Gradual Indexed Inductive Types

Kenji Maillard

Inria Nantes, team Gallinette

j.w.w.



Mara Malewski

Nicolas Tabareau

Éric Tanter

ICFP'24, Milan
Tuesday, 3rd of September, 2024

Gradual Dependent Types 101


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

Gradual Dependent Types 101


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

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

filter : forall {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len), vect A ?.
```

Gradual Dependent Types 101


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

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

filter : forall {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len), vect A ?.


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

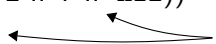
Gradual Dependent Types 101


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

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

filter : forall {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len), vect A ? .


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



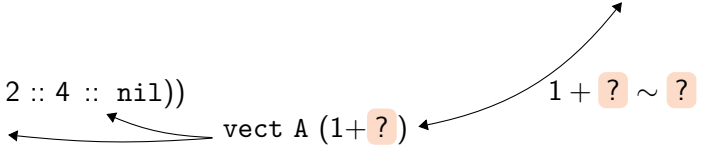
Gradual Dependent Types 101


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

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

filter : forall {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len), vect A ? .

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



Gradual Dependent Types 101


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

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

```
filter : forall {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len), vect A ?.
```

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

Gradual Dependent Types 101


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

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

```
filter : forall {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len), vect A ?.
```

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

Gradual Dependent Types 101


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

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

```
filter : forall {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len), vect A ?.
```

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

Gradual Dependent Types 101


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

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

```
filter : forall {A : Type} (P : A → 𝔹) {len : ℕ} (l : vect A len), vect A ?.
```

