# On the Semantic Expressiveness of Recursive Types – Recap


Marco Patrignani[1,2] Eric M. Martin[1] Dominique Devriese[3]


17$^{th}$ March 2021


1 Stanford University 2 CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY 3 VUB VRIJE UNIVERSITEIT BRUSSEL

# What and Why?

What is the relative semantic expressiveness of iso- and $equi$-recursive types?

# What and Why?

What is the relative semantic expressiveness of **iso**- and *equi*-recursive types?

- open question

## What and Why?

What is the relative semantic expressiveness of iso- and *equi*-recursive types?

- open question
- clarifies the design of emerging languages

# What and Why?

What is the relative semantic expressiveness of **iso-** and *equi*-recursive types?

- open question
- clarifies the design of emerging languages
- better understanding of how to answer language expressiveness questions

# How?

- Rely on Fully-Abstract Compilation to
  - phrase relative semantic expressiveness
  - compare language expressiveness

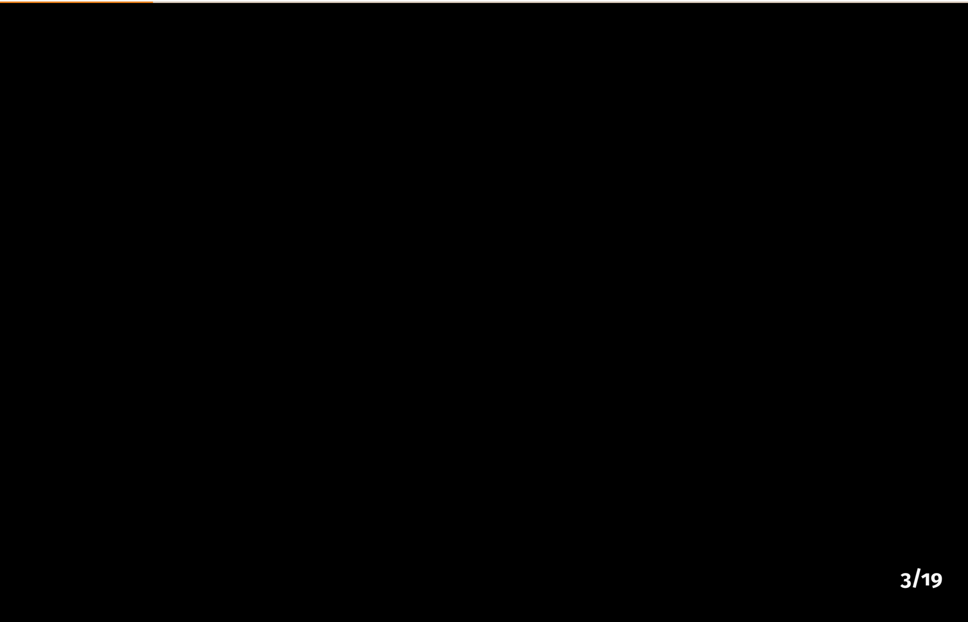
# How?

- Rely on Fully-Abstract Compilation to
  - phrase relative semantic expressiveness
  - compare language expressiveness
- Prove FAC between $\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}$ and $\lambda^{\mu}_{E}$ (and between them and $\lambda^{\mathsf{fx}}$)

# How?

- Rely on Fully-Abstract Compilation to
  - phrase relative semantic expressiveness
  - compare language expressiveness
- Prove FAC between $\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}$ and $\lambda^{\mu}_{E}$ (and between them and $\lambda^{\mathsf{fx}}$) using approximate Backtranslations and step-indexed logical approximations (or, "directional" step-indexed logical relations)

# Our Contributions, Visually



# Our Contributions, Visually

$$\boldsymbol{\lambda}^{\mu}_{\mathbf{I}} \underset{\langle\!\langle \cdot \rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}\, n}}{\overset{[\![ \cdot ]\!]^{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}}{\rightleftarrows}} \lambda^{\mu}_{E}$$

$[\![ \cdot ]\!]^{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}$ erases $\mathbf{fold}$ / $\mathbf{unfold}$, $\langle\!\langle \cdot \rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}\, n}$ is approximate

# Our Contributions, Visually

$$
\begin{tikzcd}
 & \lambda^{\mathsf{fx}} \arrow[dl, "{[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}}"'] \\
\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}} \arrow[ur, dotted, bend right, "{\langle\!\langle\cdot\rangle\!\rangle^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}}\, n}"'] \arrow[rr, "{[\![\cdot]\!]^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}}"] & & \lambda^{\mu}_{E} \arrow[ll, dotted, bend left, "{\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}\, n}"]
\end{tikzcd}
$$

$[\![\cdot]\!]^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}$ erases **fold**/**unfold**, $\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}\, n$ is approximate

$[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}$ compiles fix into Z-comb, $\langle\!\langle\cdot\rangle\!\rangle^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}}\, n$ is approximate

