Verifying Compilers using Multi-Language Semantics

Amal Ahmed (with James T. Perconti)

Northeastern University

Semantics-preserving compilation

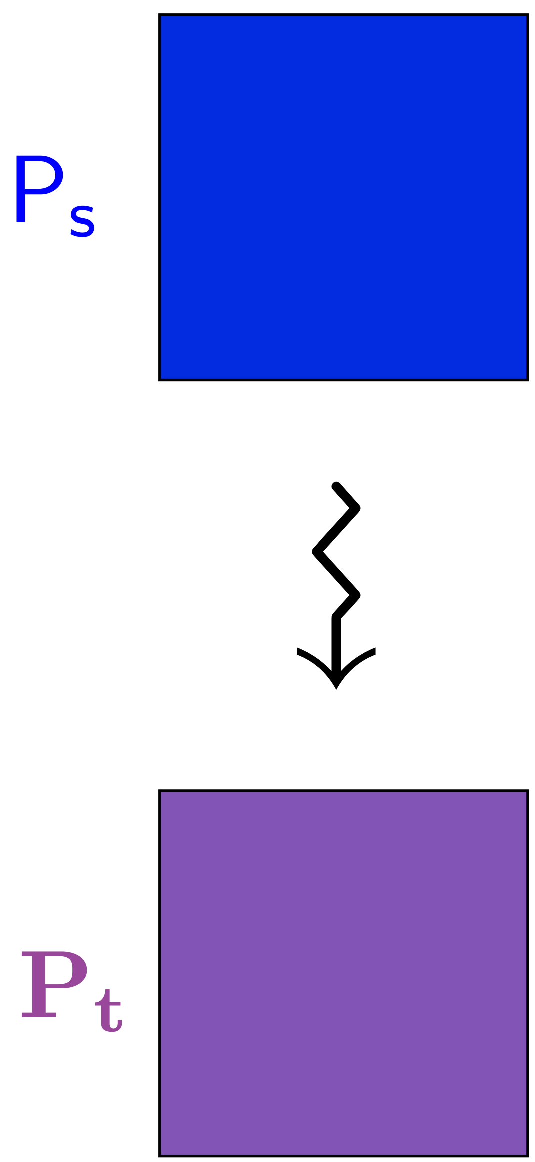
$$s \leadsto t \quad\Longrightarrow\quad s \approx t$$

compiles to ($\leadsto$)

same meaning ($\approx$)

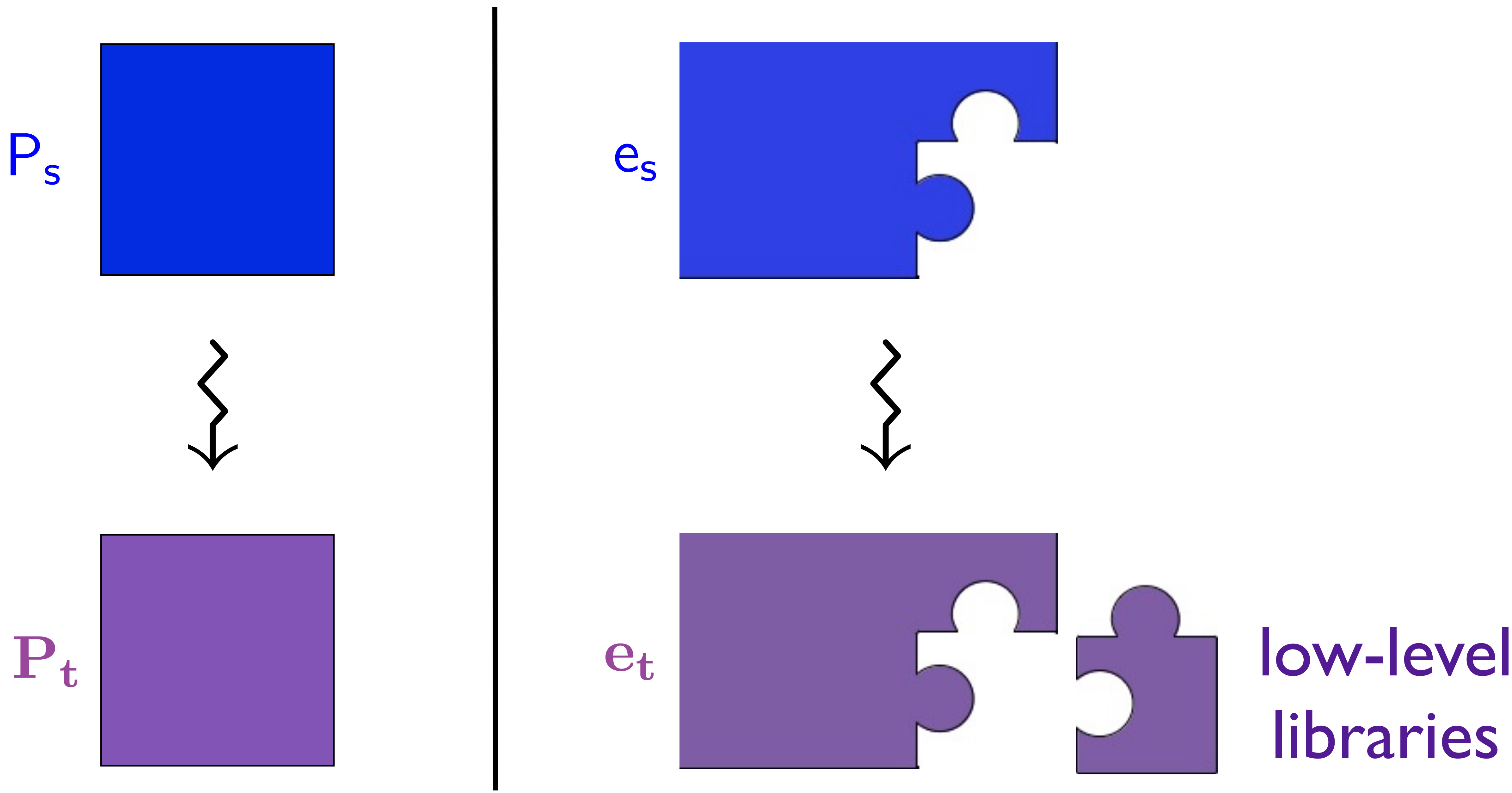
Problem: Closed-World Assumption

Correct compilation guarantee only applies to **whole** programs!



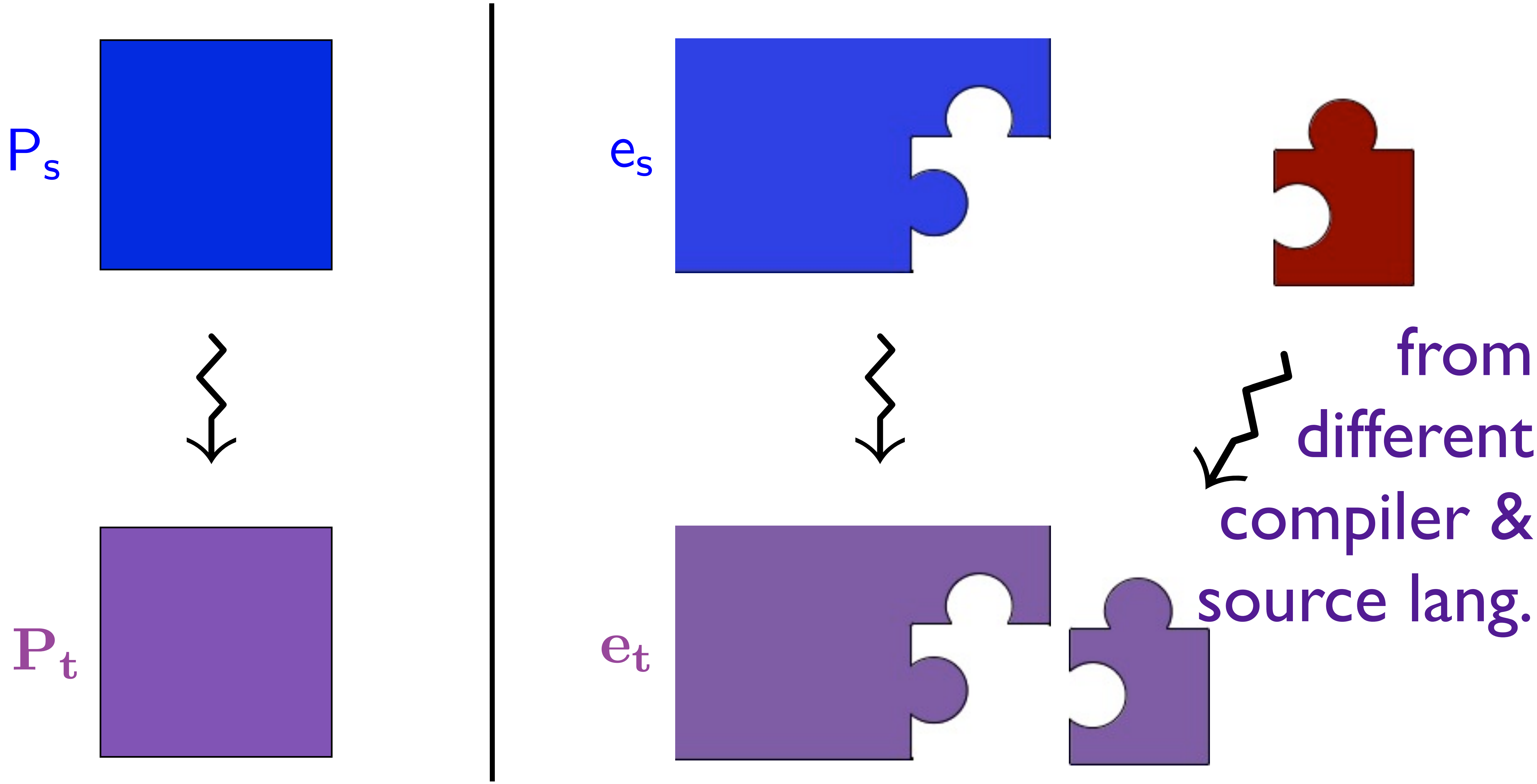
Problem: Closed-World Assumption

Correct compilation guarantee only applies to whole programs!



Problem: Closed-World Assumption

Correct compilation guarantee only applies to **whole** programs!



Why Whole Programs?

$$s \leadsto t \quad \Longrightarrow \quad s \approx t$$

expressed how?

Why Whole Programs?

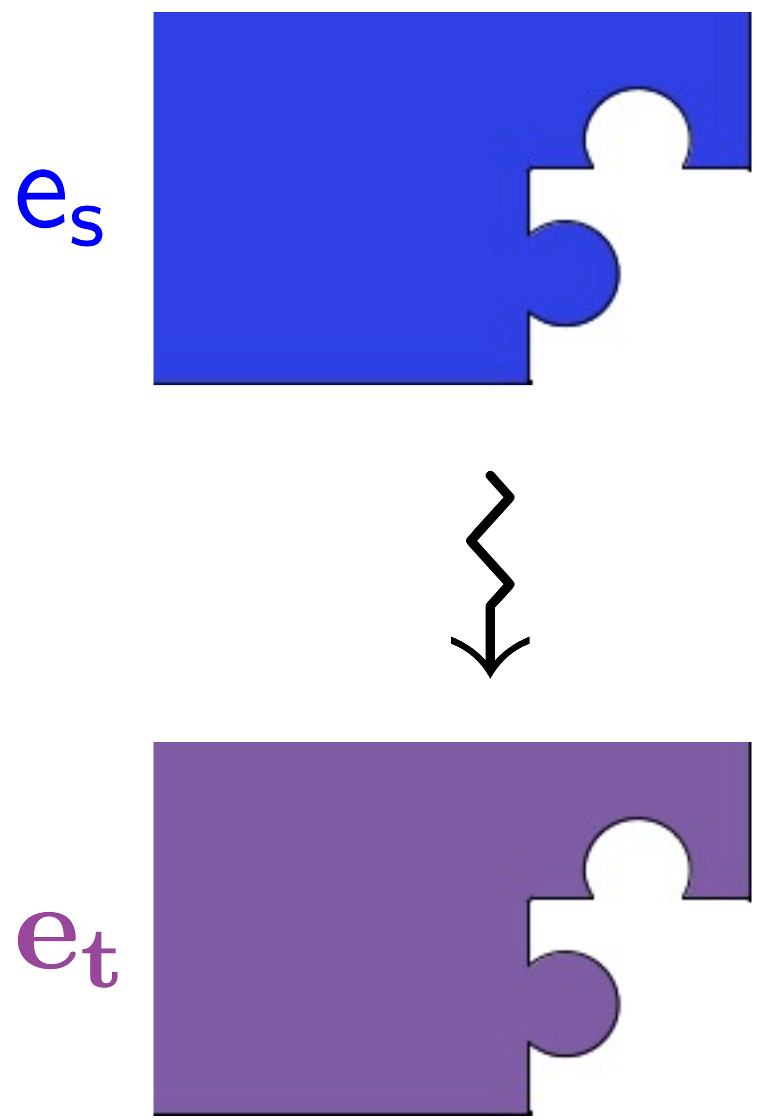
$$P_s \rightsquigarrow P_t \quad \Longrightarrow \quad P_s \approx P_t$$

expressed how?

CompCert

$$\begin{array}{ccccccccc}
P_s & \longmapsto & \ldots & \longmapsto & P_s^i & \longmapsto & P_s^{i+1} & \longmapsto & \ldots \\
| & & & & | & & \backslash & & \\
P_t & \longmapsto & \ldots & \longmapsto & P_t^j & \longmapsto^* & P_t^{j+n} & \longmapsto & \ldots
\end{array}$$

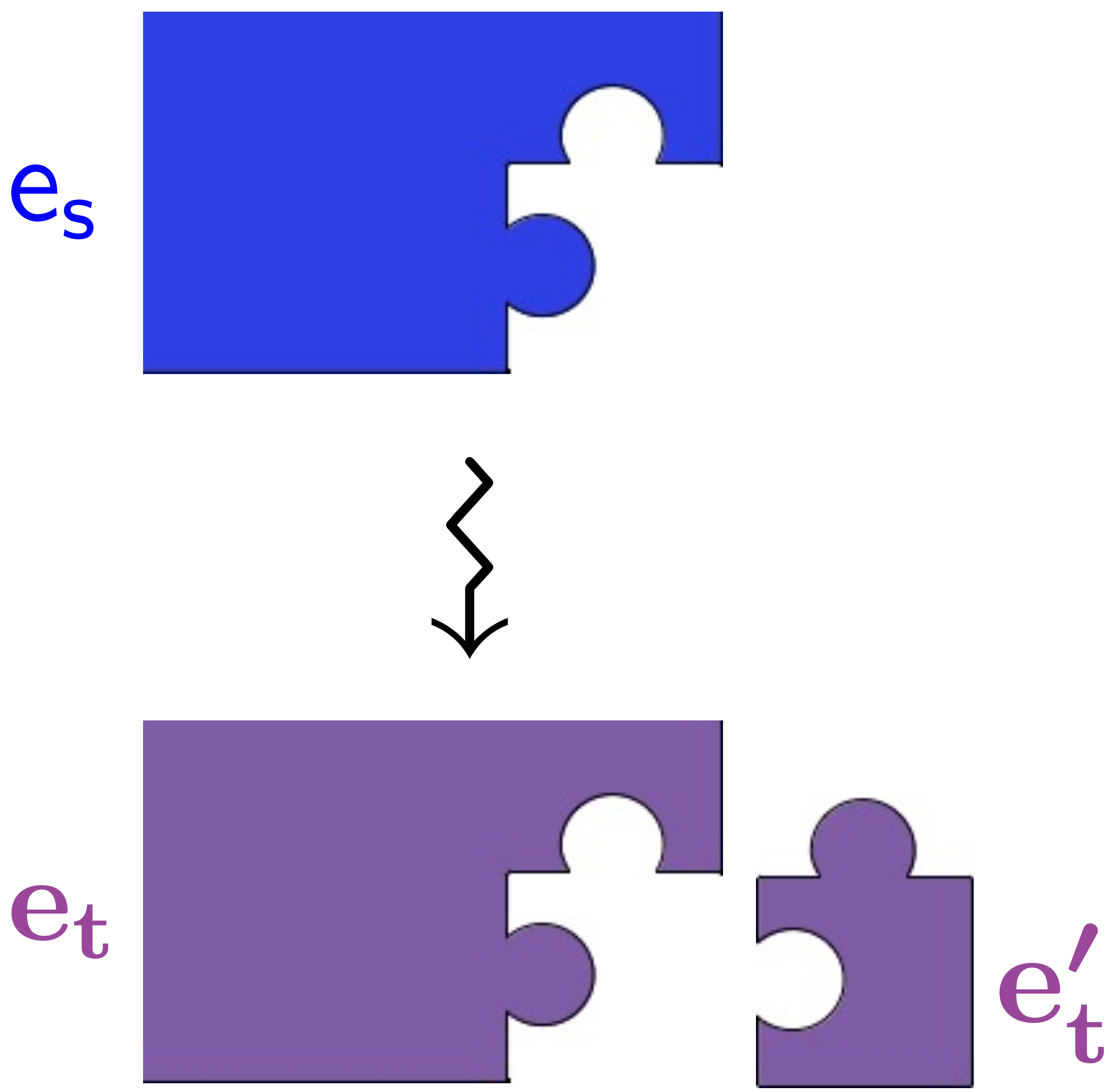
Correct Compilation of Components?