```
if l = nil
then 0
else ?
```

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

~~head ℕ ? (filter even nil)~~ ⇝ ~~head ℕ ? nil~~ ⇝ ~~err~~

GCIC & CastCIC: a marriage of Gradual & Dependent Types

Gradualizing the CIC [TOPLAS'22]

GCIC $\xrightarrow{\text{elaborates to}}$ CastCIC

GCIC & CastCIC: a marriage of Gradual & Dependent Types

Gradualizing the CIC [TOPLAS'22]

$$\text{GCIC} \xrightarrow{\text{elaborates to}} \text{CastCIC} \begin{cases} \texttt{?}_A : A & \text{Unknown terms and types} \\ \texttt{err}_A : A & \text{Errors} \\ \langle B \Leftarrow A \rangle\, t & \text{Casts} \end{cases}$$

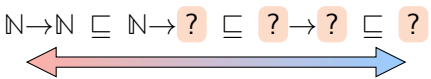
GCIC & CastCIC: a marriage of Gradual & Dependent Types

Gradualizing the CIC [TOPLAS'22]

GCIC $\xrightarrow{\text{elaborates to}}$ CastCIC

$$\begin{cases} ?_A : A & \text{Unknown terms and types} \\ \texttt{err}_A : A & \text{Errors} \\ \langle B \Leftarrow A \rangle\, t & \text{Casts} \end{cases}$$

Precision between types



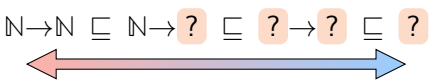
GCIC & CastCIC: a marriage of Gradual & Dependent Types

Gradualizing the CIC [TOPLAS'22]

GCIC $\xrightarrow{\text{elaborates to}}$ CastCIC

$$\begin{cases} ?_A : A & \text{Unknown terms and types} \\ \texttt{err}_A : A & \text{Errors} \\ \langle B \Leftarrow A \rangle\, t & \text{Casts} \end{cases}$$

Precision between types



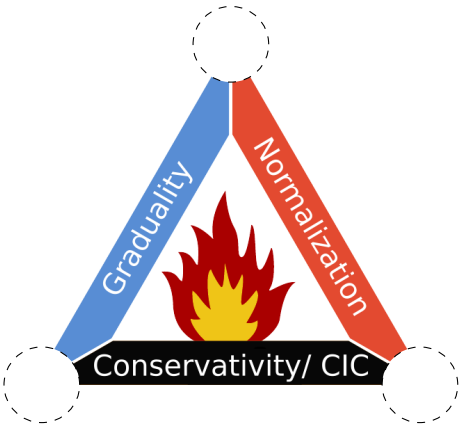
Precision enforces cast validity:

$$A \sqsubseteq B \implies A \underset{\langle A \Leftarrow B \rangle}{\overset{\langle B \Leftarrow A \rangle}{\rightleftarrows}} B$$

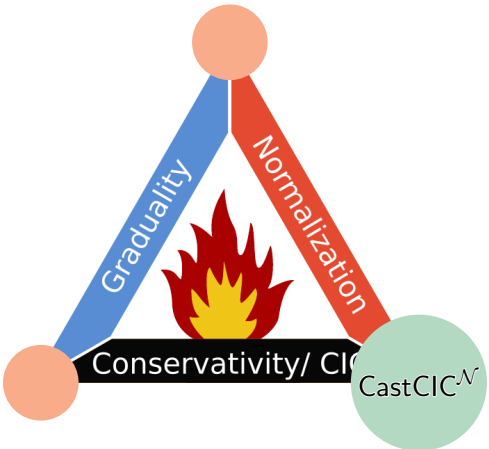
Graduality [New & Ahmed 2018]

The Fire Triangle of Graduality & GRIP

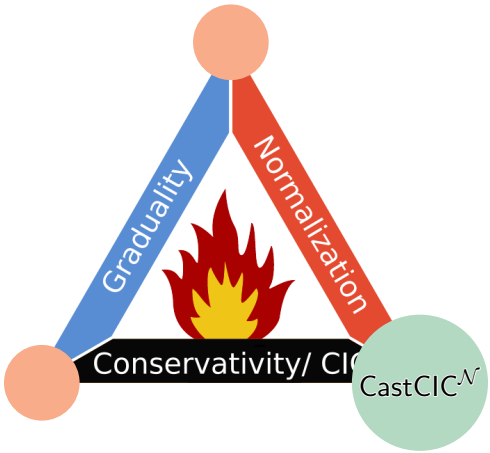
The Fire Triangle of Graduality & GRIP



The Fire Triangle of Graduality & GRIP



The Fire Triangle of Graduality & GRIP



Recovering graduality: Internal Precision

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

$$\frac{\Gamma \vdash A, B : \text{Type}_i}{\Gamma \vdash A \sqsubseteq_i B : \mathbb{P}}$$

Semi-synthetic: carving out gradual types

$$A \text{ gradual} \iff A \sqsubseteq A$$

Not all constructions preserve precision !
A Reasonably Gradual Type Theory [ICFP'22]

Gradual Inductive Families ?

- [TOPLAS'22]: Ad-hoc vectors, encodings with Subset/refinement types, Index-first

Gradual Inductive Families ?

- [TOPLAS'22]: Ad-hoc vectors, encodings with Subset/refinement types, Index-first
- *Propositional Equality for GDTP*, Eremondi et.al [ICFP'22]

$$\frac{w \sqsubseteq a \qquad w \sqsubseteq b}{\texttt{rfl}\ w\ b : \texttt{eq}\ a\ b}$$

$$\langle \texttt{eq}\ a\ b \Leftarrow \texttt{eq}\ a\ a \rangle\ \texttt{rfl}\ a\ a \rightsquigarrow \texttt{rfl}\ (a \sqcap b)\ a$$

Gradual Inductive Families ?

- [TOPLAS'22]: Ad-hoc vectors, encodings with Subset/refinement types, Index-first
- *Propositional Equality for GDTP*, Eremondi et.al [ICFP'22]

$$\frac{w \sqsubseteq a \qquad w \sqsubseteq b}{\texttt{rfl}\ w\ b : \texttt{eq}\ a\ b}$$