# Our Contributions, Visually

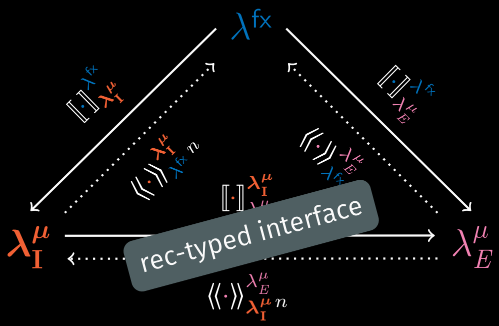
$$\begin{tikzcd}
 & \lambda^{\mathsf{fx}} \arrow[dl, "{[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}}"'] \arrow[dr, "{[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\lambda^{\mu}_{E}}}"] & \\
\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}} \arrow[ur, dotted, bend right, "{\langle\!\langle\cdot\rangle\!\rangle^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}}\, n}"'] \arrow[rr, "{[\![\cdot]\!]^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}}"] & & \lambda^{\mu}_{E} \arrow[ul, dotted, bend left, "{\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\lambda^{\mathsf{fx}}}\, n}"] \arrow[ll, dotted, bend left, "{\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}\, n}"]
\end{tikzcd}$$

$[\![\cdot]\!]^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}$ erases **fold** / **unfold**, $\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}\, n$ is approximate

$[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}$ compiles fix into Z-comb, $\langle\!\langle\cdot\rangle\!\rangle^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}}\, n$ is approximate

$[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\lambda^{\mu}_{E}} = \left[\!\!\left[ [\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}} \right]\!\!\right]^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}$, $\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\lambda^{\mathsf{fx}}}\, n = \left\langle\!\!\left\langle \langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}} \right\rangle\!\!\right\rangle^{\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}}\, n$

# Our Contributions, Visually

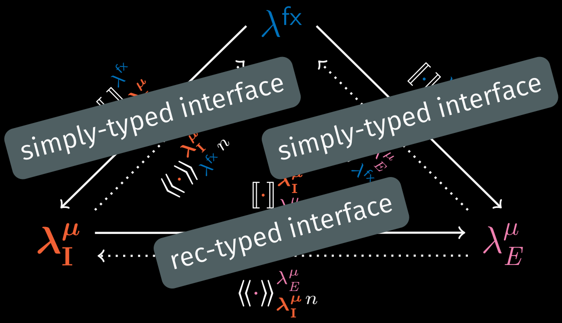



$[\![\cdot]\!]^{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}$ erases **fold**/**unfold**, $\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}\,n}$ is approximate

$[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}}$ compiles fix into Z-comb, $\langle\!\langle\cdot\rangle\!\rangle^{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}\,n}$ is approximate

$[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\lambda^{\mu}_{E}} = \left[\!\!\left[ [\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}} \right]\!\!\right]^{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}$, $\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\lambda^{\mathsf{fx}}\,n} = \left\langle\!\!\left\langle \langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}} \right\rangle\!\!\right\rangle^{\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}\,n}$

# Our Contributions, Visually




$[\![\cdot]\!]^{\lambda^{\mu}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}$ erases **fold**/**unfold**, $\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\lambda^{\mu}_{\mathbf{I}}\,n}$ is approximate

$[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\lambda^{\mu}_{\mathbf{I}}}$ compiles fix into Z-comb, $\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}\,n}$ is approximate

$[\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\lambda^{\mu}_{E}} = \left[\!\!\left[ [\![\cdot]\!]^{\lambda^{\mathsf{fx}}}_{\lambda^{\mu}_{\mathbf{I}}} \right]\!\!\right]^{\lambda^{\mu}_{\mathbf{I}}}_{\lambda^{\mu}_{E}}$, $\langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\lambda^{\mathsf{fx}}\,n} = \left\langle\!\!\left\langle \langle\!\langle\cdot\rangle\!\rangle^{\lambda^{\mu}_{E}}_{\lambda^{\mu}_{\mathbf{I}}} \right\rangle\!\!\right\rangle^{\lambda^{\mu}_{\mathbf{I}}}_{\lambda^{\mathsf{fx}}\,n}$

# Comparing Language Expressiveness

# Comparing Language Expressiveness

- Turing-expressiveness is not ok: not higher-order (see Mitchell'93)

# Comparing Language Expressiveness

- Turing-expressiveness is not ok: not higher-order (see Mitchell'93)
- Instead: reason with elements of the language, bound to its semantics

# Comparing Language Expressiveness

- Turing-expressiveness is not ok: not higher-order (see Mitchell'93)

---

- Instead: reason with elements of the language, bound to its semantics
- Observe the interaction between terms ($t$) and contexts ($C$) over an interface ($C[t]$)

# Comparing Language Expressiveness

- Turing-expressiveness is not ok: not higher-order (see Mitchell'93)

---

- Instead: reason with elements of the language, bound to its semantics
- Observe the interaction between terms ($t$) and contexts ($C$) over an interface ($C[t]$)
- Hp: take the same $t$ in $\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}$ and $\lambda^{\mu}_{E}$

# Comparing Language Expressiveness

- Turing-expressiveness is not ok: not higher-order (see Mitchell'93)

---

- Instead: reason with elements of the language, bound to its semantics
- Observe the interaction between terms ($t$) and contexts ($C$) over an interface ($C[t]$)
- Hp: take the same $t$ in $\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}$ and $\lambda^{\mu}_{E}$
- Q: does $\mathbf{C}[t]$ behave differently from $C[t]$?

# Comparing Language Expressiveness

- Turing-expressiveness is not ok: not higher-order (see Mitchell'93)
- Instead: reason with elements of the language, bound to its semantics
- Observe the interaction between terms ($t$) and contexts ($C$) over an interface ($C[t]$)
- Hp: take the same $t$ in $\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}$ and $\lambda^{\mu}_{E}$
- Q: does $\mathbf{C}[t]$ behave differently from $C[t]$?