$e_S \approx e_T$

expressed how?

Correct Compilation of Components?

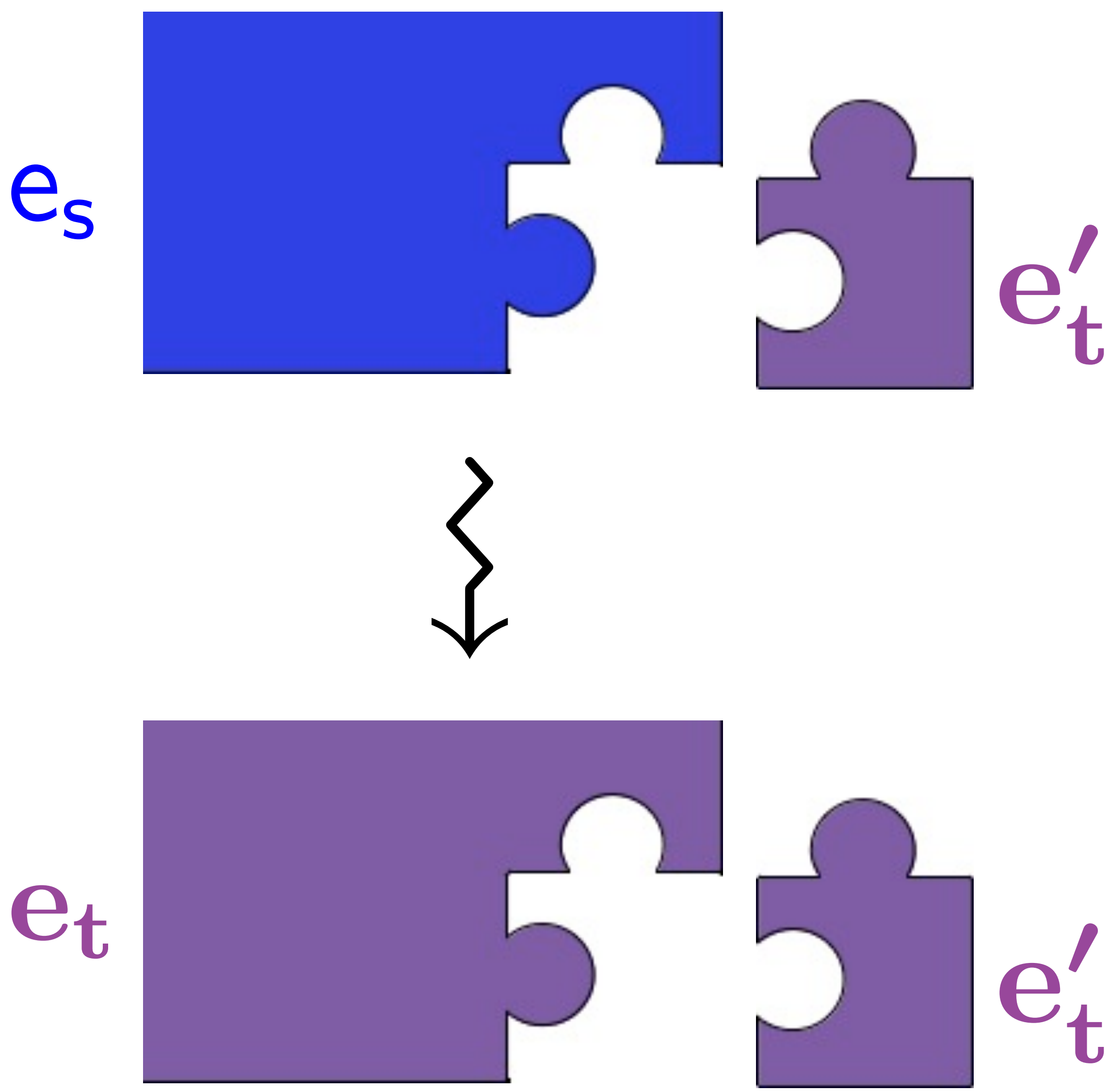


$e_S \approx e_T$

↑

expressed how?

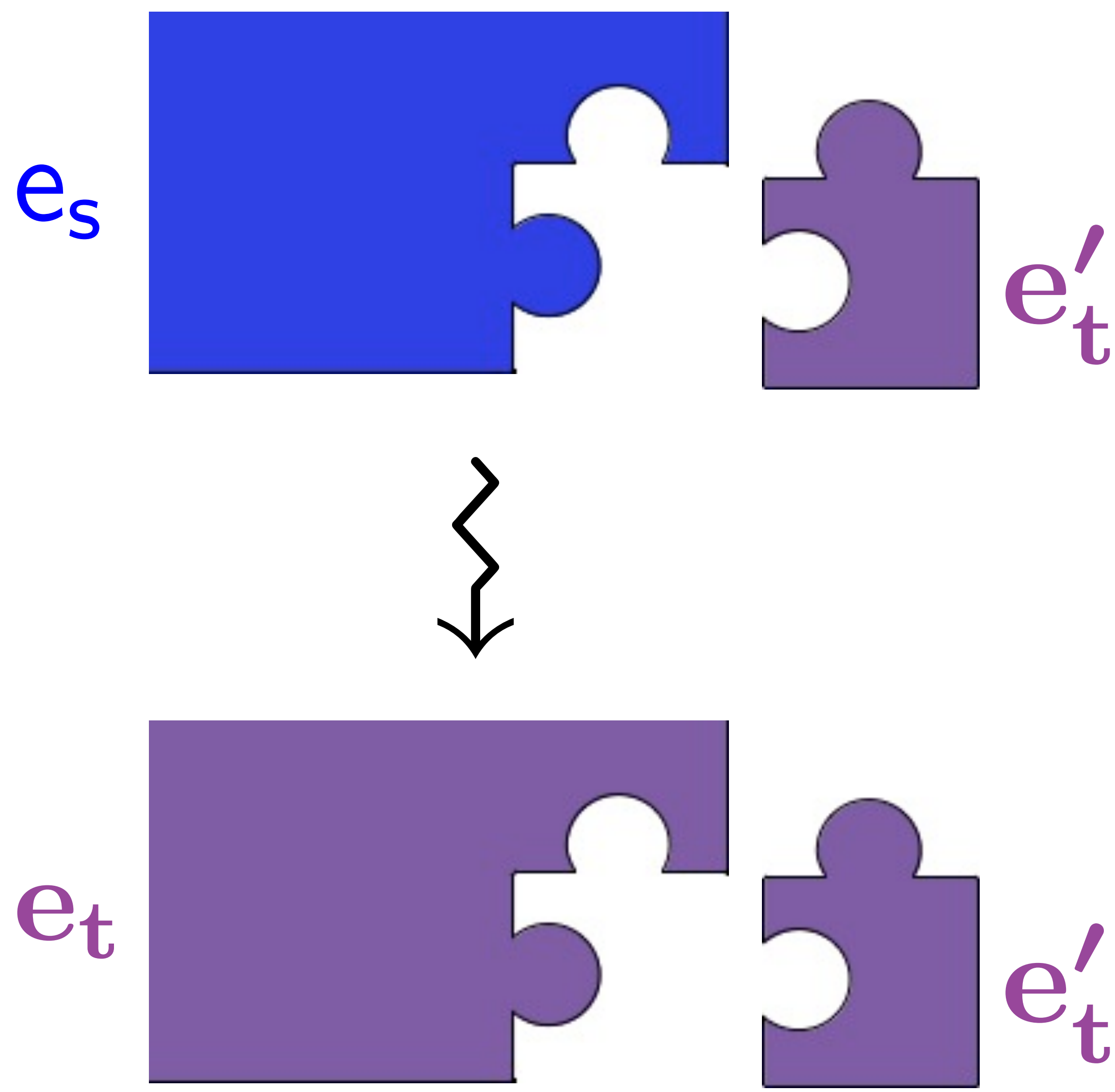
Correct Compilation of Components?



$e_S \approx e_T$

expressed how?

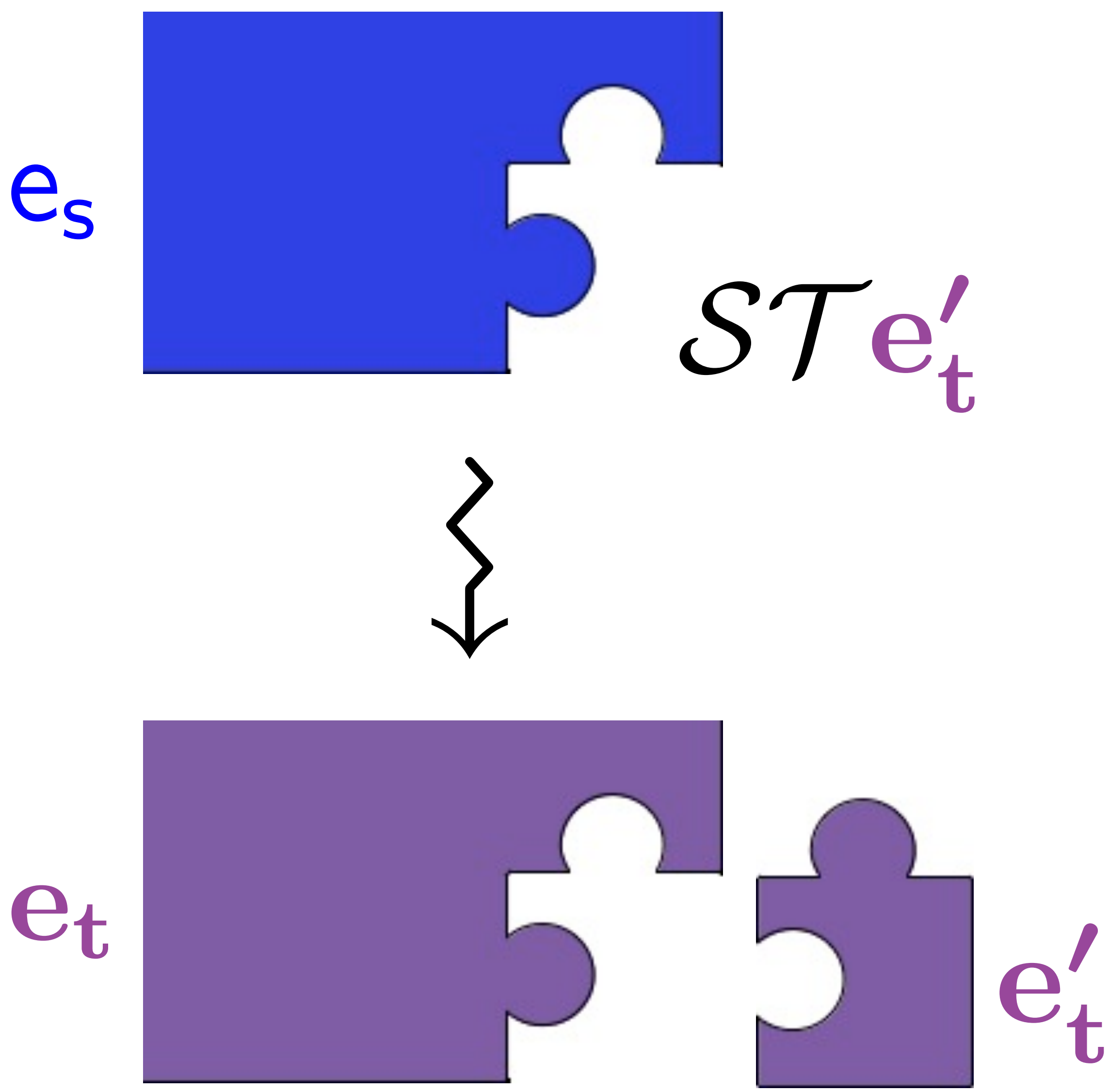
Correct Compilation of Components?



Need a semantics of source-target interoperability:

$\mathcal{ST}\, e_t \qquad \mathcal{TS}\, e_s$

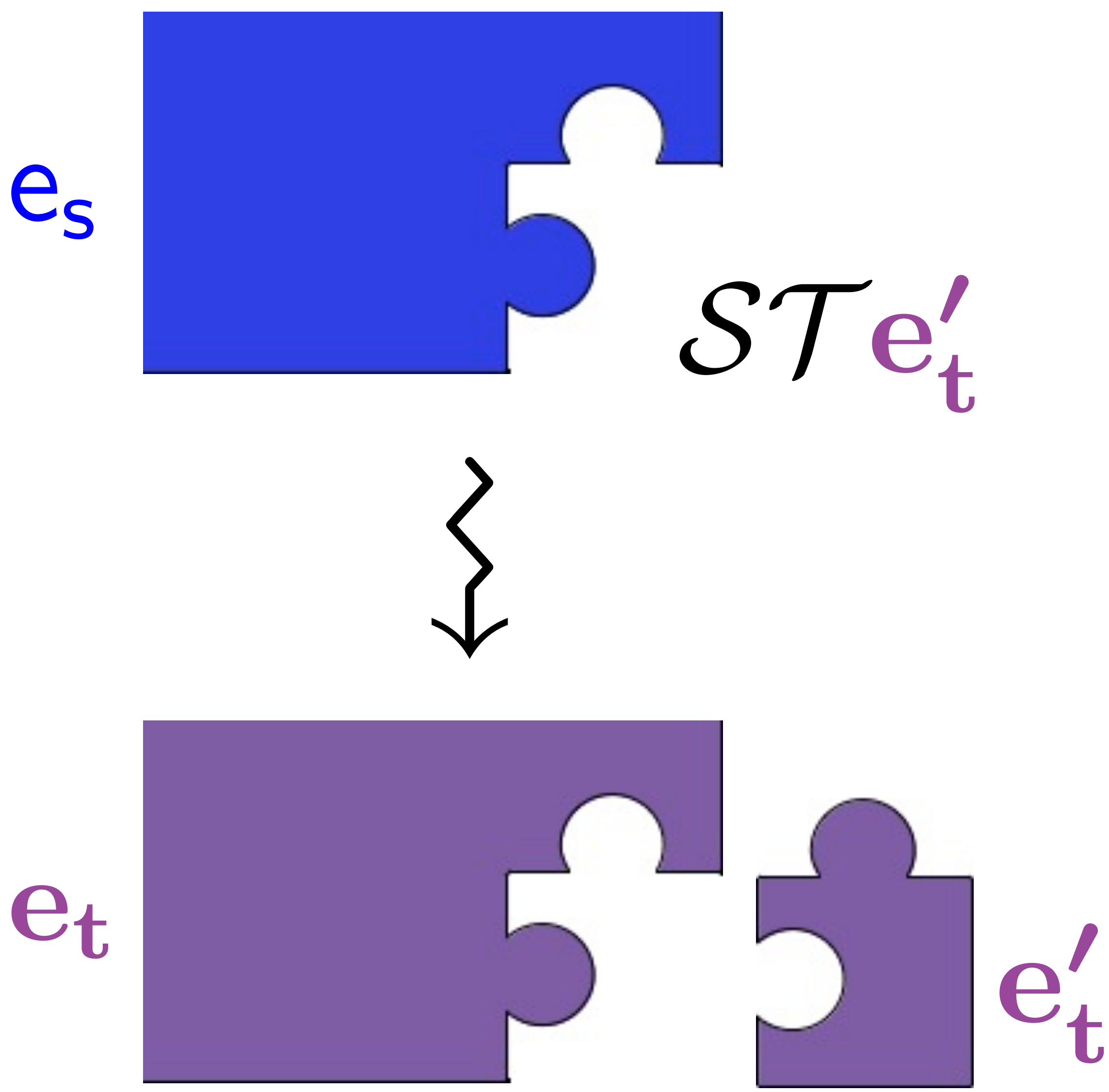
Correct Compilation of Components?