$$\langle \texttt{eq}\ a\ b \Leftarrow \texttt{eq}\ a\ a \rangle\ \texttt{rfl}\ a\ a \rightsquigarrow \texttt{rfl}\ (a \sqcap b)\ a$$

- GRIP: right environment to study the graduality of arbitrary families

Gradual Inductive Families ?

- ▷ [TOPLAS'22]: Ad-hoc vectors, encodings with Subset/refinement types, Index-first
- ▷ *Propositional Equality for GDTP*, Eremondi et.al [ICFP'22]

$$\frac{w \sqsubseteq a \qquad w \sqsubseteq b}{\texttt{rfl}\ w\ b : \texttt{eq}\ a\ b}$$

$$\langle \texttt{eq}\ a\ b \Leftarrow \texttt{eq}\ a\ a \rangle\ \texttt{rfl}\ a\ a \rightsquigarrow \texttt{rfl}\ (a \sqcap b)\ a$$

- ▷ GRIP: right environment to study the graduality of arbitrary families

A general treatment of Gradual Families in GRIP!

Varieties of Casts on Indices

Abbreviating $\mathbb{V} := \texttt{vect}\, A$, $\langle C \Leftarrow B \Leftarrow A \rangle := \langle C \Leftarrow B \rangle \langle B \Leftarrow A \rangle$,

$$\langle \mathbb{V}\, 0 \Leftarrow \mathbb{V}\, 1 \Leftarrow \mathbb{V}\, 0 \rangle \, \texttt{nil}$$

Varieties of Casts on Indices

Abbreviating $\mathbb{V} := \mathtt{vect}\, A$, $\langle C \Leftarrow B \Leftarrow A \rangle := \langle C \Leftarrow B \rangle \langle B \Leftarrow A \rangle$,

$$\langle \mathbb{V}\, 0 \Leftarrow \mathbb{V}\, 1 \Leftarrow \mathbb{V}\, 0 \rangle\, \mathtt{nil}$$

normal form

(Free)

Varieties of Casts on Indices

Abbreviating $\mathbb{V} := \mathtt{vect}\, A$, $\langle C \Leftarrow B \Leftarrow A \rangle := \langle C \Leftarrow B \rangle \langle B \Leftarrow A \rangle$,

$$\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\, \mathtt{nil}$$

normal form

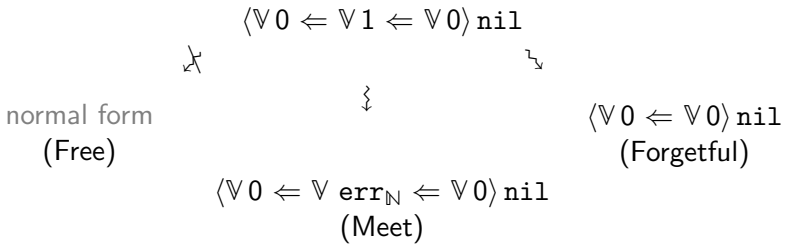
(Free)

$$\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\ \mathtt{err}_{\mathbb{N}} \Leftarrow \mathbb{V}\,0 \rangle\, \mathtt{nil}$$

(Meet)

Varieties of Casts on Indices

Abbreviating $\mathbb{V} := \mathtt{vect}\, A$, $\langle C \Leftarrow B \Leftarrow A \rangle := \langle C \Leftarrow B \rangle \langle B \Leftarrow A \rangle$,



Organizing admissible behaviors of casts

($\mathbb{V} := \mathtt{vect}\, A$)

$$
\begin{array}{rcll}
 & \not\rightsquigarrow & \text{(normal form)} & \text{(Free)} \\
\langle \mathbb{V}\, 0 \Leftarrow \mathbb{V}\, 1 \Leftarrow \mathbb{V}\, 0 \rangle\, \mathtt{nil} & \rightsquigarrow & \langle \mathbb{V}\, 0 \Leftarrow \mathbb{V}\, \mathtt{err}_{\mathbb{N}} \Leftarrow \mathbb{V}\, 0 \rangle\, \mathtt{nil} & \text{(Meet)} \\
 & \rightsquigarrow & \langle \mathbb{V}\, 0 \Leftarrow \mathbb{V}\, 0 \rangle\, \mathtt{nil} & \text{(Forgetful)}
\end{array}
$$

How can we organize this multiplicity of behaviors ?

Organizing admissible behaviors of casts

($\mathbb{V} := \mathtt{vect}\ A$)

$$
\begin{array}{lcll}
 & \not\rightsquigarrow & \text{(normal form)} & \text{(Free)} \\
\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \rightsquigarrow & \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\ \mathtt{err}_{\mathbb{N}} \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \text{(Meet)} \\