# Comparing Language Expressiveness

- Turing-expressiveness is not ok: not higher-order (see Mitchell'93)

---

- Instead: reason with elements of the language, bound to its semantics
- Observe the interaction between terms ($t$) and contexts ($C$) over an interface ($C[t]$)
- Hp: take the same $t$ in $\boldsymbol{\lambda}^{\mu}_{\mathbf{I}}$ and $\lambda^{\mu}_{E}$
- Q: does $\mathbf{C}[t]$ behave differently from $C[t]$?

# Observing Behaviour through Contexts

- use prog. eq: lets us compare language abstractions (i.e., hiding)

  e.g., $\mathbf{pack}\,\langle \mathrm{N}, \mathbf{0}\rangle\,\mathbf{as}\ \tau$ and $\mathbf{pack}\,\langle \mathrm{N}, \mathbf{1}\rangle\,\mathbf{as}\ \tau$

  compiled into $\langle \mathit{0}, \mathit{0}\rangle$ and $\langle \mathit{0}, \mathit{1}\rangle$

# Observing Behaviour through Contexts

- use prog. eq: lets us compare language abstractions (i.e., hiding)

  e.g., $\mathbf{pack}\,\langle \mathrm{N}, \mathbf{0}\rangle\,\mathbf{as}\,\tau$ and $\mathbf{pack}\,\langle \mathrm{N}, \mathbf{1}\rangle\,\mathbf{as}\,\tau$

  compiled into $\langle \mathit{0}, \mathit{0}\rangle$ and $\langle \mathit{0}, \mathit{1}\rangle$
- take two programs $\mathbf{t_1}$ and $\mathbf{t_2}$

# Observing Behaviour through Contexts

- use prog. eq: lets us compare language abstractions (i.e., hiding)
  e.g., $\mathbf{pack}\,\langle \mathrm{N}, \mathbf{0}\rangle\,\mathbf{as}\ \tau$ and $\mathbf{pack}\,\langle \mathrm{N}, \mathbf{1}\rangle\,\mathbf{as}\ \tau$
  compiled into $\langle \mathit{0}, \mathit{0}\rangle$ and $\langle \mathit{0}, \mathit{1}\rangle$
- take two programs $\mathbf{t_1}$ and $\mathbf{t_2}$
- Q: do $\mathbf{C}\,[\mathbf{t_1}]$ and $\mathbf{C}\,[\mathbf{t_2}]$ behave the same <u>if and only if</u> $C\,[\mathbf{t_1}]$ and $C\,[\mathbf{t_2}]$ also behave the same?

# Observing Behaviour through Contexts

- use prog. eq: lets us compare language abstractions (i.e., hiding)
  e.g., $\mathbf{pack}\,\langle \mathrm{N}, \mathbf{0}\rangle\,\mathbf{as}\,\tau$ and $\mathbf{pack}\,\langle \mathrm{N}, \mathbf{1}\rangle\,\mathbf{as}\,\tau$
  compiled into $\langle \mathit{0}, \mathit{0}\rangle$ and $\langle \mathit{0}, \mathit{1}\rangle$
- take two programs $\mathbf{t_1}$ and $\mathbf{t_2}$
- Q: do $\mathbf{C}\,[\mathbf{t_1}]$ and $\mathbf{C}\,[\mathbf{t_2}]$ behave the same if and only if $C\,[\mathbf{t_1}]$ and $C\,[\mathbf{t_2}]$ also behave the same?
- behaviour: $\Uparrow$ vs $\Downarrow$
  alternatives exist (e.g., traces) but this is operational-semantics-based

# The same $t$ in $\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}$ & $\lambda^{\mu}_{E}$

- Compiler must be “canonical”

$$[\![\cdot]\!] : \mathbf{t} \to t$$

# The same $t$ in $\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}$ & $\lambda^{\mu}_{E}$

- Compiler must be “canonical”

$$[\![\cdot]\!] : \mathbf{t} \to t$$

- $[\![\cdot]\!]$: identity and erase $\mathbf{fold}$ / $\mathbf{unfold}$

# The same $t$ in $\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}$ & $\lambda^{\mu}_{E}$

- Compiler must be "canonical"

$$[\![\cdot]\!] : \mathbf{t} \rightarrow t$$

- $[\![\cdot]\!]$: identity and erase $\mathbf{fold}$ / $\mathbf{unfold}$
  $\boldsymbol{\lambda}^{\boldsymbol{\mu}}_{\mathbf{I}}$ & $\lambda^{\mu}_{E}$ semantics are identical (almost)

# Fully Abstract Compilation (FAC)

- does $[\![\cdot]\!]$ attain FAC?