Need a semantics of source-target interoperability:

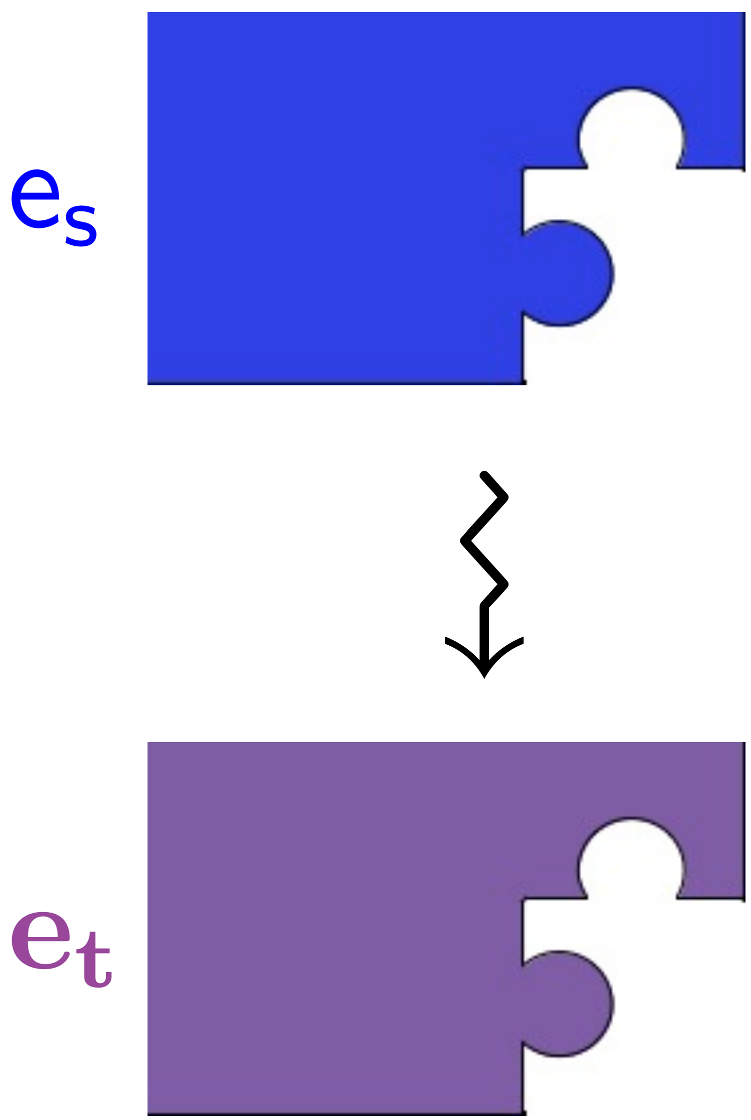
$\mathcal{ST}e_t \quad \mathcal{TS}e_s$

Correct Compilation of Components?



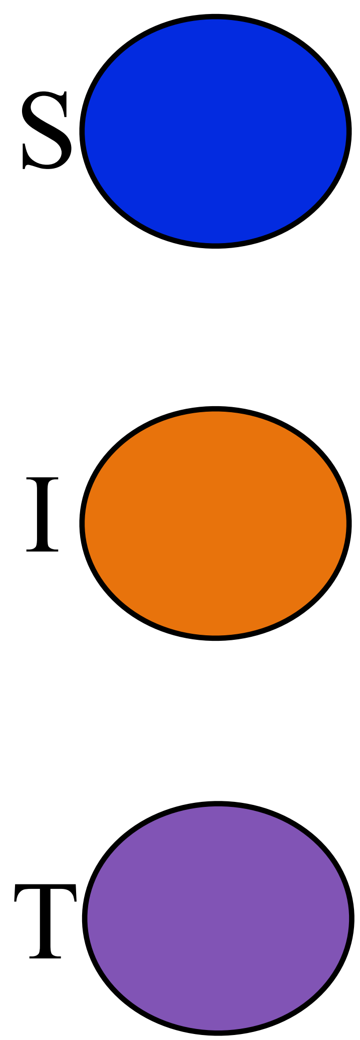
$$\mathcal{TS}(e_s\ (\mathcal{ST}\,e_t')) \approx^{ctx} e_t\ e_t'$$

Correct Compilation of Components

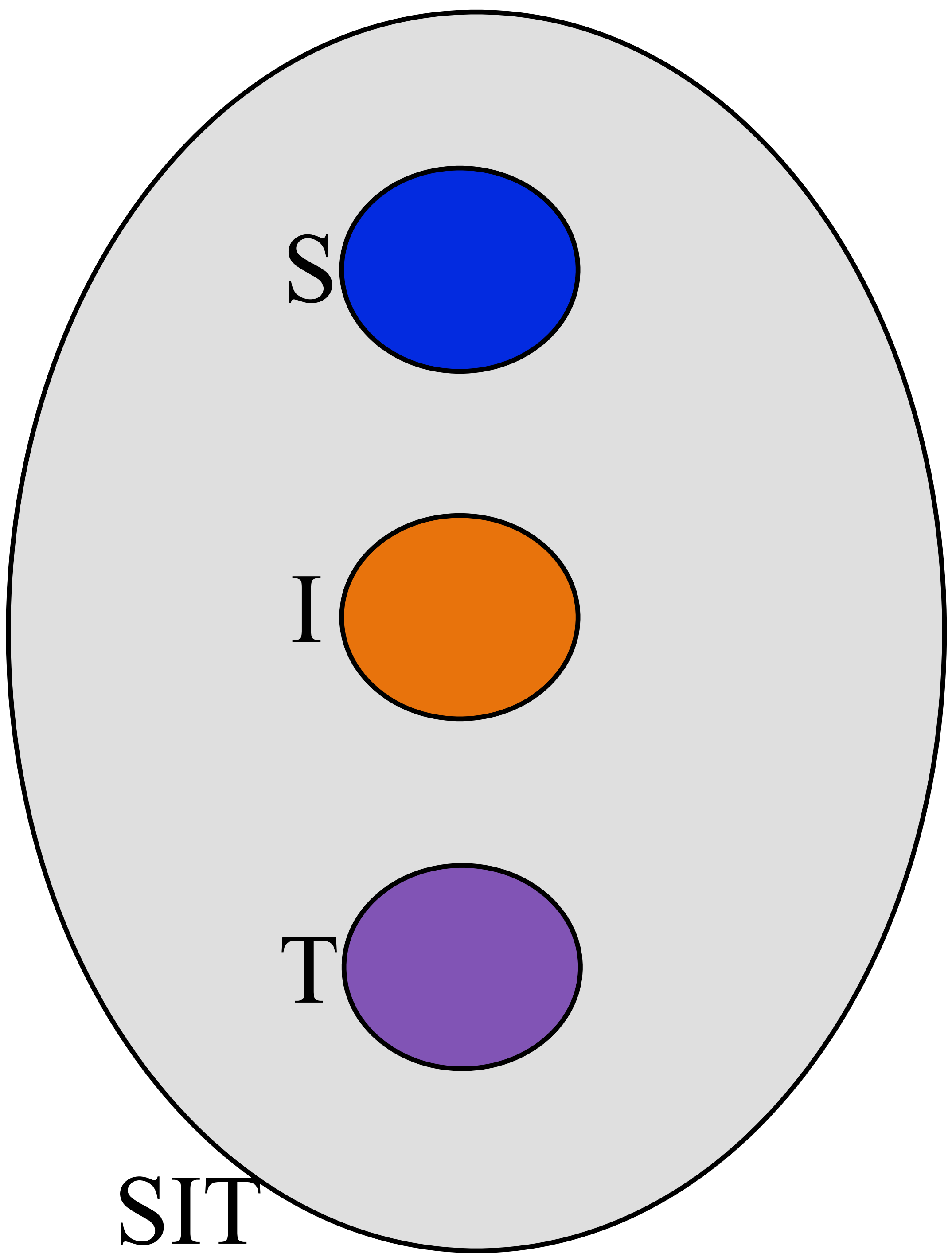


$$\mathbf{e_S} \approx \mathbf{e_T} \stackrel{\text{def}}{=} \mathbf{e_S} \approx^{ctx} \mathcal{S}\mathcal{T}\mathbf{e_T}$$

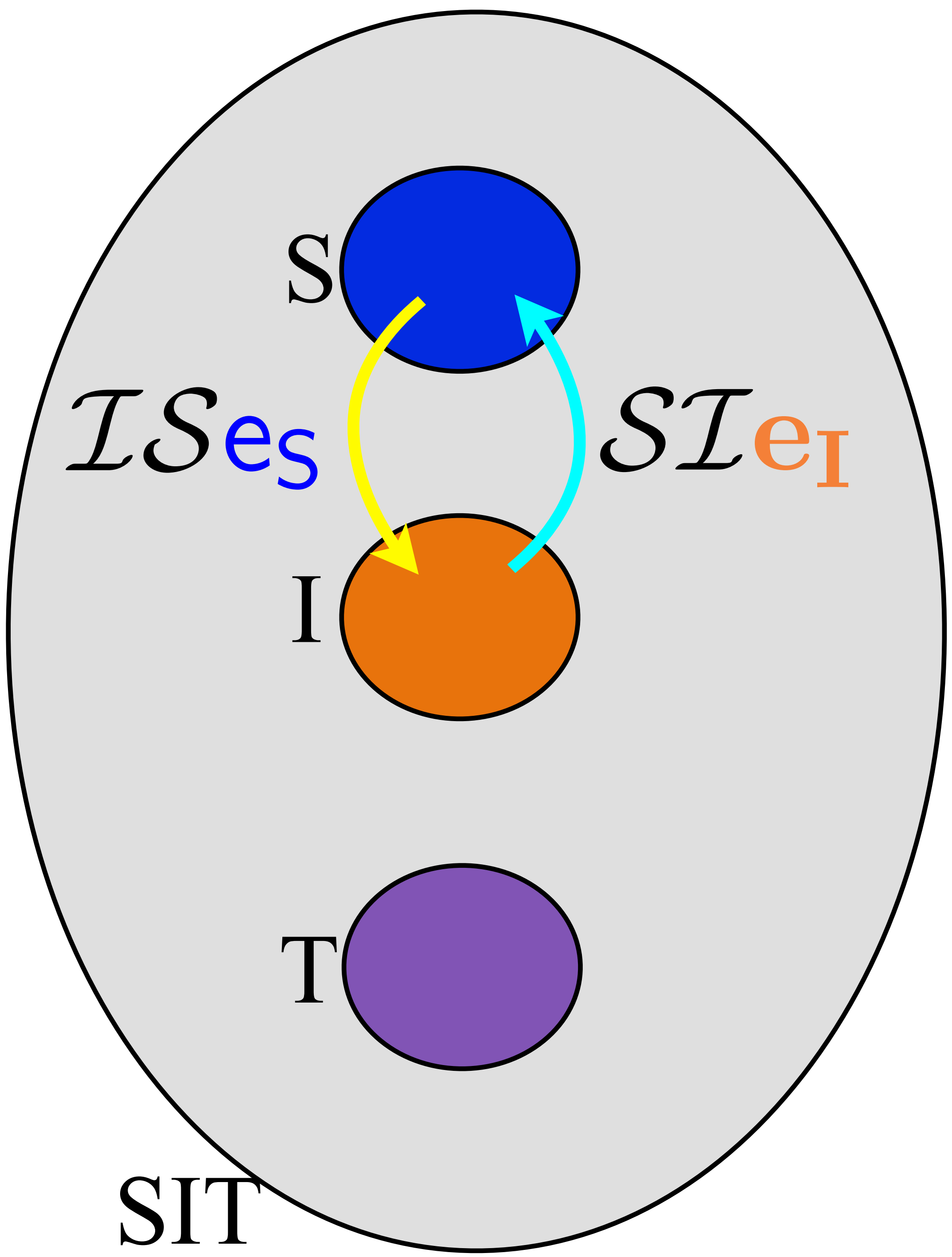
Our Approach (multi-pass compiler)



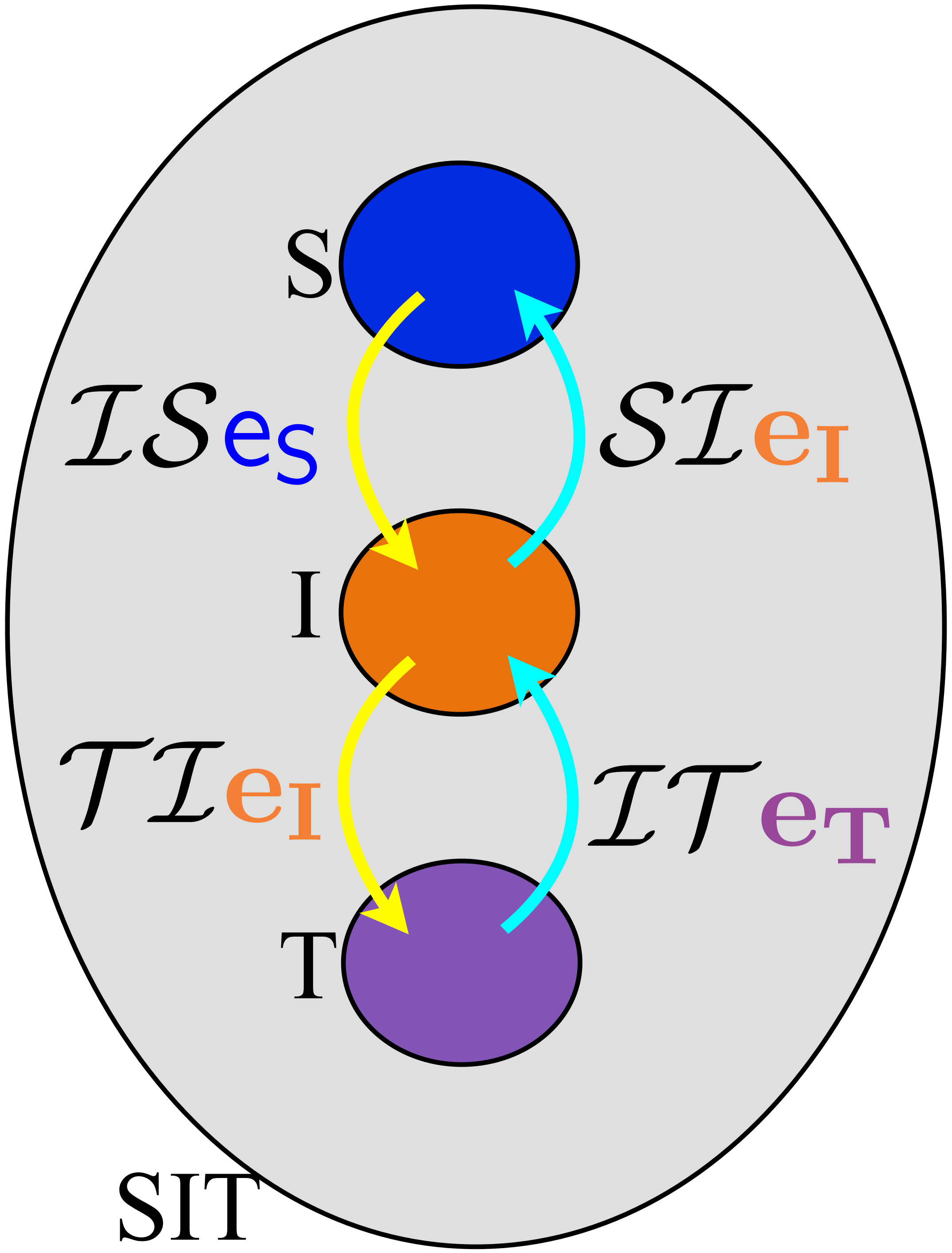
Our Approach (multi-pass compiler)