 & \rightsquigarrow & \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \text{(Forgetful)}
\end{array}
$$

Characterize the interactions of eliminator with casts !

$$
\mathtt{ind}_{\mathbb{V}}\ P\ h_{\mathtt{nil}}\ h_{\mathtt{cons}}\ (\langle \mathbb{V}\,m \Leftarrow \mathbb{V}\,n \rangle\,v) \qquad \text{vs} \qquad \langle P\,m \Leftarrow P\,n \rangle\,(\mathtt{ind}_{\mathbb{V}}\ P\ h_{\mathtt{nil}}\ h_{\mathtt{cons}}\ v)
$$

Organizing admissible behaviors of casts

($\mathbb{V} := \mathtt{vect}\ A$)

$$
\begin{array}{lcll}
 & \not\rightsquigarrow & \text{(normal form)} & \text{(Free)} \\
\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \rightsquigarrow & \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\ \mathtt{err}_{\mathbb{N}} \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \text{(Meet)} \\
 & \rightsquigarrow & \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \text{(Forgetful)}
\end{array}
$$

Characterize the interactions of eliminator with casts !

$$
\mathtt{ind}_{\mathbb{V}}\ P\ h_{\mathtt{nil}}\ h_{\mathtt{cons}}\ (\langle \mathbb{V}\,m \Leftarrow \mathbb{V}\,n \rangle\,v) \quad \text{vs} \quad \langle P\,m \Leftarrow P\,n \rangle\,(\mathtt{ind}_{\mathbb{V}}\ P\ h_{\mathtt{nil}}\ h_{\mathtt{cons}}\ v)
$$

$\sqsubseteq$

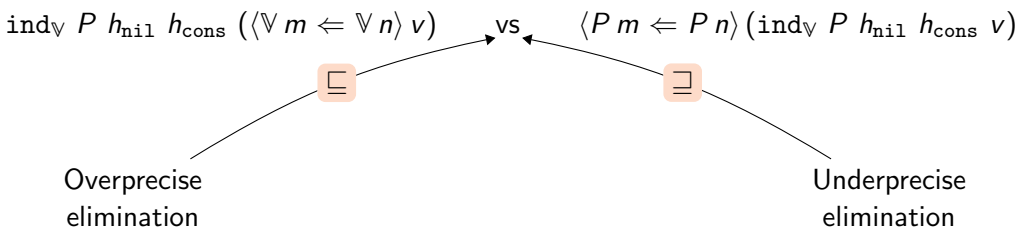
Overprecise elimination

Organizing admissible behaviors of casts

($\mathbb{V} := \mathtt{vect}\ A$)

$$
\begin{array}{lcll}
 & \not\rightsquigarrow & \text{(normal form)} & \text{(Free)} \\
\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \rightsquigarrow & \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\ \mathtt{err}_{\mathbb{N}} \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \text{(Meet)} \\
 & \rightsquigarrow & \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \text{(Forgetful)}
\end{array}
$$

Characterize the interactions of eliminator with casts !

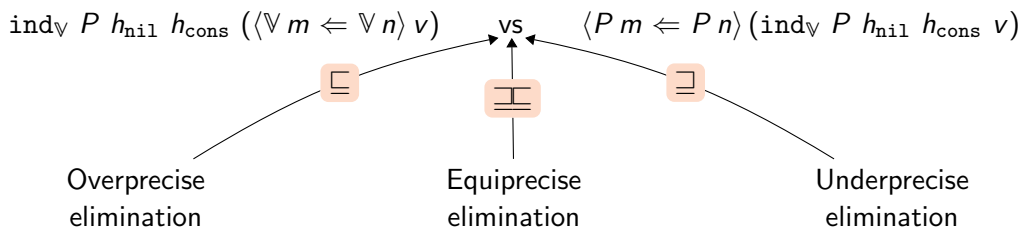


Organizing admissible behaviors of casts

($\mathbb{V} := \mathtt{vect}\ A$)

$$
\begin{array}{lcll}
 & \not\rightsquigarrow & \text{(normal form)} & \text{(Free)} \\
\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \rightsquigarrow & \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\ \mathtt{err}_{\mathbb{N}} \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \text{(Meet)} \\
 & \rightsquigarrow & \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,0 \rangle\,\mathtt{nil} & \text{(Forgetful)}
\end{array}
$$

Characterize the interactions of eliminator with casts !



Criteria for Gradual Inductive Families

Shape Relevance: No confusion of inductive type constructors

$\mathbb{V}\,0 \sqsubseteq \mathbb{V}\,?_{\mathbb{N}}$ ✓ $\mathbb{V}\,0 \sqsubseteq ?_{\mathtt{Type}}$ ✓ $\mathbb{B} \sqsubseteq \mathbb{V}\,?_{\mathbb{N}}$ ✗

Criteria for Gradual Inductive Families

Shape Relevance: No confusion of inductive type constructors