# Fully Abstract Compilation (FAC)

- does $[\![\cdot]\!]$ attain FAC?
- $\vdash [\![\cdot]\!] : \mathsf{FAC} \overset{\mathsf{def}}{=} \forall \mathbf{t_1}, \mathbf{t_2}$

$$\mathbf{t_1} \simeq_{\mathbf{ctx}} \mathbf{t_2} \iff [\![\mathbf{t_1}]\!] \simeq_{\mathrm{ctx}} [\![\mathbf{t_2}]\!]$$

# Fully Abstract Compilation (FAC)

- does $[\![\cdot]\!]$ attain FAC?
- $\vdash [\![\cdot]\!] : \text{FAC} \stackrel{\text{def}}{=} \forall \mathbf{t_1}, \mathbf{t_2}$

$$\mathbf{t_1} \simeq_{\mathbf{ctx}} \mathbf{t_2} \iff [\![\mathbf{t_1}]\!] \simeq_{ctx} [\![\mathbf{t_2}]\!] \qquad \text{or:}$$

$$\begin{gathered}(\forall \mathbf{C}.\mathbf{C}[\mathbf{t_1}]\Downarrow \iff \mathbf{C}[\mathbf{t_2}]\Downarrow) \\ \Updownarrow \\ (\forall C.C[[\![\mathbf{t_1}]\!]]\Downarrow \iff C[[\![\mathbf{t_2}]\!]]\Downarrow)\end{gathered}$$

# Preservation via Step-Idx Log. Approx.

$$\mathbf{t}_1 \simeq_{\mathrm{ctx}} \mathbf{t}_2$$

ctx. eq. preservation

$$[\![\mathbf{t}_1]\!] \overset{?}{\simeq}_{\mathrm{ctx}} [\![\mathbf{t}_2]\!]$$

# Preservation via Step-Idx Log. Approx.

$$\mathrm{t}_1 \simeq_{\mathrm{ctx}} \mathrm{t}_2$$

ctx. eq. preservation

$$C\left[\llbracket \mathrm{t}_1 \rrbracket\right] \Downarrow_n \quad \overset{?}{\Rightarrow} \quad C\left[\llbracket \mathrm{t}_2 \rrbracket\right] \Downarrow_\_$$

$$\llbracket \mathrm{t}_1 \rrbracket \overset{?}{\simeq}_{\mathrm{ctx}} \llbracket \mathrm{t}_2 \rrbracket$$

# Preservation via Step-Idx Log. Approx.

$$\mathbf{t}_1 \simeq_{\mathrm{ctx}} \mathbf{t}_2$$

$\mathbf{t}_1 \gtrsim_{\_} [\![\mathbf{t}_1]\!]$

$? \gtrsim_n C$

(1)

$$C\left[[\![\mathbf{t}_1]\!]\right] \Downarrow_n \quad \overset{?}{\Rightarrow} \quad C\left[[\![\mathbf{t}_2]\!]\right] \Downarrow_{\_}$$

$$[\![\mathbf{t}_1]\!] \overset{?}{\simeq}_{\mathrm{ctx}} [\![\mathbf{t}_2]\!]$$

ctx. eq. preservation

# Preservation via Step-Idx Log. Approx.

$$\mathbf{t_1} \simeq_{\mathbf{ctx}} \mathbf{t_2}$$

$$\mathbf{t_1} \gtrsim_{\_} [\![\mathbf{t_1}]\!]$$

$$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_n C$$

(1)

$$C\Big[[\![\mathbf{t_1}]\!]\Big]\Downarrow_n \quad \overset{?}{\Rightarrow} \quad C\Big[[\![\mathbf{t_2}]\!]\Big]\Downarrow_{\_}$$

$$[\![\mathbf{t_1}]\!] \overset{?}{\simeq}_{\mathrm{ctx}} [\![\mathbf{t_2}]\!]$$

ctx. eq. preservation

# Preservation via Step-Idx Log. Approx.

$\mathbf{t} \gtrsim_{\_}$ and $\lesssim_{\_} [\![\mathbf{t}]\!]$
$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_{\_}$ and $\lesssim_{\_} C$

$$\mathbf{t_1} \simeq_{\mathbf{ctx}} \mathbf{t_2}$$

$\mathbf{t_1} \gtrsim_{\_} [\![\mathbf{t_1}]\!]$
$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_n C$

(1)

ctx. eq. preservation

$$C\left[[\![\mathbf{t_1}]\!]\right] \Downarrow_n \quad \overset{?}{\Rightarrow} \quad C\left[[\![\mathbf{t_2}]\!]\right] \Downarrow_{\_}$$

$$[\![\mathbf{t_1}]\!] \overset{?}{\simeq}_{\mathrm{ctx}} [\![\mathbf{t_2}]\!]$$

# Preservation via Step-Idx Log. Approx.

$\mathbf{t} \gtrsim_{\_}$ and $\lesssim_{\_} [\![\mathbf{t}]\!]$
$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_{\_}$ and $\lesssim_{\_} C$

$$\mathbf{t_1} \simeq_{\mathbf{ctx}} \mathbf{t_2}$$

$$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}}[\mathbf{t_1}] \Downarrow_{\_}$$

$$\mathbf{t_1} \gtrsim_{\_} [\![\mathbf{t_1}]\!]$$
$$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_n C$$

(1)

$$C\left[[\![\mathbf{t_1}]\!]\right] \Downarrow_n \quad \overset{?}{\Rightarrow} \quad C\left[[\![\mathbf{t_2}]\!]\right] \Downarrow_{\_}$$

$$[\![\mathbf{t_1}]\!] \overset{?}{\simeq}_{\mathrm{ctx}} [\![\mathbf{t_2}]\!]$$

ctx. eq. preservation

# Preservation via Step-Idx Log. Approx.

$\mathbf{t} \gtrsim_{\_}$ and $\lesssim_{\_} [\![\mathbf{t}]\!]$
$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_{\_}$ and $\lesssim_{\_} C$

$$\mathbf{t_1} \simeq_{\mathbf{ctx}} \mathbf{t_2}$$

$$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}}[\mathbf{t_1}] \Downarrow_{\_} \underset{(2)}{\Rightarrow} \langle\!\langle C \rangle\!\rangle_{\mathbf{n}}[\mathbf{t_2}] \Downarrow_{\_}$$