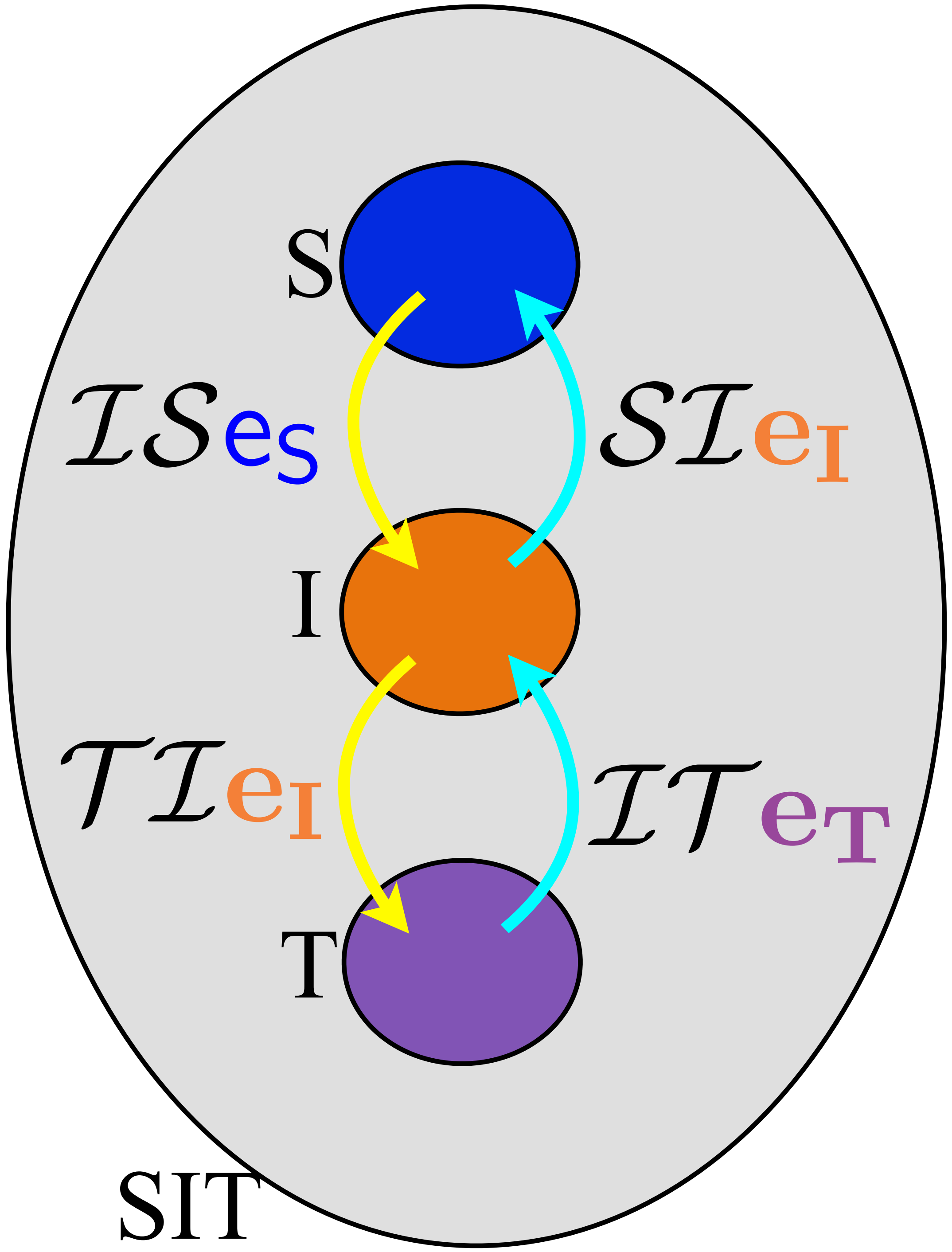
Our Approach (multi-pass compiler)



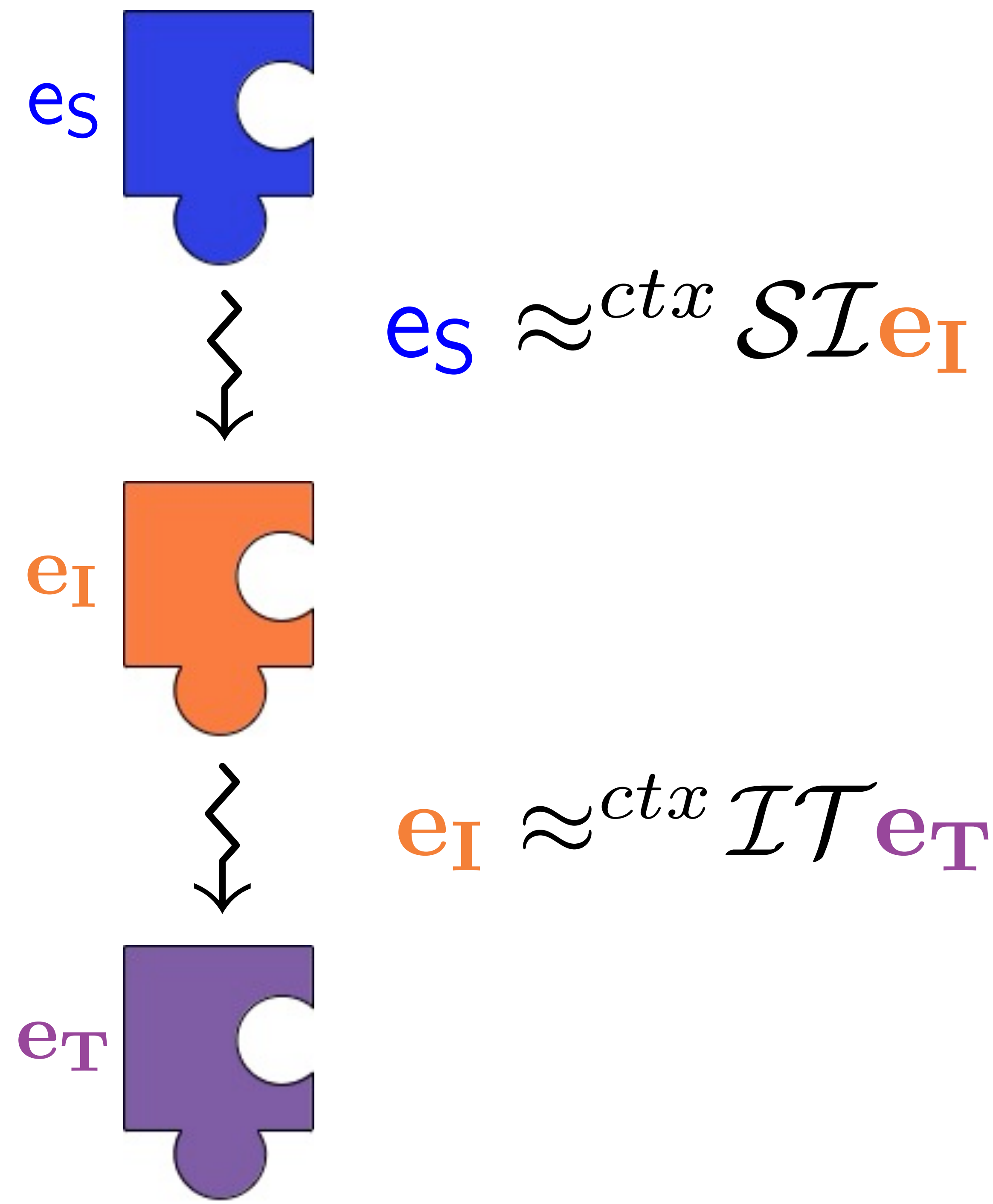
Our Approach (multi-pass compiler)



Our Approach (multi-pass compiler)

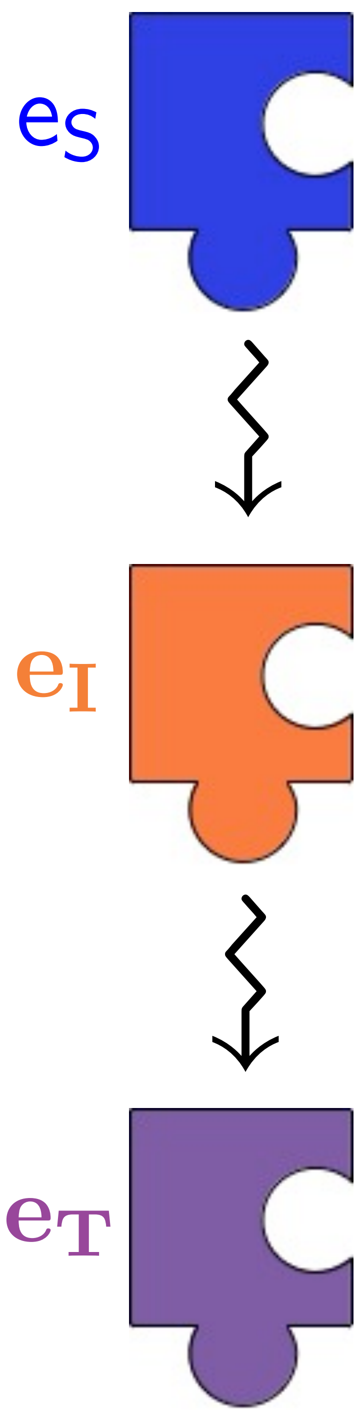


Compiler Correctness



Our Approach

Compiler Correctness

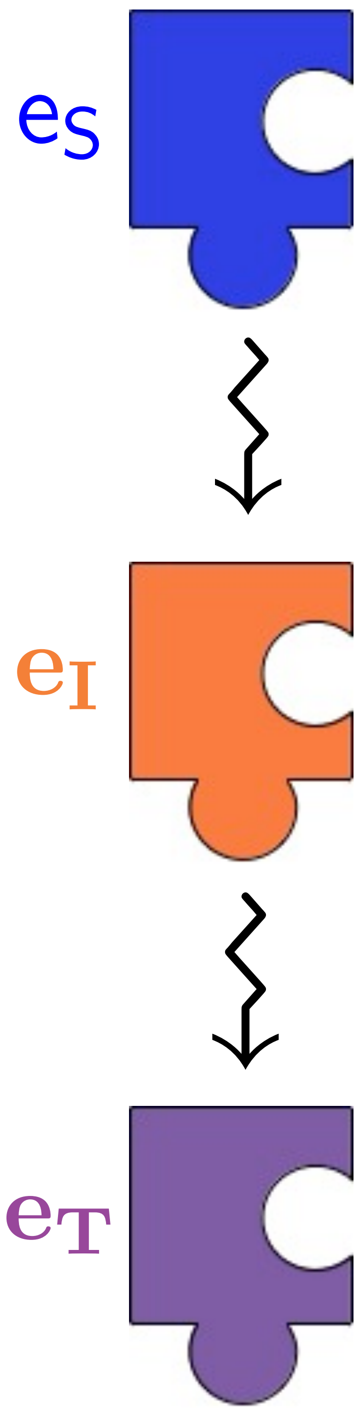


$$e_S \approx^{ctx}_{\mathcal{SI}} e_I$$

$$e_I \approx^{ctx}_{\mathcal{IT}} e_T$$

Our Approach

Compiler Correctness



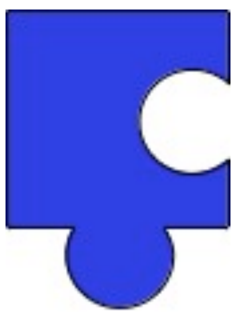
$$e_S \approx^{ctx} \mathcal{SI}\mathbf{e_I}$$

$$\mathcal{SI}\mathbf{e_I} \approx^{ctx} \mathcal{SI}(\mathcal{IT}\mathbf{e_T})$$

Our Approach

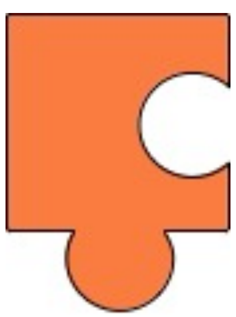
Compiler Correctness

e_S



$$e_S \approx^{ctx} \mathcal{S}\mathcal{I}\, e_I$$

e_I

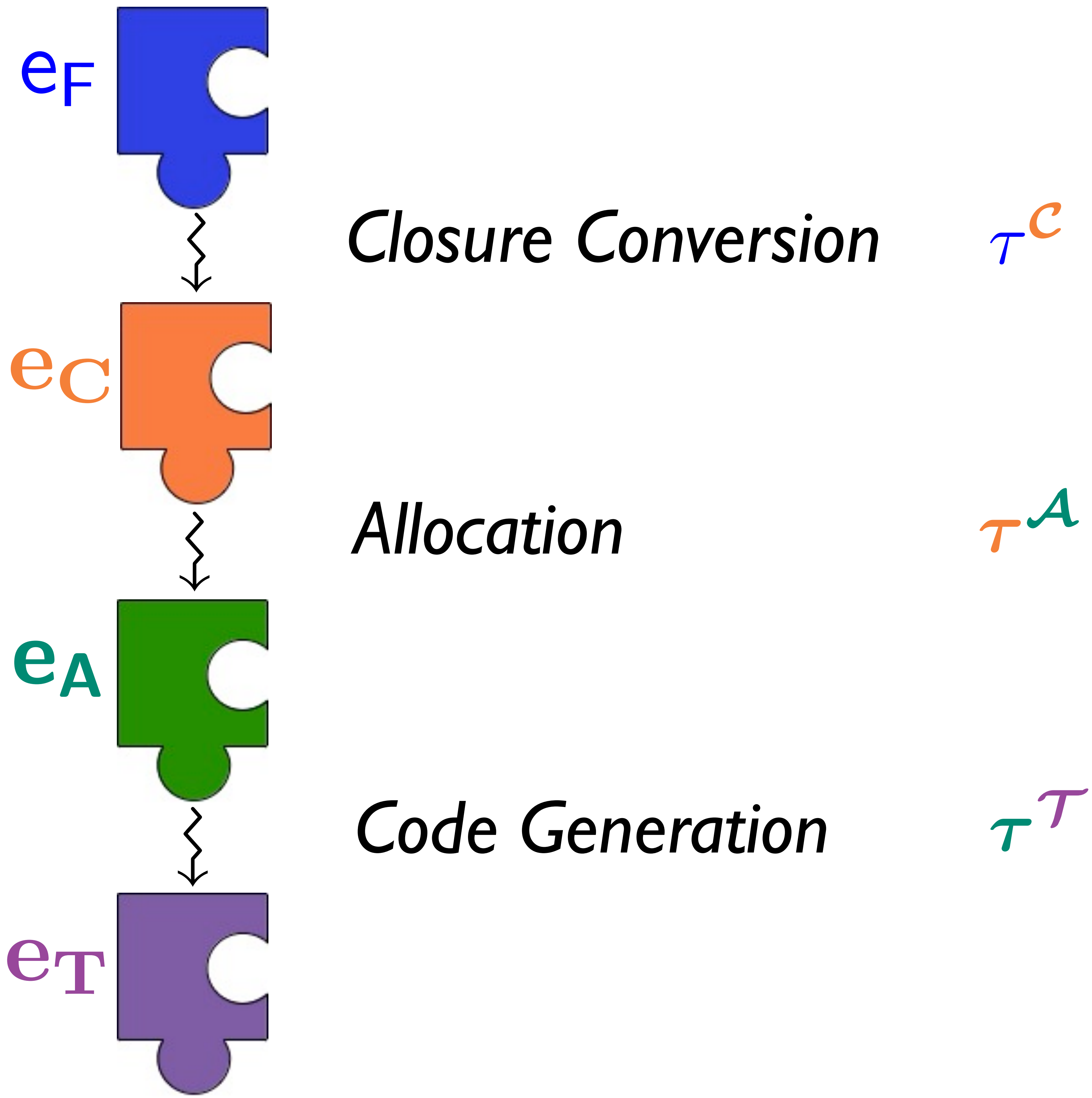


$$\mathcal{S}\mathcal{I}\, e_I \approx^{ctx} \mathcal{S}\mathcal{I}(\mathcal{I}\mathcal{T}\, e_T)$$