$$\mathbb{V}\,0 \sqsubseteq \mathbb{V}\,?_{\mathbb{N}} \quad \checkmark \qquad \mathbb{V}\,0 \sqsubseteq ?_{\mathtt{Type}} \quad \checkmark \qquad \mathbb{B} \sqsubseteq \mathbb{V}\,?_{\mathbb{N}} \quad \times$$

Index Relevance: Inductive families reflect precision of indices

$$\mathbb{V}\,p \sqsubseteq \mathbb{V}\,q \qquad \Longrightarrow \qquad p \;{}_{\mathbb{N}}\!\sqsubseteq_{\mathbb{N}}\; q$$

Criteria for Gradual Inductive Families

Shape Relevance: No confusion of inductive type constructors

$$\mathbb{V}\,0 \sqsubseteq \mathbb{V}\,?_{\mathbb{N}} \quad \checkmark \qquad \mathbb{V}\,0 \sqsubseteq ?_{\mathtt{Type}} \quad \checkmark \qquad \mathbb{B} \sqsubseteq \mathbb{V}\,?_{\mathbb{N}} \quad \times$$

Index Relevance: Inductive families reflect precision of indices

$$\mathbb{V}\,p \sqsubseteq \mathbb{V}\,q \qquad \Longrightarrow \qquad p \;{}_{\mathbb{N}}\!\sqsubseteq_{\mathbb{N}}\; q$$

Graduality for inductive families: Casts induce embedding-projection pairs

$$\mathbb{V}\,p \sqsubseteq \mathbb{V}\,q \quad \Longrightarrow \quad \mathbb{V}\,p \;\rightleftarrows\; \mathbb{V}\,q$$

PUNK - A framework for Gradual Indexed Inductive Types

Parametrize **Idx**-indexed inductive families with a Listable Index Accumulator **Acc**

$\mathbb{1} : \mathbf{Acc}$ $\quad \otimes : \mathbf{Idx} \times \mathbf{Acc} \to \mathbf{Acc}$ $\quad \mathtt{s} : \mathbf{Acc} \to \mathtt{list}\,\mathbf{Idx}$

PUNK - A framework for Gradual Indexed Inductive Types

Parametrize **Idx**-indexed inductive families with a Listable Index Accumulator **Acc**

$$\mathbb{1} : \mathbf{Acc} \qquad \otimes : \mathbf{Idx} \times \mathbf{Acc} \to \mathbf{Acc} \qquad \mathbb{s} : \mathbf{Acc} \to \mathtt{list}\,\mathbf{Idx}$$

Instrument constructors with accumulators ($\mathbf{Idx} := \mathbb{N}$ for $\mathbb{V}$)

$$\begin{aligned} \mathtt{nil} &: \{A : \mathtt{Type}\}(acc : \mathbf{Acc}) \to \mathbb{V}\,(0 \rtimes acc) \\ \mathtt{cons} &: \{A : \mathtt{Type}\}(acc : \mathbf{Acc})\{n : \mathbb{N}\}(a : A)(v : \mathbb{V}\,n) \to \mathbb{V}\,(\mathtt{S}\,n \rtimes acc) \end{aligned}$$

$$\text{where } n \rtimes acc := \begin{cases} k & \text{if } \mathbb{s}\,acc = [k; \ldots] \\ n & \text{otherwise} \end{cases}$$

PUNK - A framework for Gradual Indexed Inductive Types

Parametrize **Idx**-indexed inductive families with a Listable Index Accumulator **Acc**

$$\mathbb{1} : \mathbf{Acc} \qquad \otimes : \mathbf{Idx} \times \mathbf{Acc} \to \mathbf{Acc} \qquad \mathtt{s} : \mathbf{Acc} \to \mathtt{list}\,\mathbf{Idx}$$

Instrument constructors with accumulators ($\mathbf{Idx} := \mathbb{N}$ for $\mathbb{V}$)

$$\begin{aligned} \mathtt{nil} &: \{A : \mathtt{Type}\}(\mathit{acc} : \mathbf{Acc}) \to \mathbb{V}\,(0 \rtimes \mathit{acc}) \\ \mathtt{cons} &: \{A : \mathtt{Type}\}(\mathit{acc} : \mathbf{Acc})\{n : \mathbb{N}\}(a : A)(v : \mathbb{V}\,n) \to \mathbb{V}\,(\mathtt{S}\,n \rtimes \mathit{acc}) \end{aligned}$$

Accumulate casts in constructors