$\mathbf{t_1} \gtrsim_{\_} [\![\mathbf{t_1}]\!]$
$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_{n} C$

(1)

$$C\left[[\![\mathbf{t_1}]\!]\right] \Downarrow_{n} \overset{?}{\Rightarrow} C\left[[\![\mathbf{t_2}]\!]\right] \Downarrow_{\_}$$

$$[\![\mathbf{t_1}]\!] \overset{?}{\simeq}_{\mathrm{ctx}} [\![\mathbf{t_2}]\!]$$

ctx. eq. preservation

# Preservation via Step-Idx Log. Approx.

$\mathbf{t} \gtrsim_{\_}$ and $\lesssim_{\_} [\![\mathbf{t}]\!]$

$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_{\_}$ and $\lesssim_{\_} C$

$$\mathbf{t_1} \simeq_{\mathbf{ctx}} \mathbf{t_2}$$

$$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}}[\mathbf{t_1}] \Downarrow_{\_} \quad \underset{(2)}{\Rightarrow} \quad \langle\!\langle C \rangle\!\rangle_{\mathbf{n}}[\mathbf{t_2}] \Downarrow_{\_}$$

$\mathbf{t_1} \gtrsim_{\_} [\![\mathbf{t_1}]\!]$

$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \gtrsim_{n} C$

(1)

(3)

$\mathbf{t_2} \lesssim_{\_} [\![\mathbf{t_2}]\!]$

$\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \lesssim_{\_} C$

$$C\left[[\![\mathbf{t_1}]\!]\right] \Downarrow_{n} \quad \overset{?}{\Rightarrow} \quad C\left[[\![\mathbf{t_2}]\!]\right] \Downarrow_{\_}$$

$$[\![\mathbf{t_1}]\!] \overset{?}{\simeq}_{\mathrm{ctx}} [\![\mathbf{t_2}]\!]$$

ctx. eq. preservation

# Overapproximation

Define a cross-language logical approximation $(\mathcal{V}\llbracket\cdot\rrbracket_{\triangledown}, \mathcal{E}\llbracket\cdot\rrbracket_{\triangledown}, \cdots)$ $\triangledown$ is the direction

- $\mathbf{t} \lesssim_n \llbracket\mathbf{t}\rrbracket$ : by def. $\cdots$ $\mathbf{t}$ and $\llbracket\mathbf{t}\rrbracket$ are in the obs.:

$$O(W)_{\lesssim} \stackrel{\text{def}}{=} \left\{ (\mathbf{t}, \llbracket\mathbf{t}\rrbracket) \;\middle|\; \begin{array}{l} \text{if } lev(W) > n \text{ and } \mathbf{t} \hookrightarrow^{\mathbf{n}} \mathbf{v} \\ \text{then } \exists \mathbf{k}.\ \llbracket\mathbf{t}\rrbracket \hookrightarrow^{k} v \end{array} \right\}$$

# Overapproximation

Define a cross-language logical approximation $(\mathcal{V}[\![\cdot]\!]_{\triangledown}, \mathcal{E}[\![\cdot]\!]_{\triangledown}, \cdots)$ $\triangledown$ is the direction

- $\mathbf{t} \lesssim_n [\![\mathbf{t}]\!]$ : by def. $\cdots$ $\mathbf{t}$ and $[\![\mathbf{t}]\!]$ are in the obs.:

$$O(W)_{\lesssim} \stackrel{\text{def}}{=} \left\{ (\mathbf{t}, [\![\mathbf{t}]\!]) \;\middle|\; \begin{array}{l} \text{if } lev(W) > n \text{ and } \mathbf{t} \hookrightarrow^{\mathbf{n}} \mathbf{v} \\ \text{then } \exists \mathbf{k}.\ [\![\mathbf{t}]\!] \hookrightarrow^{k} v \end{array} \right\}$$

- $\mathbf{t} \gtrsim_n [\![\mathbf{t}]\!]$ :
  same, flipped implication

# The Approximate Backtranslation

- Need to craft $\mathbf{C}$, we only have $C$

# The Approximate Backtranslation

- Need to craft $\mathbf{C}$, we only have $C$
- $[\![\cdot]\!]$: $\mathbf{t} \rightarrow t$ is defined on $\mathbf{t}$'s syntax

# The Approximate Backtranslation

- Need to craft $\mathbf{C}$, we only have $C$
- $[\![\cdot]\!]$: $\mathbf{t} \to t$ is defined on $\mathbf{t}$'s syntax
- $\langle\!\langle \cdot \rangle\!\rangle_n : C \to \mathbf{C}$
  approximate $C$'s coinductive derivation

# The Approximate Backtranslation

- Need to craft $\mathbf{C}$, we only have $C$
- $[\![\cdot]\!]$: $\mathbf{t} \to t$ is defined on $\mathbf{t}$'s syntax
- $\langle\!\langle\cdot\rangle\!\rangle_n : C \to \mathbf{C}$
  approximate $C$'s coinductive derivation
- sufficient because FAC cares about co-termination