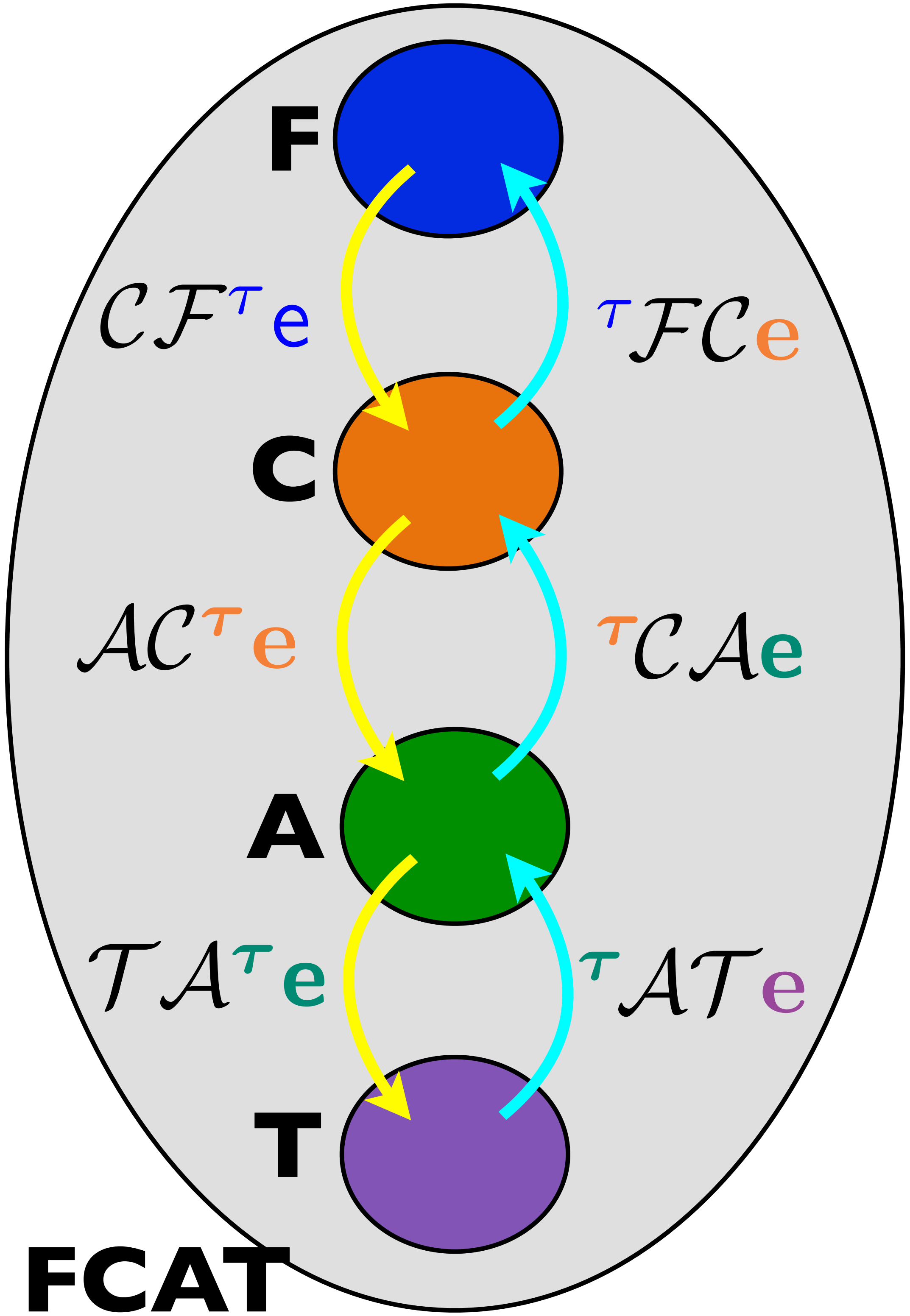
e_T

$$e_S \approx^{ctx} \mathcal{S}\mathcal{I}\mathcal{T}\, e_T$$

Our Compiler: System F to TAL

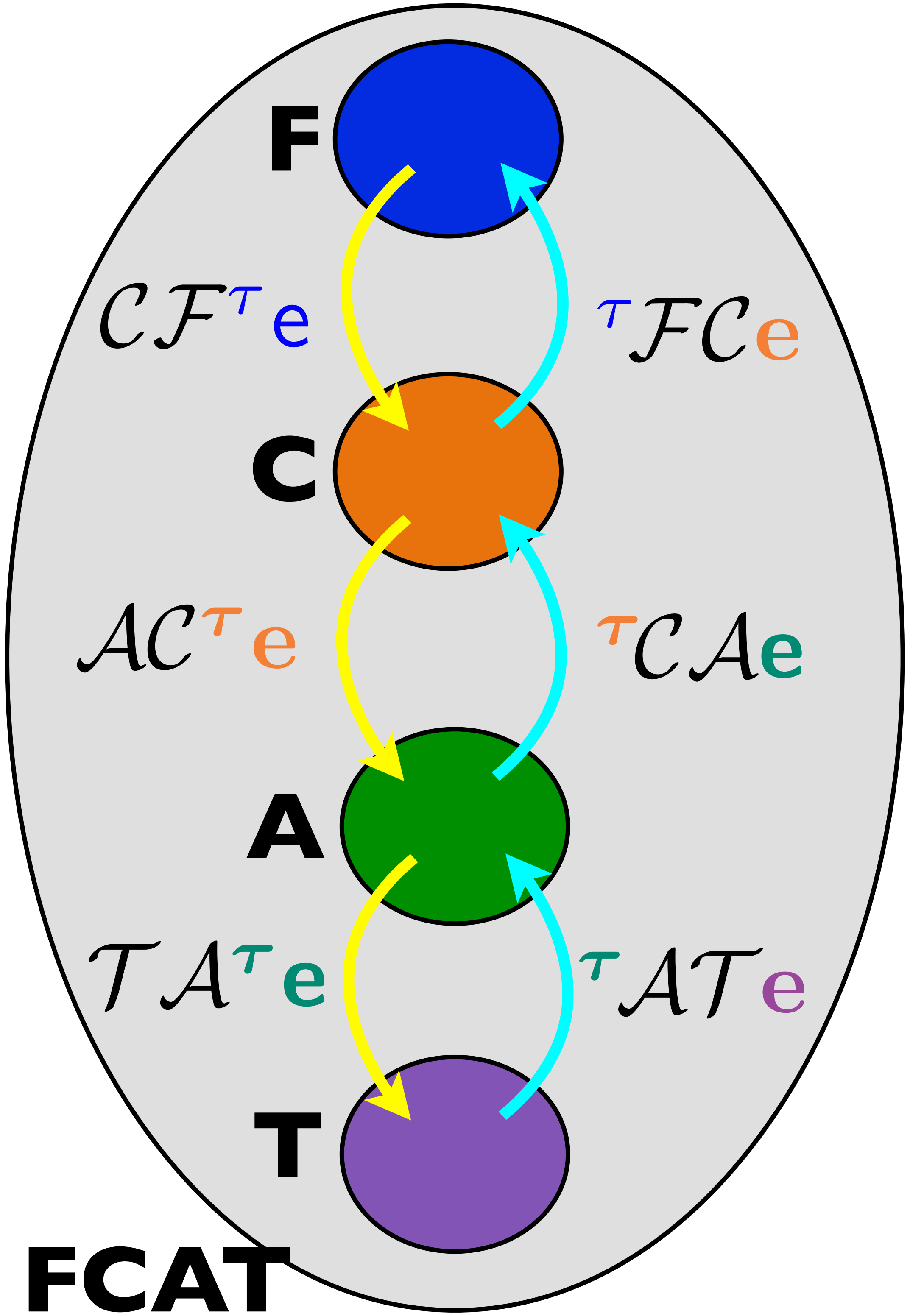


Combined language **FCAT**



- Boundaries mediate between
  - τ & $\tau^{\mathcal{C}}$ τ & $\tau^{\mathcal{A}}$ τ & $\tau^{\mathcal{T}}$

Combined language **FCAT**

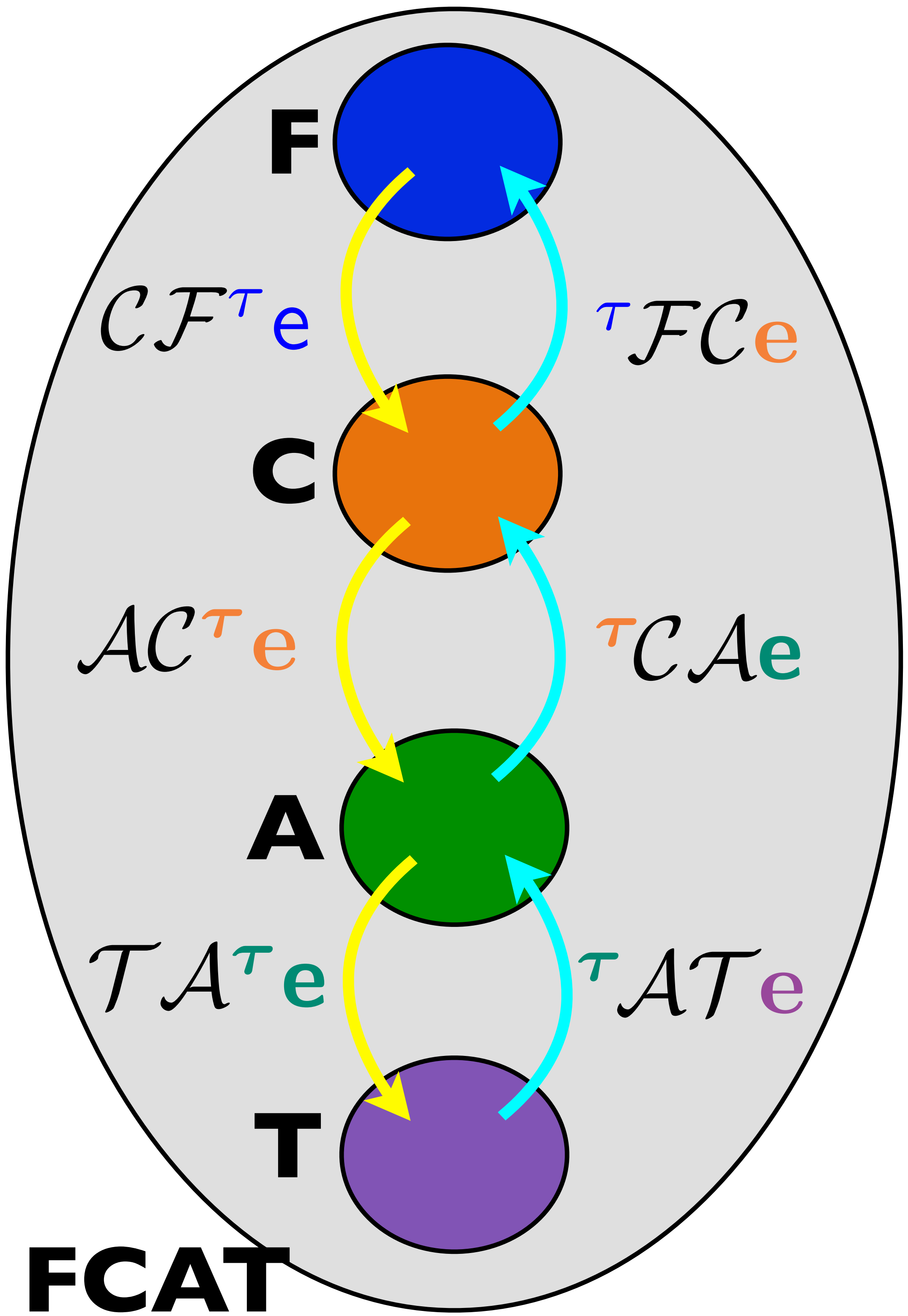


- Boundaries mediate between
  - τ & $\tau^{\mathcal{C}}$ τ & $\tau^{\mathcal{A}}$ τ & $\tau^{\mathcal{T}}$
- Operational semantics

$$\mathcal{CF}^{\tau}\mathsf{e} \longmapsto^{*} \mathcal{CF}^{\tau}\mathsf{v} \longmapsto \mathbf{v}$$

$$^{\tau}\mathcal{FC}\mathbf{e} \longmapsto^{*} {}^{\tau}\mathcal{FC}\mathbf{v} \longmapsto \mathsf{v}$$

Combined language **FCAT**



- Boundaries mediate between
  - τ & $\tau^{\mathcal{C}}$ τ & $\tau^{\mathcal{A}}$ τ & $\tau^{\mathcal{T}}$
- Operational semantics

$$\mathcal{CF}^{\tau}\mathsf{e} \longmapsto^{*} \mathcal{CF}^{\tau}\mathsf{v} \longmapsto \mathbf{v}$$

