{A}\; a \wedge b \;{}_{B}{\sqsubseteq}_{B}\; b,$$

- adjunction properties

$$A \sqsubseteq_i B \rightarrow \langle B \Leftarrow A\rangle\, a \;{}_{B}{\sqsubseteq}_{B}\; b \Longleftrightarrow a \;{}_{A}{\sqsubseteq}_{B}\; b \Longleftrightarrow a \;{}_{A}{\sqsubseteq}_{A}\; \langle A \Leftarrow B\rangle\, b,$$

- decomposition through upper bounds

$$A, B \sqsubseteq_i X \rightarrow a \;{}_{A}{\sqsubseteq}_{B}\; b \Longleftrightarrow \langle X \Leftarrow A\rangle\, a \;{}_{X}{\sqsubseteq}_{X}\; \langle X \Leftarrow B\rangle\, b.$$

However

- $A \;{}_{\square_i}{\sqsubseteq}_{\square_i}\; B \iff A \sqsubseteq_i B \wedge B \sqsubseteq_i \mathbf{?}_{\square_i}$
- $\mathbf{?}_{\square_i} \rightarrow \mathbf{?}_{\square_i} \not\sqsubseteq_i \mathbf{?}_{\square_i}$

Proofs of precision – Principles

Reasoning principles for $\sqsubseteq$

- heterogeneous transitivity

$$a \; {}_{A}{\sqsubseteq}_{B} \; b \wedge b \; {}_{B}{\sqsubseteq}_{C} \; c \to a \; {}_{A}{\sqsubseteq}_{C} \; c,$$

- quasi-reflexivity

$$a \; {}_{A}{\sqsubseteq}_{B} \; b \to a \; {}_{A}{\sqsubseteq}_{A} \; a \wedge b \; {}_{B}{\sqsubseteq}_{B} \; b,$$

- adjunction properties

$$A \sqsubseteq_i B \to \langle B \Leftarrow A \rangle \, a \; {}_{B}{\sqsubseteq}_{B} \; b \Longleftrightarrow a \; {}_{A}{\sqsubseteq}_{B} \; b \Longleftrightarrow a \; {}_{A}{\sqsubseteq}_{A} \; \langle A \Leftarrow B \rangle \, b,$$

- decomposition through upper bounds

$$A, B \sqsubseteq_i X \to a \; {}_{A}{\sqsubseteq}_{B} \; b \Longleftrightarrow \langle X \Leftarrow A \rangle \, a \; {}_{X}{\sqsubseteq}_{X} \; \langle X \Leftarrow B \rangle \, b.$$

However

- $A \; {}_{\square_i}{\sqsubseteq}_{\square_i} \; B \iff A \sqsubseteq_i B \wedge B \sqsubseteq_i \mathbf{?}_{\square_i}$
- $\mathbf{?}_{\square_i} {\to} \mathbf{?}_{\square_i} \not\sqsubseteq_i \mathbf{?}_{\square_i}$
- $\mathbf{?}_{\square_i}$ maximal for ${}_{\square_i}{\sqsubseteq}_{\square_i}$, not for $\sqsubseteq_i$

Proofs of precision – Principles

Reasoning principles for $\sqsubseteq$

- heterogeneous transitivity

$$a \;{}_{A}{\sqsubseteq}_{B}\; b \wedge b \;{}_{B}{\sqsubseteq}_{C}\; c \to a \;{}_{A}{\sqsubseteq}_{C}\; c,$$

- quasi-reflexivity

$$a \;{}_{A}{\sqsubseteq}_{B}\; b \to a \;{}_{A}{\sqsubseteq}_{A}\; a \wedge b \;{}_{B}{\sqsubseteq}_{B}\; b,$$

- adjunction properties

$$A \sqsubseteq_i B \to \langle B \Leftarrow A \rangle\, a \;{}_{B}{\sqsubseteq}_{B}\; b \Longleftrightarrow a \;{}_{A}{\sqsubseteq}_{B}\; b \Longleftrightarrow a \;{}_{A}{\sqsubseteq}_{A}\; \langle A \Leftarrow B \rangle\, b,$$

- decomposition through upper bounds

$$A, B \sqsubseteq_i X \to a \;{}_{A}{\sqsubseteq}_{B}\; b \Longleftrightarrow \langle X \Leftarrow A \rangle\, a \;{}_{X}{\sqsubseteq}_{X}\; \langle X \Leftarrow B \rangle\, b.$$