# Approximate Backtranslation Type

- Backtranslation does not know for which $n$ it runs

# Approximate Backtranslation Type

- Backtranslation does not know for which $n$ it runs
  embed $n$ in the type of backtranslated ctx.
  NO: $\langle\!\langle\cdot\rangle\!\rangle_n : \tau \to \boldsymbol{\tau}$ YES: $\langle\!\langle\cdot\rangle\!\rangle_n : \tau \to \mathbf{BtT}_{\mathbf{n};\boldsymbol{\tau}}$

# Approximate Backtranslation Type

- Backtranslation does not know for which $n$ it runs
  embed $n$ in the type of backtranslated ctx.
  NO: $\langle\langle\cdot\rangle\rangle_n : \tau \to \boldsymbol{\tau}$ YES: $\langle\langle\cdot\rangle\rangle_n : \tau \to \mathbf{BtT}_{\mathbf{n};\boldsymbol{\tau}}$

$$\mathbf{BtT}_{\mathbf{0};\tau} \stackrel{\text{def}}{=} \mathbf{Unit}$$

$$\mathbf{BtT}_{\mathbf{n+1};\tau} \stackrel{\text{def}}{=} \begin{cases} \mathbf{Unit} \uplus \mathbf{Unit} & \text{if } \tau = \mathit{Unit} \\ (\mathbf{BtT}_{\mathbf{n};\boldsymbol{\tau}} \to \mathbf{BtT}_{\mathbf{n};\boldsymbol{\tau}'}) \uplus \mathbf{Unit} & \text{if } \tau = \tau \to \tau' \\ (\mathbf{BtT}_{\mathbf{n};\boldsymbol{\tau}} \uplus \mathbf{BtT}_{\mathbf{n};\boldsymbol{\tau}'}) \uplus \mathbf{Unit} & \text{if } \tau = \tau \uplus \tau' \\ \mathbf{BtT}_{\mathbf{n+1};\boldsymbol{\tau}'[\mu\alpha.\boldsymbol{\tau}'/\alpha]} \uplus \mathbf{Unit} & \text{if } \tau = \mu\alpha.\,\tau' \end{cases}$$

# Backtranslation Example and Relation

$\langle\!\langle\, unit \,\rangle\!\rangle_{n>0}$ = ?

$\mathbf{BtT}_{\mathbf{n+1};\,Unit} = \mathbf{Unit} \uplus \mathbf{Unit}$

# Backtranslation Example and Relation

$\langle\langle unit \rangle\rangle_{n>0}$ = $\mathbf{inl\ unit}$

$\mathbf{BtT}_{\mathbf{n+1};Unit}$ = $\mathbf{Unit} \uplus \mathbf{Unit}$

# Backtranslation Example and Relation

$\langle\langle unit \rangle\rangle_{n>0}$ = $\mathbf{inl\ unit}$ $\mathbf{BtT}_{\mathbf{n+1};Unit} = \mathbf{Unit} \uplus \mathbf{Unit}$

Cannot relate using normal LR:

$$\mathcal{V}\,[\![\mathbf{Unit}]\!]_{\triangledown} \stackrel{\text{def}}{=} \{(\mathbf{unit}, unit)\}$$

# Backtranslation Example and Relation

$\langle\langle unit \rangle\rangle_{n>0}$ = $\mathbf{inl\ unit}$ $\mathbf{BtT}_{\mathbf{n+1};Unit} = \mathbf{Unit} \uplus \mathbf{Unit}$

Cannot relate using normal LR:

$$\mathcal{V}\,[\![\mathbf{Unit}]\!]_{\triangledown} \stackrel{\text{def}}{=} \{(\mathbf{unit}, unit)\}$$

Need a special value relation:

$$\mathcal{V}\,[\![\mathbf{EmulT}_{\mathbf{n+1};\tau}]\!]_{\triangledown} \stackrel{\text{def}}{=} \Big\{(\mathbf{v}, v) \mid \text{either } \mathbf{v} = \mathbf{inr\ unit}$$
$$\text{or } \tau = Unit \text{ and } \exists \mathbf{v}'.\ \mathbf{v} = \mathbf{inl\ v}' \text{ and}$$
$$(\mathbf{v}', v) \in \mathcal{V}\,[\![\mathbf{Unit}]\!]_{\triangledown}\Big\}$$

# Backtranslation Definition

- $\langle\!\langle \cdot \rangle\!\rangle$ is made of

# Backtranslation Definition

- $\langle\!\langle \cdot \rangle\!\rangle$ is made of
- context ‘emulation’ (to generate the context)
  which needs upgrading and downgrading (for well-typedness)

# Backtranslation Definition

- $\langle\!\langle \cdot \rangle\!\rangle$ is made of
- context ‘emulation’ (to generate the context)
  which needs upgrading and downgrading (for well-typedness)
- inject/extract: to fix typing of context interface

# Technicality #1: Emulate

- translate to ‘the same’ term
  (switch to TR, p 146)

# Technicality #1: Emulate

- translate to 'the same' term
  (switch to TR, p 146)
- to typecheck constructors, we need to $\mathbf{inl}$ them

# Technicality #1: Emulate

- translate to ‘the same’ term
  (switch to TR, p 146)
- to typecheck constructors, we need to $\mathbf{inl}$ them
- typing error! need to lose a step!