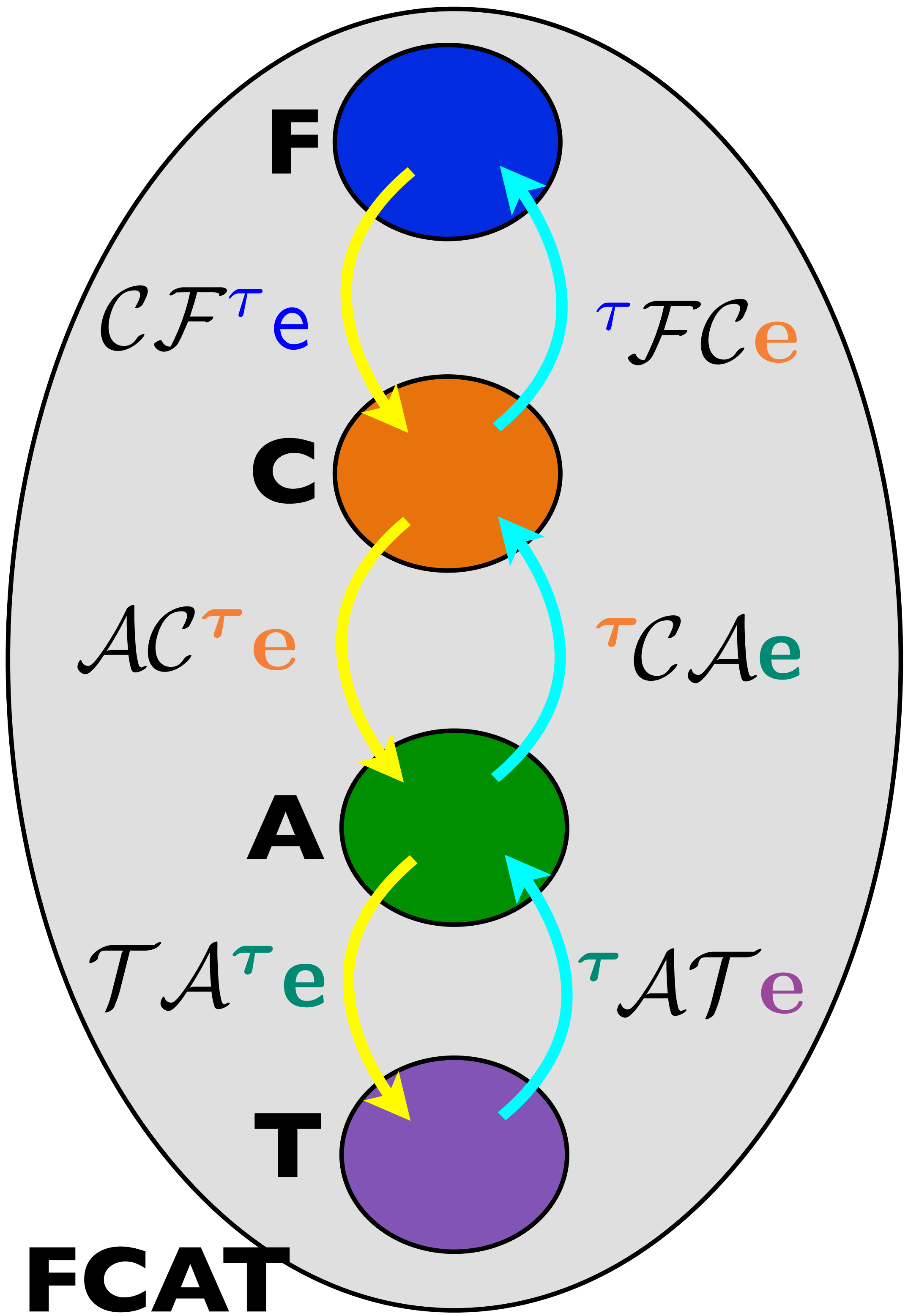
$$^{\tau}\mathcal{FC}\mathbf{e} \longmapsto^{*} {}^{\tau}\mathcal{FC}\mathbf{v} \longmapsto \mathsf{v}$$

- Boundary cancellation

$$^{\tau}\mathcal{FC}\mathcal{CF}^{\tau}\mathsf{e} \approx^{ctx} \mathsf{e} : \tau$$

$$\mathcal{CF}^{\tau}\,{}^{\tau}\mathcal{FC}\mathbf{e} \approx^{ctx} \mathbf{e} : \tau^{\mathcal{C}}$$

Challenges / Roadmap for rest of talk

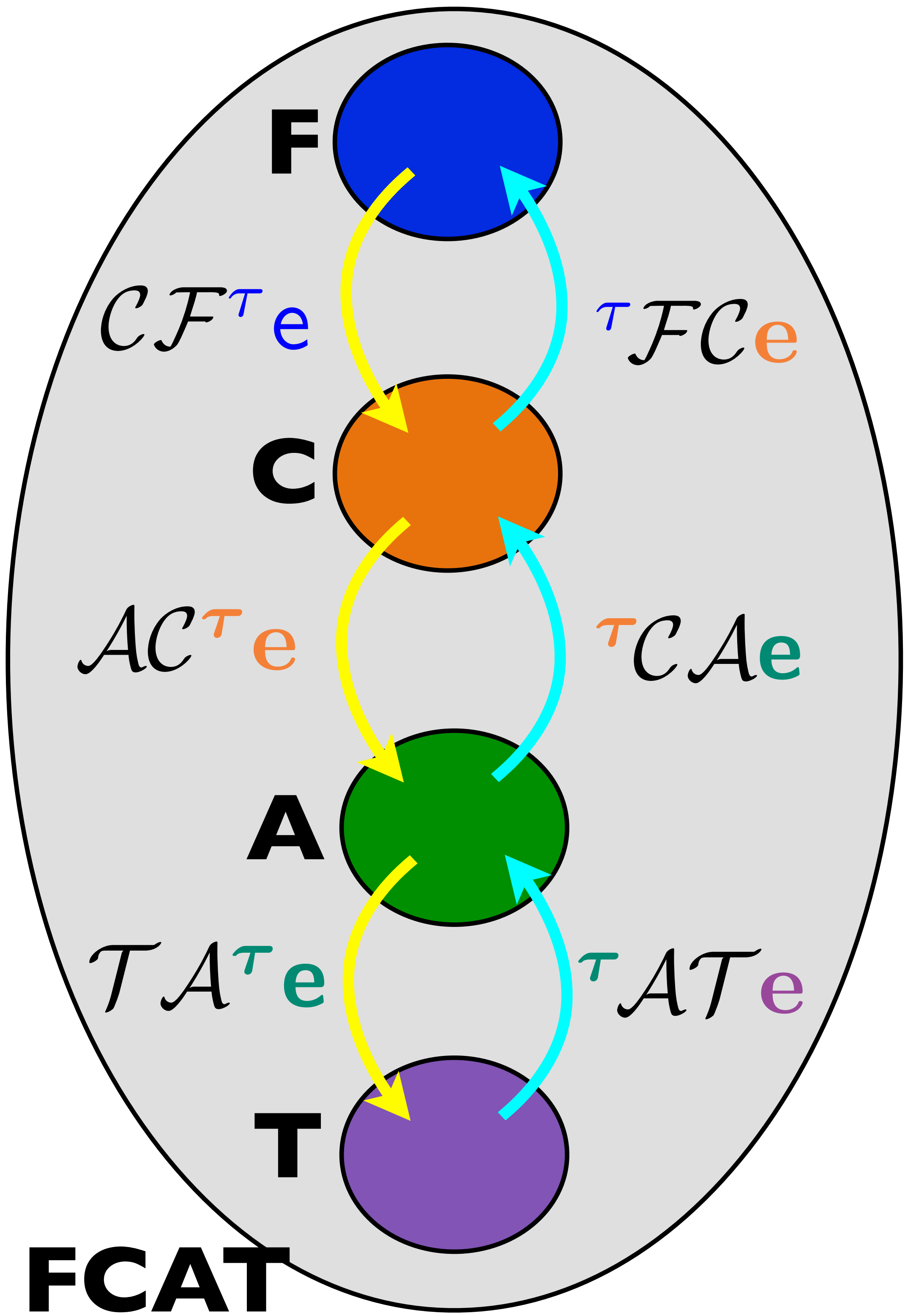


F+C: Interoperability semantics with type abstraction in both languages

C+A: Interoperability when compiler pass allocates code & tuples on heap

A+T: What is e? What is $\mathbf{v}$? How to define contextual equiv. for TAL *components*? How to define logical relation?

Challenges / Roadmap for rest of talk



F+C: Interoperability semantics with type abstraction in both languages

C+A: Interoperability when compiler pass allocates code & tuples on heap

A+T: What is e? What is $\mathbf{v}$? How to define contextual equiv. for TAL *components*? How to define logical relation?

Abstract Types & Interoperability

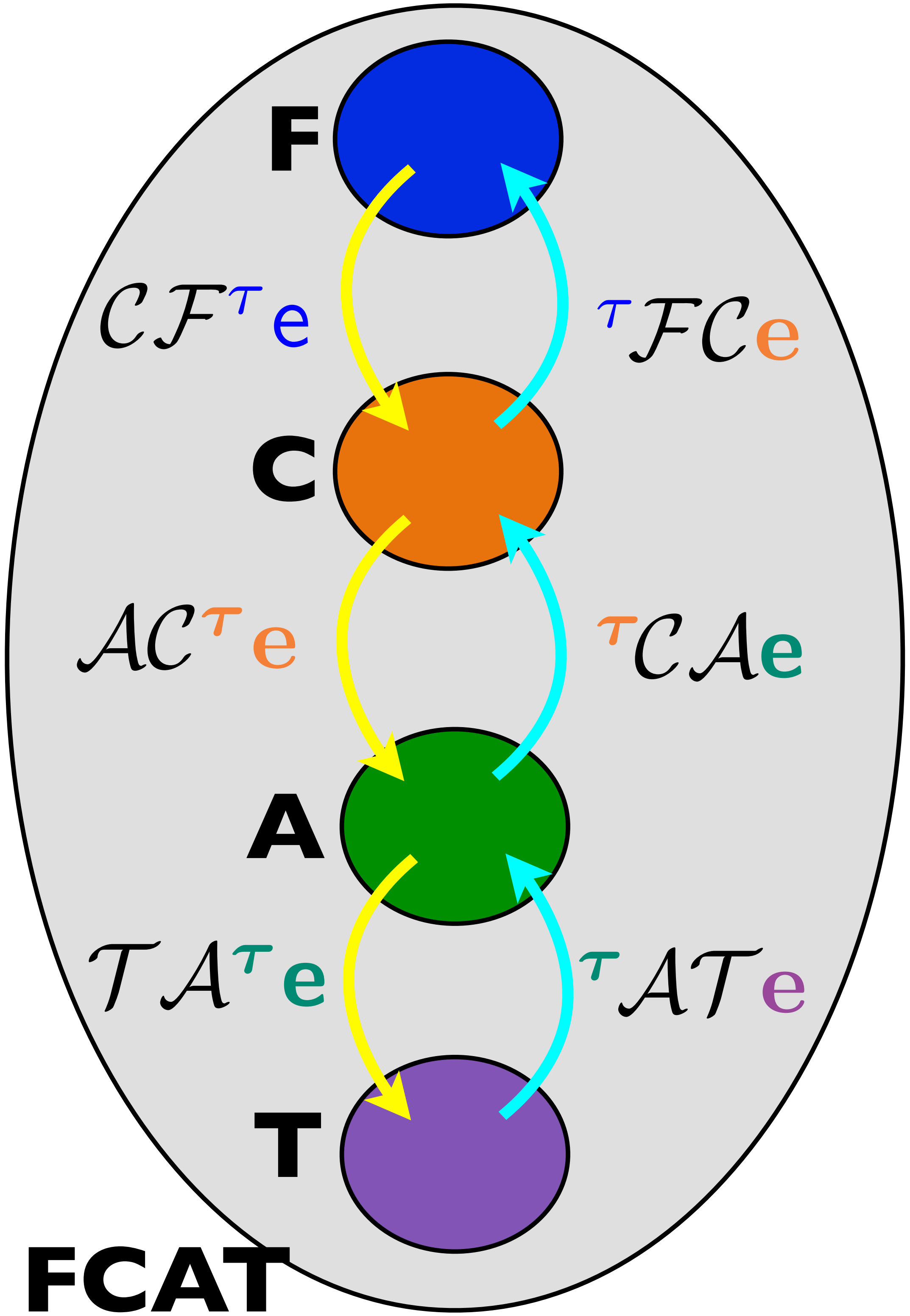
Add new type $\mathsf{L}\langle\boldsymbol{\tau}\rangle$ & new value form ${}^{\mathsf{L}\langle\boldsymbol{\tau}\rangle}\mathcal{FC}\mathbf{v}$

Add new type $\lceil\alpha\rceil$ & define $\lceil\alpha\rceil[\tau/\alpha] = \tau^{\langle\mathcal{C}\rangle}$

Requires novel admissibility relations in logical relation.

(draft paper: www.ccs.neu.edu/home/amal/voc.pdf)

Challenges / Roadmap

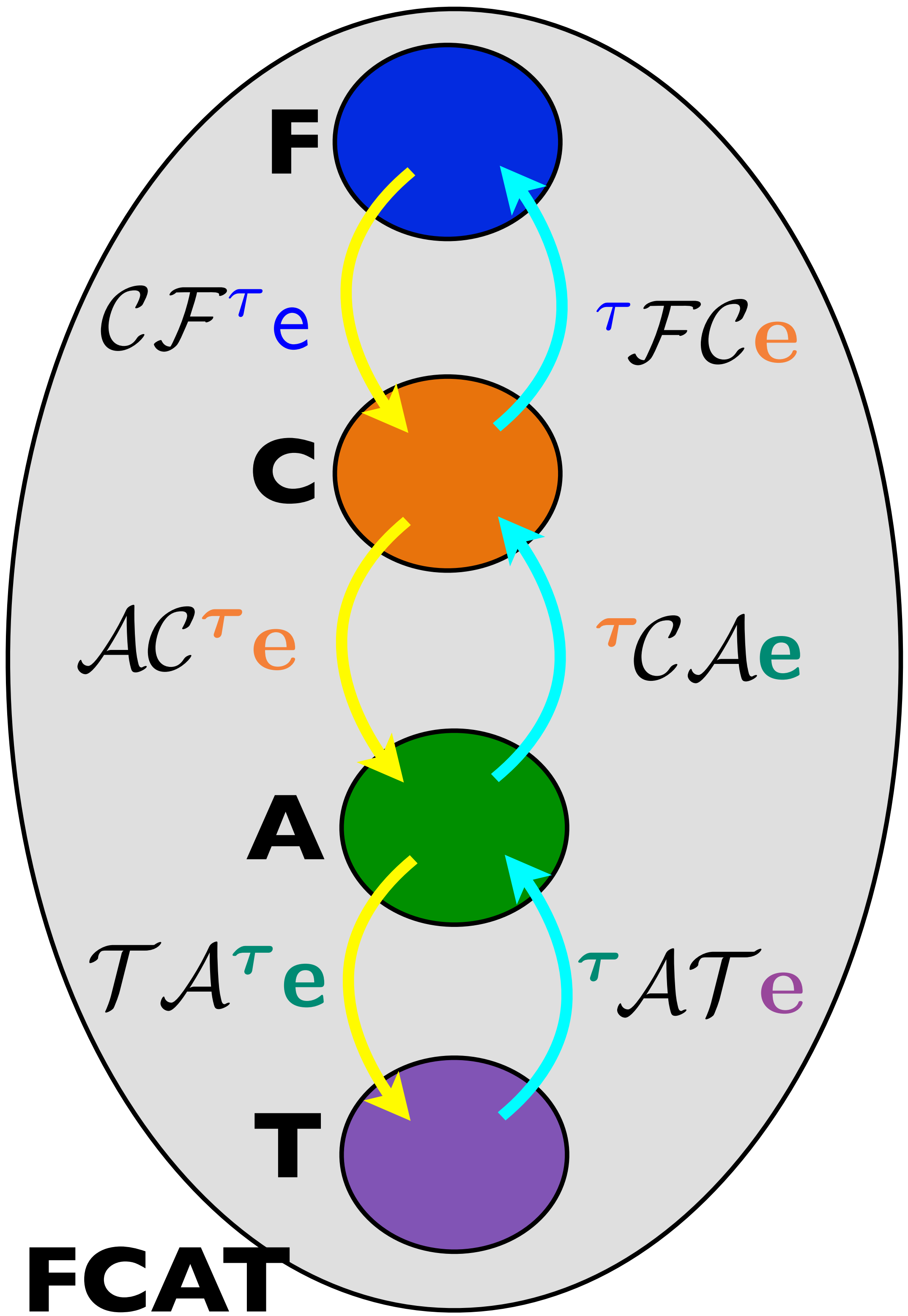


F+C: Interoperability semantics with type abstraction in both languages

C+A: Interoperability when compiler pass allocates code & tuples on heap

A+T: What is e? What is v? How to define contextual equiv. for TAL *components*? How to define logical relation?