$$\langle \mathbb{V}\,q \Leftarrow \mathbb{V}\,p \rangle\,(\mathtt{nil}\ \mathit{acc}) \qquad \rightsquigarrow \qquad \mathtt{nil}\ (q \otimes \mathit{acc})$$

PUNK - A framework for Gradual Indexed Inductive Types

Parametrize **Idx**-indexed inductive families with a Listable Index Accumulator **Acc**

$$\mathbb{1} : \mathbf{Acc} \qquad \otimes : \mathbf{Idx} \times \mathbf{Acc} \to \mathbf{Acc} \qquad \mathbb{s} : \mathbf{Acc} \to \mathtt{list}\ \mathbf{Idx}$$

Instrument constructors with accumulators ($\mathbf{Idx} := \mathbb{N}$ for $\mathbb{V}$)

$$\begin{aligned} \mathtt{nil} &: \{A : \mathtt{Type}\}(acc : \mathbf{Acc}) \to \mathbb{V}\,(0 \rtimes acc) \\ \mathtt{cons} &: \{A : \mathtt{Type}\}(acc : \mathbf{Acc})\{n : \mathbb{N}\}(a : A)(v : \mathbb{V}\,n) \to \mathbb{V}\,(\mathtt{S}\,n \rtimes acc) \end{aligned}$$

Replay casts upon elimination

$$\mathtt{ind}_{\mathtt{vect}}\ P\ h_{\mathtt{nil}}\ h_{\mathtt{cons}}\ (\mathtt{nil}\ acc) \rightsquigarrow \langle P\, i_p \Leftarrow \ldots \Leftarrow P\, i_1 \Leftarrow P\, 0 \rangle h_{\mathtt{nil}}$$

where $\mathbb{s}\, acc = [i_p; \ldots; i_1]$

Examples of Index Accumulators

Free

$$\mathbf{Acc}_F := \texttt{list}\,\mathbf{Idx}$$
$$\mathbb{1}_F := []$$
$$i \otimes_F acc := i :: acc$$

Meet

$$\mathbf{Acc}_M := \{\top\} + \mathbf{Idx} \times \mathbf{Idx}$$
$$\mathbb{1}_M := \top$$
$$i \otimes_M \top := (i, i)$$
$$i \otimes_M (j, m) := (i, i \sqcap m)$$

Forgetful

$$\mathbf{Acc}_U := \texttt{option}\,\mathbf{Idx}$$
$$\mathbb{1}_U := \texttt{none}$$
$$i \otimes_U acc := \texttt{some}\,i$$

Examples of Index Accumulators

Free	Meet	Forgetful
$\mathbf{Acc}_F := \texttt{list}\,\mathbf{Idx}$	$\mathbf{Acc}_M := \{\top\} + \mathbf{Idx} \times \mathbf{Idx}$	$\mathbf{Acc}_U := \texttt{option}\,\mathbf{Idx}$
$\mathbb{1}_F := []$	$\mathbb{1}_M := \top$	$\mathbb{1}_U := \texttt{none}$
$i \otimes_F acc := i :: acc$	$i \otimes_M \top := (i, i)$	$i \otimes_U acc := \texttt{some}\,i$
	$i \otimes_M (j, m) := (i, i \sqcap m)$	

$$\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\,\texttt{nil}\,[] \quad \rightsquigarrow \quad \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \rangle\,\texttt{nil}\,[1] \quad \rightsquigarrow \quad \texttt{nil}\,[0;1] \qquad \text{(Free)}$$

Examples of Index Accumulators

Free	Meet	Forgetful
$\mathbf{Acc}_F := \texttt{list}\,\mathbf{Idx}$	$\mathbf{Acc}_M := \{\top\} + \mathbf{Idx} \times \mathbf{Idx}$	$\mathbf{Acc}_U := \texttt{option}\,\mathbf{Idx}$
$\mathbb{1}_F := []$	$\mathbb{1}_M := \top$	$\mathbb{1}_U := \texttt{none}$
$i \otimes_F acc := i :: acc$	$i \otimes_M \top := (i, i)$	$i \otimes_U acc := \texttt{some}\,i$
	$i \otimes_M (j, m) := (i, i \sqcap m)$	

$$\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\,\texttt{nil}\,[] \rightsquigarrow \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \rangle\,\texttt{nil}\,[1] \rightsquigarrow \texttt{nil}\,[0;1] \quad \text{(Free)}$$