# Technicality #1.1: Upgrade/Downgrade

- needed to ensure well-typedness only

# Technicality #1.1: Upgrade/Downgrade

- needed to ensure well-typedness only
- recursively traverse a term and add or lose a level (i.e., an $\mathbf{inl}$ )
  (switch to TR, only in blue though, p 96)

# Technicality #1.1: Upgrade/Downgrade

$$\text{if } (n < m \text{ and } p = \mathtt{precise}) \text{ or } (\triangledown = \lesssim \text{ and } p = \mathtt{imprecise})$$
$$\Gamma \vdash \mathbf{t} \triangledown_n t : \mathbf{EmulT}_{\mathbf{m+d;p;}\tau}$$
$$\text{then } \Gamma \vdash \mathbf{downgrade}_{\mathbf{m;d;}\tau}\ \mathbf{t} \triangledown_n t : \mathbf{EmulT}_{\mathbf{m;p;}\tau}$$

$$\text{if } (n < m \text{ and } p = \mathtt{precise}) \text{ or } (\triangledown = \lesssim \text{ and } p = \mathtt{imprecise})$$
$$\Gamma \vdash \mathbf{t} \triangledown_n t : \mathbf{EmulT}_{\mathbf{m;p;}\tau}$$
$$\text{then } \Gamma \vdash \mathbf{upgrade}_{\mathbf{m;d;}\tau}\ \mathbf{t} \triangledown_n t : \mathbf{EmulT}_{\mathbf{m+d;p;}\tau}$$

# Technicality #1: Emulate

if $(m > n$ and $p = \texttt{precise})$ or $(\triangledown = \lesssim$ and $p = \texttt{imprecise})$

$\Gamma \vdash t : \tau$

then $\texttt{toEmul}_{\mathbf{m;p}}(\Gamma) \vdash \mathbf{emulate_m}(\Gamma \vdash t : \tau) \triangledown_n t : \mathbf{EmulT}_{\mathbf{m;p};\tau}$

Key:

if $\tau \triangleq \sigma$

and $\texttt{ftv}(\tau) = \texttt{ftv}(\sigma) = \varnothing$

then $\mathbf{BtT}_{\mathbf{n};\tau} = \mathbf{BtT}_{\mathbf{n};\sigma}$ for all $n$

# Technicality #2: Inject/Extract

- Since $\mathtt{t} : \boldsymbol{\tau}$ implies $[\![\mathtt{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[: \tau]$

# Technicality #2: Inject/Extract

- Since $\mathbf{t} : \boldsymbol{\tau}$ implies $[\![\mathbf{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[: \tau]$
- $\langle\!\langle C \rangle\!\rangle_{\mathbf{n}}\,[: ?]$

# Technicality #2: Inject/Extract

- Since $\mathbf{t} : \boldsymbol{\tau}$ implies $[\![\mathbf{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[: \tau]$
- $\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \left[: \mathbf{BtT}_{\mathbf{n};\tau}\right]$
- Mismatch! $\boldsymbol{\tau} \neq \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$

# Technicality #2: Inject/Extract

- Since $\mathbf{t} : \boldsymbol{\tau}$ implies $[\![\mathbf{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[: \tau]$
- $\langle\!\langle C \rangle\!\rangle_{\mathbf{n}} \left[: \mathbf{BtT}_{\mathbf{n};\tau}\right]$
- Mismatch! $\boldsymbol{\tau} \neq \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$ e.g.,

  $\boldsymbol{\mu\alpha}.\,\mathbf{Unit} \uplus \boldsymbol{\alpha} \neq \mathbf{BtT}_{\mathbf{1};[\![\boldsymbol{\mu\alpha}.\mathbf{Unit}\uplus\boldsymbol{\alpha}]\!]} = (\mathbf{Unit} \uplus \mathbf{Unit}) \uplus \mathbf{Unit}$

# Technicality #2: Inject/Extract

- Since $\mathbf{t} : \boldsymbol{\tau}$ implies $[\![\mathbf{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[: \tau]$
- $\langle\!\langle\, C \,\rangle\!\rangle_{\mathbf{n}} \left[: \mathbf{BtT}_{\mathbf{n};\tau}\right]$
- Mismatch! $\boldsymbol{\tau} \neq \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$ e.g.,
  $\boldsymbol{\mu\alpha}.\,\mathbf{Unit} \uplus \boldsymbol{\alpha} \neq \mathbf{BtT}_{\mathbf{1};[\![\boldsymbol{\mu\alpha}.\mathbf{Unit}\uplus\boldsymbol{\alpha}]\!]} = (\mathbf{Unit} \uplus \mathbf{Unit}) \uplus \mathbf{Unit}$
- $\langle\!\langle [\cdot] \rangle\!\rangle_{\mathbf{n}} = \left[\mathbf{inject}_{\mathbf{n};\boldsymbol{\tau}}\ \cdot\right]$