Challenges / Roadmap



F+C: Interoperability semantics with type abstraction in both languages

C+A: Interoperability when compiler pass allocates code & tuples on heap

A+T: What is e? What is v? How to define contextual equiv. for TAL *components*? How to define logical relation?

A

$$\tau ::= \alpha \mid \mathsf{unit} \mid \mathsf{int} \mid \exists\alpha.\tau \mid \mu\alpha.\tau \mid \mathsf{box}\,\psi$$

$$\psi ::= \forall[\overline{\alpha}].(\overline{\tau}) \to \tau \mid \langle \tau, \ldots, \tau \rangle$$

$$\mathsf{e} ::= (\mathsf{t}, \mathsf{H}) \mid \mathsf{t}$$

$$\mathsf{t} ::= \mathsf{x} \mid () \mid \mathsf{n} \mid \mathsf{t\ p\ t} \mid \mathsf{if0\ t\ t\ t} \mid \ell \mid \mathsf{t}\,[]\,\bar{\mathsf{t}} \mid \mathsf{t}[\tau]$$
$$\mid \mathsf{pack}\langle \tau, \mathsf{t} \rangle\ \mathsf{as}\ \exists\alpha.\tau \mid \mathsf{unpack}\ \langle \alpha, \mathsf{x} \rangle = \mathsf{t\ in\ t} \mid \mathsf{fold}_{\mu\alpha.\tau}\ \mathsf{t}$$
$$\mid \mathsf{unfold\ t} \mid \mathsf{balloc}\ \langle \bar{\mathsf{t}} \rangle \mid \mathsf{read}[\mathsf{i}]\ \mathsf{t}$$

$$\mathsf{p} ::= + \mid - \mid *$$

$$\mathsf{v} ::= () \mid \mathsf{n} \mid \mathsf{pack}\langle \tau, \mathsf{v} \rangle\ \mathsf{as}\ \exists\alpha.\tau \mid \mathsf{fold}_{\mu\alpha.\tau}\ \mathsf{v} \mid \ell \mid \mathsf{v}[\tau]$$

$$\mathsf{H} ::= \cdot \mid \mathsf{H}, \ell \mapsto \mathsf{h}$$

$$\mathsf{h} ::= \lambda[\overline{\alpha}]\,(\overline{\mathsf{x} : \tau}).\mathsf{t} \mid \langle \mathsf{v}, \ldots, \mathsf{v} \rangle$$

$\boxed{\langle \mathsf{H} \mid \mathsf{e} \rangle \longmapsto \langle \mathsf{H}' \mid \mathsf{e}' \rangle}$ Reduction Relation (selected cases)

$$\langle \mathsf{H} \mid (\mathsf{t}, \mathsf{H}') \rangle \longmapsto \langle (\mathsf{H}, \mathsf{H}') \mid \mathsf{t} \rangle \qquad \mathrm{dom}(\mathsf{H}) \cap \mathrm{dom}(\mathsf{H}') = \emptyset$$

$$\langle \mathsf{H} \mid \mathsf{E}[\ell\ [\overline{\tau'}]\ \overline{\mathsf{v}}] \rangle \longmapsto \langle \mathsf{H} \mid \mathsf{E}[\mathsf{t}[\overline{\tau'}/\overline{\alpha}][\overline{\mathsf{v}}/\overline{\mathsf{x}}]] \rangle \quad \mathsf{H}(\ell) = \lambda[\overline{\alpha}]\,(\overline{\mathsf{x} : \tau}).\mathsf{t}$$

T

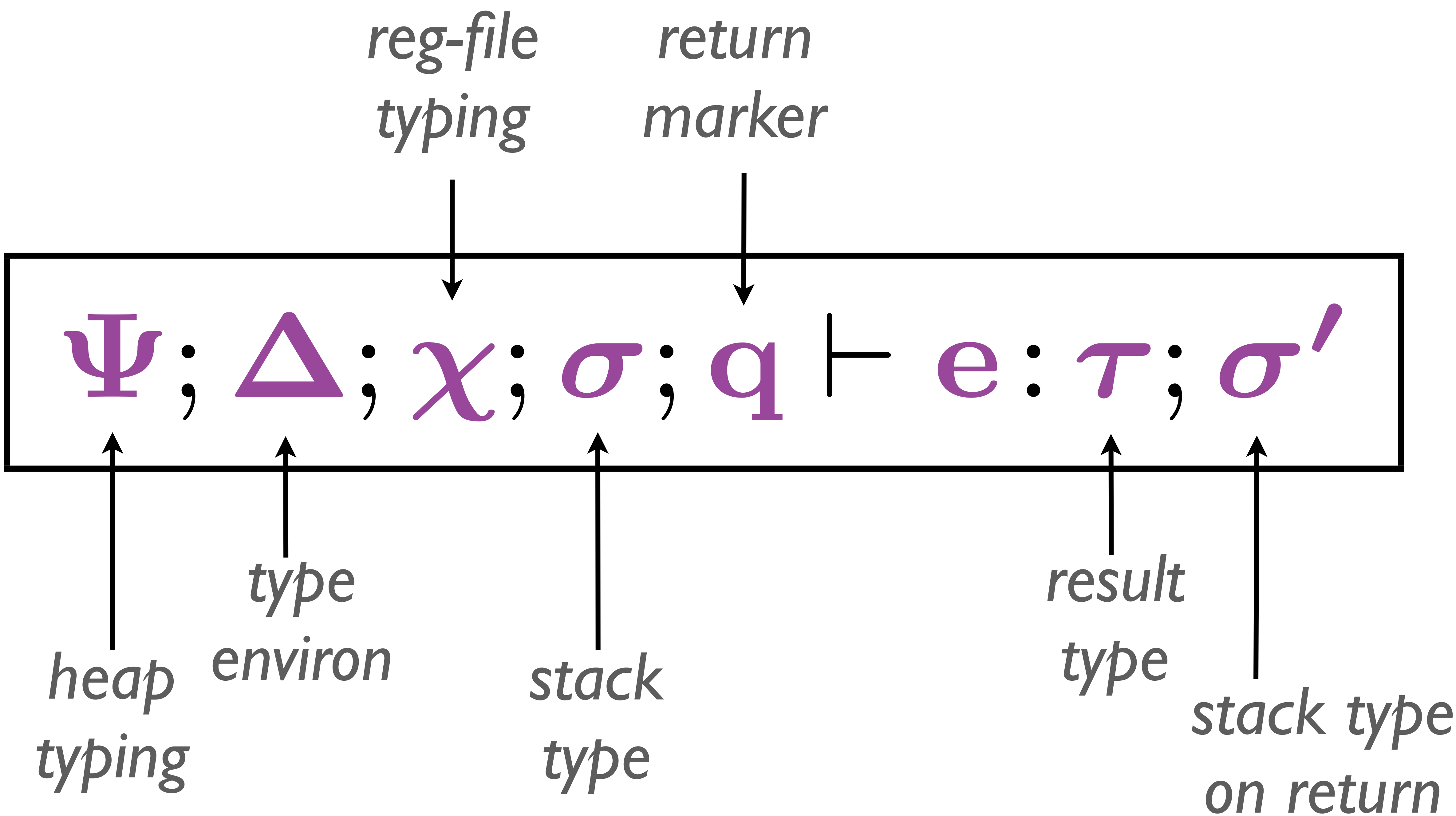
$$
\begin{array}{llll}
\tau & ::= & \alpha \mid \mathbf{unit} \mid \mathbf{int} \mid \exists\alpha.\tau \mid \mu\alpha.\tau & \textit{Type} \\
 & \mid & \mathbf{ref}\ \langle\tau,\ldots,\tau\rangle \mid \mathbf{box}\ \psi & \\
\psi & ::= & \forall[\Delta].\{\chi;\sigma\}^{\mathrm{q}} \mid \langle\tau,\ldots,\tau\rangle & \textit{Heap value type} \\
\chi & ::= & \cdot \mid \chi, \mathrm{r}:\tau & \textit{Register file type} \\
\sigma & ::= & \zeta \mid \bullet \mid \tau :: \sigma & \textit{Stack type} \\
\mathrm{q} & ::= & \epsilon \mid \mathrm{r} \mid \mathrm{i} \mid \mathrm{end}[\tau;\sigma] & \textit{Return marker} \\
\Delta & ::= & \cdot \mid \Delta,\alpha \mid \Delta,\zeta \mid \Delta,\epsilon & \textit{Type variable environment} \\
\omega & ::= & \tau \mid \sigma \mid \mathrm{q} & \textit{Instantiation of type variable} \\
\mathrm{r} & ::= & \mathtt{r1} \mid \mathtt{r2} \mid \cdots \mid \mathtt{r7} \mid \mathtt{ra} & \textit{Register} \\
\mathrm{h} & ::= & \mathrm{code}[\Delta]\{\chi;\sigma\}^{\mathrm{q}}.\mathrm{I} \mid \langle\mathrm{w},\ldots,\mathrm{w}\rangle & \textit{Heap value} \\
\mathrm{w} & ::= & () \mid \mathrm{n} \mid \ell \mid \mathrm{pack}\langle\tau,\mathrm{w}\rangle\ \mathrm{as}\ \exists\alpha.\tau & \textit{Word value} \\
 & \mid & \mathrm{fold}_{\mu\alpha.\tau}\ \mathrm{w} \mid \mathrm{w}[\omega] & \\
\mathrm{u} & ::= & \mathrm{w} \mid \mathrm{r} \mid \mathrm{pack}\langle\tau,\mathrm{u}\rangle\ \mathrm{as}\ \exists\alpha.\tau & \textit{Small value} \\
 & \mid & \mathrm{fold}_{\mu\alpha.\tau}\ \mathrm{u} \mid \mathrm{u}[\omega] & \\
\mathrm{I} & ::= & \iota;\mathrm{I} \mid \mathtt{jmp}\ \mathrm{u} \mid \mathtt{ret}\ \mathrm{q},\mathrm{r} & \textit{Instruction sequence}
\end{array}
$$

T

$$
\begin{array}{rcll}
\iota & ::= & \mathtt{aop}\ \mathbf{r_d}, \mathbf{r_s}, \mathbf{u} \mid \mathtt{bnz}\ \mathbf{r}, \mathbf{u} \mid \mathtt{mv}\ \mathbf{r_d}, \mathbf{u} & \textit{Instruction} \\
& \mid & \mathtt{ralloc}\ \mathbf{r_d}, \mathbf{n} \mid \mathtt{balloc}\ \mathbf{r_d}, \mathbf{n} \mid \mathtt{ld}\ \mathbf{r_d}, \mathbf{r_s}[\mathbf{i}] \mid \mathtt{st}\ \mathbf{r_d}[\mathbf{i}], \mathbf{r_s} & \\
& \mid & \mathtt{unpack}\ \langle \alpha, \mathbf{r_d} \rangle\ \mathbf{u} \mid \mathtt{unfold}\ \mathbf{r_d}, \mathbf{u} \mid \mathtt{salloc}\ \mathbf{n} \mid \mathtt{sfree}\ \mathbf{n} & \\
& \mid & \mathtt{sld}\ \mathbf{r_d}, \mathbf{i} \mid \mathtt{sst}\ \mathbf{i}, \mathbf{r_s} & \\
\mathbf{aop} & ::= & \mathtt{add} \mid \mathtt{sub} \mid \mathtt{mult} & \textit{Arithmetic operation} \\
\mathbf{e} & ::= & (\mathbf{I}, \mathbf{H}) \mid \mathbf{I} & \textit{Component} \\
\mathbf{v} & ::= & \mathtt{ret}\ \mathbf{q}, \mathbf{r} & \textit{Term value} \\
\mathbf{E} & ::= & (\mathbf{E_I}, \cdot) & \textit{Evaluation context} \\
\mathbf{E_I} & ::= & [\cdot] & \textit{Instruction evaluation context} \\
\mathbf{H} & ::= & \cdot \mid \mathbf{H}, \ell \mapsto \mathbf{h} & \textit{Heap or Heap fragment} \\
\mathbf{R} & ::= & \cdot \mid \mathbf{R}, \mathbf{r} \mapsto \mathbf{w} & \textit{Register file} \\
\mathbf{S} & ::= & \mathbf{nil} \mid \mathbf{w} :: \mathbf{S} & \textit{Stack} \\
\mathbf{M} & ::= & (\mathbf{H}, \mathbf{R}, \mathbf{S} : \sigma) & \textit{Memory}
\end{array}
$$

$$
\boxed{\langle \mathbf{M} \mid \mathbf{e} \rangle \longmapsto \langle \mathbf{M'} \mid \mathbf{e'} \rangle}
$$

Typing TAL Components



Well-typed Components in $\mathbf{T}$

$$\boxed{\Psi; \Delta; \chi; \sigma; \mathbf{q} \vdash \mathbf{e} : \tau; \sigma'}$$

$$\frac{\begin{array}{c}\Psi \vdash \mathbf{H} : \Psi_{\mathbf{e}} \qquad \text{boxheap}(\Psi_{\mathbf{e}}) \\ \text{ret-type}(\mathbf{q}, \chi, \sigma) = \tau; \sigma' \qquad (\Psi, \Psi_{\mathbf{e}}); \Delta; \chi; \sigma; \mathbf{q} \vdash \mathbf{I}\end{array}}{\Psi; \Delta; \chi; \sigma; \mathbf{q} \vdash (\mathbf{I}, \mathbf{H}) : \tau; \sigma'}$$

Well-typed Instruction Sequence

$$\boxed{\Psi; \Delta; \chi; \sigma; \mathrm{q} \vdash \mathrm{I}} \quad \text{where } \mathrm{q} \neq \epsilon$$

$$\frac{\Psi; \Delta; \chi; \sigma; \mathrm{q} \vdash \iota \Rightarrow \Delta'; \chi'; \sigma'; \mathrm{q}' \qquad \Psi; \Delta'; \chi'; \sigma'; \mathrm{q}' \vdash \mathrm{I}}{\Psi; \Delta; \chi; \sigma; \mathrm{q} \vdash \iota; \mathrm{I}}$$

$$\frac{\chi(\mathrm{r}) = \mathrm{box}\, \forall[].\{\mathrm{r}' : \tau; \sigma\}^{\mathrm{q}'} \qquad \chi(\mathrm{r}') = \tau}{\Psi; \Delta; \chi; \sigma; \mathrm{r} \vdash \mathtt{ret}\; \mathrm{r}, \mathrm{r}'}$$

$$\frac{\chi(\mathrm{r}) = \tau}{\Psi; \Delta; \chi; \sigma; \mathrm{end}[\tau; \sigma] \vdash \mathtt{ret}\; \mathrm{end}[\tau; \sigma], \mathrm{r}}$$

Jmp

To next code block within component:

$$\frac{\Psi; \Delta; \chi \vdash u : \mathrm{box}\, \forall[].\{\chi'; \sigma\}^{q} \qquad \Delta \vdash \chi \leq \chi'}{\Psi; \Delta; \chi; \sigma; q \vdash \mathtt{jmp}\, u}$$

Call subroutine:

- must protect current return addr, by storing it in tail part of stack that is parametrically hidden from subroutine

$$\frac{\begin{array}{c}\Psi; \Delta; \chi \vdash u : \mathrm{box}\, \forall[\zeta, \epsilon].\{\hat{\chi}; \hat{\sigma}\}^{\hat{q}} \qquad \text{ret-addr-type}(\hat{q}, \hat{\chi}, \hat{\sigma}) = \forall[].\{r : \tau; \hat{\sigma}'\}^{\epsilon} \\ \Delta \vdash \sigma_0 \qquad \Delta \vdash \forall[].\{\hat{\chi}[\sigma_0/\zeta][i{+}k{-}j/\epsilon]; \hat{\sigma}[\sigma_0/\zeta][i{+}k{-}j/\epsilon]\}^{\hat{q}} \qquad \Delta \vdash \chi \leq \hat{\chi}[\sigma_0/\zeta][i{+}k{-}j/\epsilon] \\ \sigma = \tau_0 :: \cdots :: \tau_j :: \sigma_0 \qquad \hat{\sigma} = \tau_0 :: \cdots :: \tau_j :: \zeta \qquad j < i \qquad \hat{\sigma}' = \tau_0' :: \cdots :: \tau_k' :: \zeta\end{array}}{\Psi; \Delta; \chi; \sigma; i \vdash \mathtt{jmp}\, u[\sigma_0, i{+}k{-}j]}$$

Instruction Typing

Instructions must not clobber return address:

$$\frac{\Psi; \Delta; \chi \vdash \mathbf{u} : \tau \qquad \mathbf{q} \neq \mathbf{r_d}}{\Psi; \Delta; \chi; \sigma; \mathbf{q} \vdash \texttt{mv}\ \mathbf{r_d}, \mathbf{u} \Rightarrow \Delta; \chi[\mathbf{r_d} : \tau]; \sigma; \mathbf{q}}$$

Can move return address elsewhere:

$$\frac{\Psi; \Delta; \chi \vdash \mathbf{u} : \tau}{\Psi; \Delta; \chi; \sigma; \mathbf{r_s} \vdash \texttt{mv}\ \mathbf{r_d}, \mathbf{r_s} \Rightarrow \Delta; \chi[\mathbf{r_d} : \tau]; \sigma; \mathbf{r_d}}$$

Equivalence of **T** Components: Tricky!