$$\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\,\texttt{nil}\,\top \rightsquigarrow \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \rangle\,\texttt{nil}\,(1, 1 \sqcap 0) \rightsquigarrow \texttt{nil}\,(0, \mathrm{err}_{\mathbb{N}}) \quad \text{(Meet)}$$

Examples of Index Accumulators

Free	Meet	Forgetful
$\mathbf{Acc}_F := \texttt{list}\,\mathbf{Idx}$	$\mathbf{Acc}_M := \{\top\} + \mathbf{Idx} \times \mathbf{Idx}$	$\mathbf{Acc}_U := \texttt{option}\,\mathbf{Idx}$
$\mathbb{1}_F := []$	$\mathbb{1}_M := \top$	$\mathbb{1}_U := \texttt{none}$
$i \otimes_F acc := i :: acc$	$i \otimes_M \top := (i, i)$	$i \otimes_U acc := \texttt{some}\,i$
	$i \otimes_M (j, m) := (i, i \sqcap m)$	

$$\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\ \texttt{nil}\ [] \rightsquigarrow \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \rangle\ \texttt{nil}\ [1] \rightsquigarrow \texttt{nil}\ [0;1] \quad \text{(Free)}$$

$$\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\ \texttt{nil}\ \top \rightsquigarrow \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \rangle\ \texttt{nil}\ (1, 1 \sqcap 0) \rightsquigarrow \texttt{nil}\ (0, \mathrm{err}_{\mathbb{N}}) \quad \text{(Meet)}$$

$$\langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \Leftarrow \mathbb{V}\,0 \rangle\ \texttt{nil}\ \texttt{none} \rightsquigarrow \langle \mathbb{V}\,0 \Leftarrow \mathbb{V}\,1 \rangle\ \texttt{nil}\ (\texttt{some}\,1) \rightsquigarrow \texttt{nil}\ (\texttt{some}\,0) \quad \text{(Forgetful)}$$

Metatheory of PUNK

In the paper: General scheme for inductive families over an index accumulator

Metatheory of PUNK

In the paper: General scheme for inductive families over an index accumulator

Accumulators fit Eliminations

Forgetful accumulator	Free accumulator	Meet accumulator
⇓	⇓	⇓
Overprecise elimination	Equiprecise elimination	Underprecise elimination

Metatheory of PUNK

In the paper: General scheme for inductive families over an index accumulator

Accumulators fit Eliminations

Forgetful accumulator	Free accumulator	Meet accumulator
⇓	⇓	⇓
Overprecise elimination	Equiprecise elimination	Underprecise elimination

- ▷ Shape and Index Relevance
- ▷ Graduality for inductive families
- ▷ Subject reduction and Conservativity over CIC

Summary & Future work

Contributions

- ▷ Criteria assessing gradual inductive families: shape and index relevance, graduality
- ▷ Elim/Cast interactions organize reduction space into 3 behaviors
- ▷ PUNK & Index accumulators as a uniform approach to gradual inductive families
- ▷ Each behavior can be realized by an Index accumulator and cohabit together

Summary & Future work

Contributions

- ▷ Criteria assessing gradual inductive families: shape and index relevance, graduality
- ▷ Elim/Cast interactions organize reduction space into 3 behaviors
- ▷ PUNK & Index accumulators as a uniform approach to gradual inductive families
- ▷ Each behavior can be realized by an Index accumulator and cohabit together

PUNK is not yet a proof assistant:

- ▷ Elaboration from a gradual source
- ▷ WIP in Coq

Thank You!