# Technicality #2: Inject/Extract

- Since $\mathrm{t} : \boldsymbol{\tau}$ implies $[\![\mathrm{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[: \tau]$
- $\langle\!\langle\, C \,\rangle\!\rangle_{\mathbf{n}} \left[: \mathbf{BtT}_{\mathbf{n};\tau}\right]$
- Mismatch! $\boldsymbol{\tau} \neq \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$ e.g.,
  $\boldsymbol{\mu\alpha}.\,\mathbf{Unit} \uplus \boldsymbol{\alpha} \neq \mathbf{BtT}_{1;[\![\boldsymbol{\mu\alpha}.\mathbf{Unit}\uplus\boldsymbol{\alpha}]\!]} = (\mathbf{Unit} \uplus \mathbf{Unit}) \uplus \mathbf{Unit}$
- $\langle\!\langle [\cdot] \rangle\!\rangle_{\mathbf{n}} = \left[\mathbf{inject}_{\mathbf{n};\boldsymbol{\tau}}\ \cdot\right]$
  $\mathbf{inject}_{\mathbf{n};\boldsymbol{\tau}} : \boldsymbol{\tau} \to \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$

# Technicality #2: Inject/Extract

- Since $\mathrm{t} : \boldsymbol{\tau}$ implies $[\![\mathrm{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[: \tau]$
- $\langle\!\langle C \rangle\!\rangle_{\mathbf{n}}\,[: \mathbf{BtT}_{\mathbf{n};\tau}]$
- Mismatch! $\boldsymbol{\tau} \neq \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$ e.g.,
  $\boldsymbol{\mu\alpha}.\,\mathbf{Unit} \uplus \boldsymbol{\alpha} \neq \mathbf{BtT}_{1;[\![\boldsymbol{\mu\alpha}.\mathbf{Unit}\uplus\boldsymbol{\alpha}]\!]} = (\mathbf{Unit} \uplus \mathbf{Unit}) \uplus \mathbf{Unit}$
- $\langle\!\langle [\cdot] \rangle\!\rangle_{\mathbf{n}} = [\mathbf{inject}_{\mathbf{n};\boldsymbol{\tau}}\ \cdot]$
  $\mathbf{inject}_{\mathbf{n};\boldsymbol{\tau}} : \boldsymbol{\tau} \to \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$

# Technicality #2: Inject/Extract

- Since $\mathbf{t} : \boldsymbol{\tau}$ implies $[\![\mathbf{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[\cdot : \tau]$
- $\langle\!\langle C \rangle\!\rangle_{\mathbf{n}}\,\left[\cdot : \mathbf{BtT}_{\mathbf{n};\tau}\right]$
- Mismatch! $\boldsymbol{\tau} \neq \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$ e.g.,
  $\boldsymbol{\mu\alpha}.\,\mathbf{Unit} \uplus \boldsymbol{\alpha} \neq \mathbf{BtT}_{1;[\![\boldsymbol{\mu\alpha}.\mathbf{Unit}\uplus\boldsymbol{\alpha}]\!]} = (\mathbf{Unit} \uplus \mathbf{Unit}) \uplus \mathbf{Unit}$
- $\langle\!\langle [\cdot] \rangle\!\rangle_{\mathbf{n}} = \left[\mathbf{inject}_{\mathbf{n};\boldsymbol{\tau}}\ \cdot\right]$
  $\mathbf{inject}_{\mathbf{n};\boldsymbol{\tau}} : \boldsymbol{\tau} \rightarrow \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$

# Technicality #2: Inject/Extract

- Since $\mathrm{t} : \boldsymbol{\tau}$ implies $[\![\mathrm{t}]\!] : \overbrace{[\![\boldsymbol{\tau}]\!]}^{\tau}$
- And $C\,[: \tau]$
- $\langle\!\langle C \rangle\!\rangle_{\mathbf{n}}\left[: \mathbf{BtT}_{\mathbf{n};\tau}\right]$
- Mismatch! $\boldsymbol{\tau} \neq \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$ e.g.,
  $\boldsymbol{\mu\alpha}.\,\mathbf{Unit} \uplus \boldsymbol{\alpha} \neq \mathbf{BtT}_{\mathbf{1};[\![\boldsymbol{\mu\alpha}.\mathbf{Unit}\uplus\boldsymbol{\alpha}]\!]} = (\mathbf{Unit} \uplus \mathbf{Unit}) \uplus \mathbf{Unit}$
- $\langle\!\langle [\cdot] \rangle\!\rangle_{\mathbf{n}} = \left[\mathbf{inject}_{\mathbf{n};\tau}\ \cdot\right]$
  $\mathbf{inject}_{\mathbf{n};\boldsymbol{\tau}} : \boldsymbol{\tau} \to \mathbf{BtT}_{\mathbf{n};[\![\boldsymbol{\tau}]\!]}$

# Technicality #2: Inject/Extract

If $(m \geq n$ and $p = \texttt{precise})$ or $(\triangledown = \lesssim$ and $p = \texttt{imprecise})$

then if $\Gamma \vdash \mathbf{t} \triangledown_n t : \boldsymbol{\tau}$

then $\Gamma \vdash \mathbf{inject}_{\mathbf{m};\boldsymbol{\tau}}\ \mathbf{t} \triangledown_n t : \mathbf{EmulT}_{\mathbf{m};\mathbf{p};\texttt{isToEq}(\boldsymbol{\tau})}$

if $\Gamma \vdash \mathbf{t} \triangledown_n t : \mathbf{EmulT}_{\mathbf{m};\mathbf{p};\texttt{isToEq}(\boldsymbol{\tau})}$

then $\Gamma \vdash \mathbf{extract}_{\mathbf{m};\boldsymbol{\tau}}\ \mathbf{t} \triangledown_n t : \boldsymbol{\tau}$