However

- $A \;{}_{\square_i}{\sqsubseteq}_{\square_i}\; B \iff A \sqsubseteq_i B \wedge B \sqsubseteq_i \mathbf{?}_{\square_i}$
- $\mathbf{?}_{\square_i} \to \mathbf{?}_{\square_i} \not\sqsubseteq_i \mathbf{?}_{\square_i}$
- $\mathbf{?}_{\square_i}$ maximal for ${}_{\square_i}{\sqsubseteq}_{\square_i}$, not for $\sqsubseteq_i$
- Mitigated by explicity cumulativity

$$\iota(\mathbf{?}_{\square_i} \to \mathbf{?}_{\square_i}) \sqsubseteq_{i+1} \mathbf{?}_{\square_{i+1}}$$

How good is GRIP? Metatheory!

Consistency $\quad \nvdash_{\mathsf{GRIP}} t : \bot \quad$ (where $\bot : \mathbb{P}$)

Idea: Build a model of GRIP in MLTT+SProp using partial pre-orders.
$[\![\bot]\!]$ is empty in the model.

How good is GRIP? Metatheory!

Consistency $\nvdash_{\mathsf{GRIP}} t : \bot$ (where $\bot : \mathbb{P}$)

Idea: Build a model of GRIP in MLTT+SProp using partial pre-orders.
$[\![\bot]\!]$ is empty in the model.

Normalization

Idea: The translation $[\![-]\!]$ preserves reduction sequences
into a normalizing target (MLTT + SProp, Gilbert et al. [POPL'19]).

How good is GRIP? Metatheory!

Consistency $\nvdash_{\text{GRIP}} t : \bot$ (where $\bot : \mathbb{P}$)

Idea: Build a model of GRIP in MLTT+SProp using partial pre-orders.
$[\![\bot]\!]$ is empty in the model.

Normalization

Idea: The translation $[\![-]\!]$ preserves reduction sequences
into a normalizing target (MLTT + SProp, Gilbert et al. [POPL'19]).

Gradual fragment GRIP$^{\uparrow}$ Automatically precise terms

Idea: Raise universe levels on Π

$$\frac{\Gamma \vdash_{\text{GRIP}^{\uparrow}} A : \square_i \qquad \Gamma, x : A \vdash_{\text{GRIP}^{\uparrow}} B : \square_i}{\Gamma \vdash_{\text{GRIP}^{\uparrow}} \Pi x : A.B : \square_{i+1}}$$

Summary & Future work

GRIP = Partially gradual type theory + internal precision

- ▷ Proof of concept for GRIP in Agda
- ▷ Model in Coq based on partial pre-orders

Summary & Future work

GRIP = Partially gradual type theory + internal precision

- Proof of concept for GRIP in Agda
- Model in Coq based on partial pre-orders
- Non gradual operations, e.g. `catch` of `?`/`err` on inductives
- Adresses DGG in an exceptional world (see paper).

Summary & Future work

GRIP = Partially gradual type theory + internal precision

- Proof of concept for GRIP in Agda
- Model in Coq based on partial pre-orders
- Non gradual operations, e.g. `catch` of ? / `err` on inductives
- Adresses DGG in an exceptional world (see paper).
- Internal reasoning to establish graduality post-hoc

Summary & Future work

GRIP = Partially gradual type theory + internal precision

- Proof of concept for GRIP in Agda
- Model in Coq based on partial pre-orders
- Non gradual operations, e.g. `catch` of `?`/`err` on inductives
- Adresses DGG in an exceptional world (see paper).
- Internal reasoning to establish graduality post-hoc

GRIP is a type theory, not yet a proof assistant:

- Elaboration from a gradual source
- General inductive types
- An actual implementation?

Thank You!

Gradualizing `printf`


```
Equations argOfDescr (s : stringDescr) : Type :=
  argOfDescr "" := unit ;
  argOfDescr ("%d" @ rest) := ℕ → argOfDescr rest;
  ...

Definition printf : forall (s : stringDescr), argOfDescr s := ...
```

- ▷ `printf` does not type in GRIP$^{\uparrow}$ (arbitrary arity)
- ▷ `printf` does type in GRIP
- ▷ It is not self-precise
- ▷ but its retrictions to `stringDescr` bounded by some fixed length are self-precise.