Logical relations: related inputs to related outputs

$$\mathcal{V}[\![\tau_1 \to \tau_2]\!] = \{(W, \lambda x.e_1, \lambda x.e_1) \mid \ldots\}$$

$$\mathcal{HV}[\![\forall[\Delta].\{\chi;\sigma\}^q]\!] = \{(W, \mathbf{code}[\Delta]\{\chi;\sigma\}^q.\mathbf{I_1}, \mathbf{code}[\Delta]\{\chi;\sigma\}^q.\mathbf{I_2}) \mid \ldots\}$$

Equivalence of **T** Components: Tricky!

Logical relations: related inputs to related outputs

$$\mathcal{V}[\![\tau_1 \to \tau_2]\!] = \{(W, \lambda \mathsf{x}.\mathsf{e}_1, \lambda \mathsf{x}.\mathsf{e}_1) \mid \ldots\}$$

$$\mathcal{HV}[\![\forall[\boldsymbol{\Delta}].\{\boldsymbol{\chi}; \boldsymbol{\sigma}\}^{\mathrm{q}}]\!] = \{(W, \mathbf{code}[\boldsymbol{\Delta}]\{\boldsymbol{\chi}; \boldsymbol{\sigma}\}^{\mathrm{q}}.\mathbf{I_1}, \mathbf{code}[\boldsymbol{\Delta}]\{\boldsymbol{\chi}; \boldsymbol{\sigma}\}^{\mathrm{q}}.\mathbf{I_2}) \mid \ldots\}$$

Equivalence of **T** Components: Tricky!

Logical relations: related inputs to related outputs

$$\mathcal{V}[\![\tau_1 \to \tau_2]\!] = \{(W, \lambda x.e_1, \lambda x.e_1) \mid \ldots\}$$

$$\mathcal{HV}[\![\forall[\Delta].\{\chi;\sigma\}^{q}]\!] = \{(W, \mathbf{code}[\Delta]\{\chi;\sigma\}^{q}.\mathbf{I}_1, \mathbf{code}[\Delta]\{\chi;\sigma\}^{q}.\mathbf{I}_2) \mid \ldots\}$$

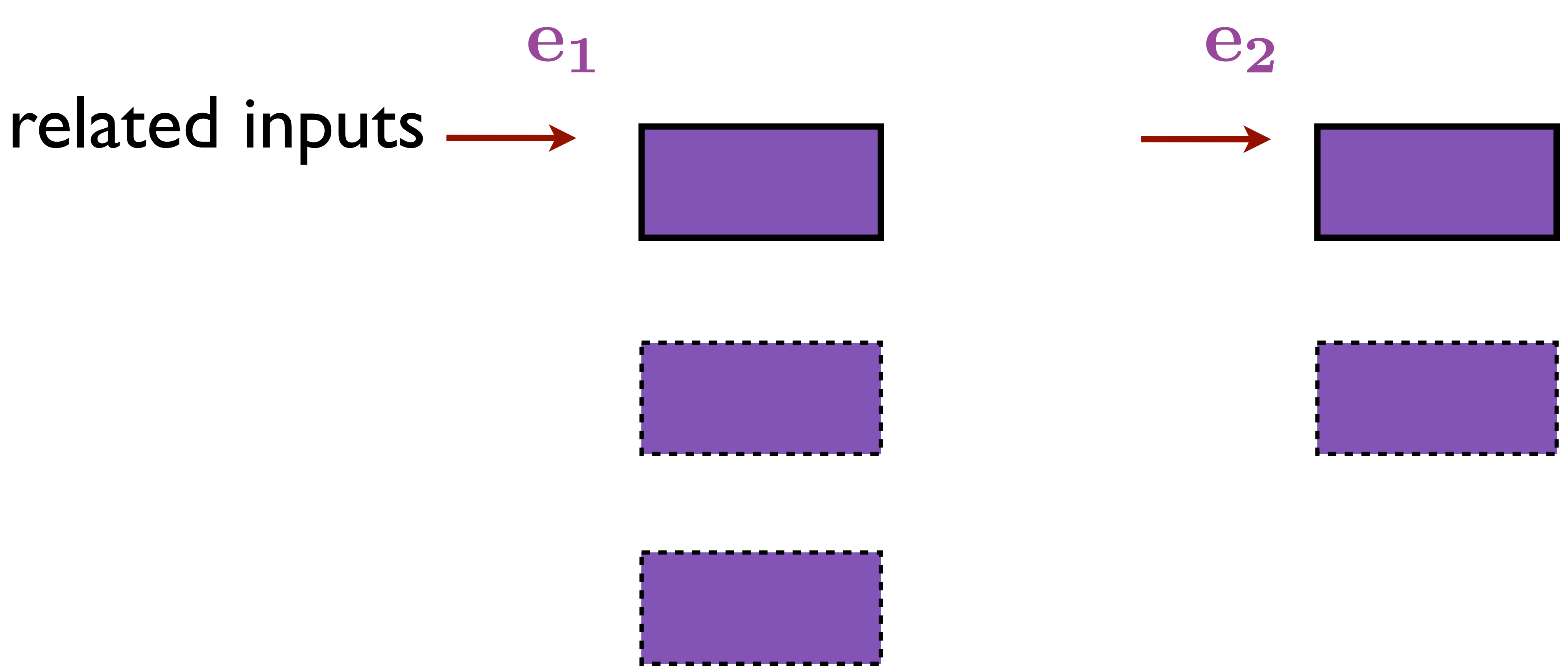

= $\mathbf{code}[\Delta]\{\chi;\sigma\}^{q}.\mathbf{I}$

Equivalence of **T** Components: Tricky!

Logical relations: related inputs to related outputs

$$\mathcal{V}[\![\tau_1 \rightarrow \tau_2]\!] = \{(W, \lambda x.e_1, \lambda x.e_1) \mid \ldots\}$$

$$\mathcal{HV}[\![\forall[\Delta].\{\chi;\sigma\}^{q}]\!] = \{(W, \mathbf{code}[\Delta]\{\chi;\sigma\}^{q}.\mathbf{I_1}, \mathbf{code}[\Delta]\{\chi;\sigma\}^{q}.\mathbf{I_2}) \mid \ldots\}$$

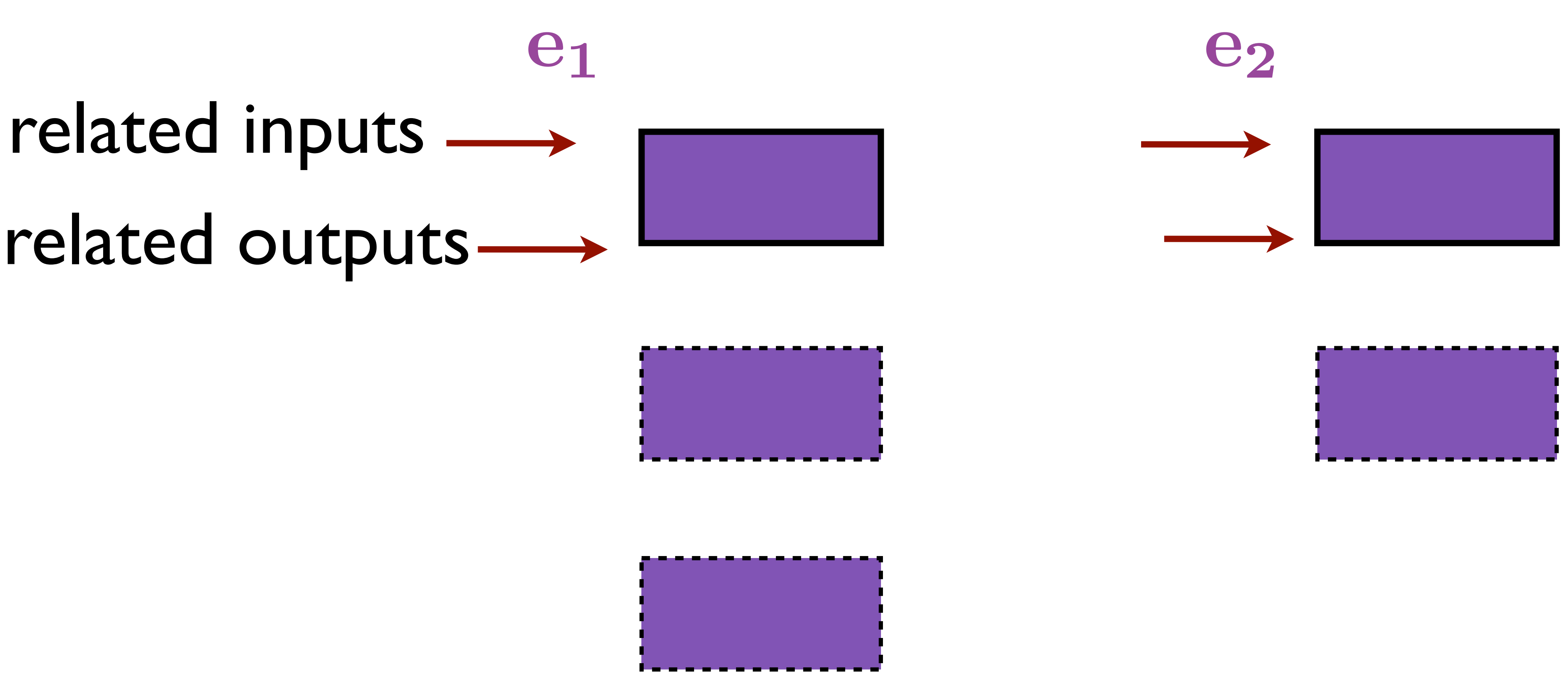


Equivalence of **T** Components: Tricky!

Logical relations: related inputs to related outputs

$$\mathcal{V}[\![\tau_1 \rightarrow \tau_2]\!] = \{(W, \lambda \mathsf{x}.\mathsf{e}_1, \lambda \mathsf{x}.\mathsf{e}_1) \mid \ldots\}$$

$$\mathcal{HV}[\![\forall[\Delta].\{\chi;\sigma\}^{\mathrm{q}}]\!] = \{(W, \mathbf{code}[\Delta]\{\chi;\sigma\}^{\mathrm{q}}.\mathbf{I}_1, \mathbf{code}[\Delta]\{\chi;\sigma\}^{\mathrm{q}}.\mathbf{I}_2) \mid \ldots\}$$

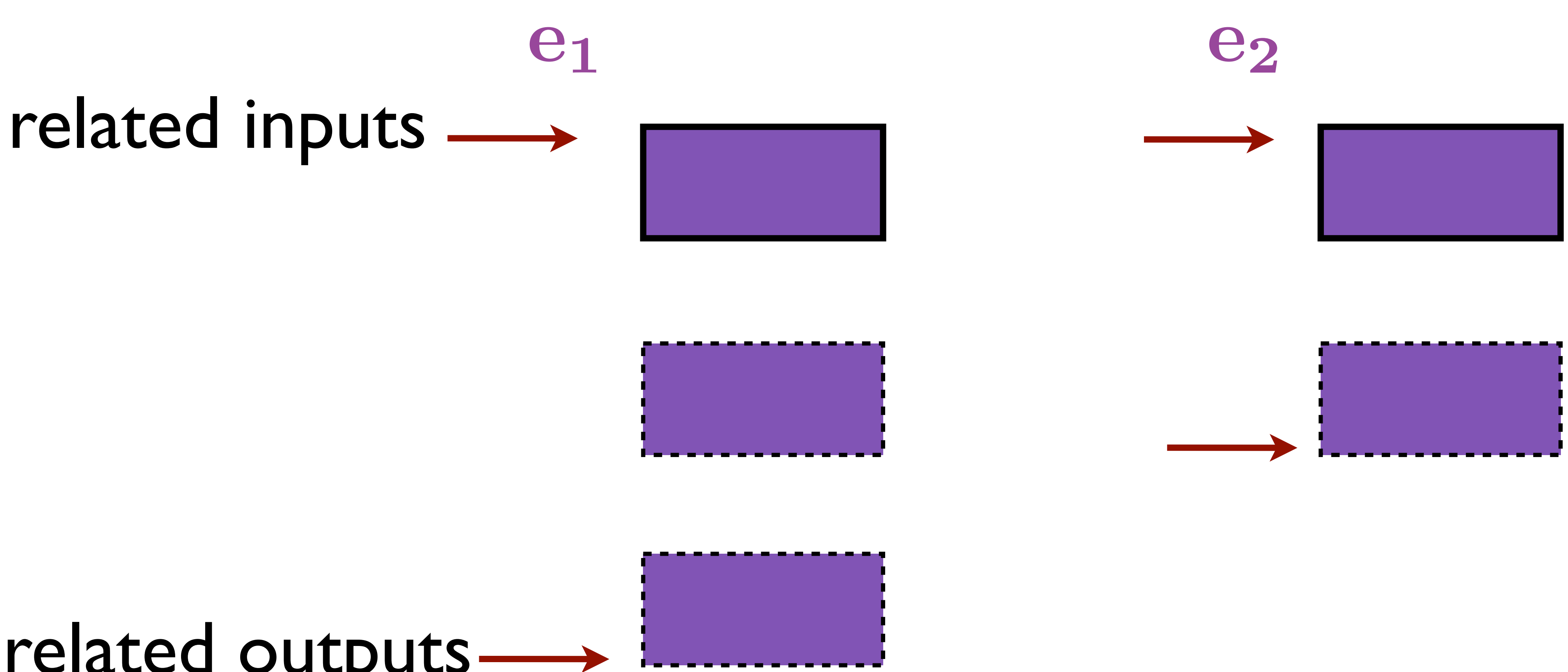


Equivalence of **T** Components: Tricky!

Logical relations: related inputs to related outputs

$$\mathcal{V}[\![\tau_1 \to \tau_2]\!] = \{(W, \lambda \mathsf{x}.\mathsf{e}_1, \lambda \mathsf{x}.\mathsf{e}_1) \mid \ldots\}$$

$$\mathcal{HV}[\![\forall[\Delta].\{\chi; \sigma\}^{\mathrm{q}}]\!] = \{(W, \mathbf{code}[\Delta]\{\chi; \sigma\}^{\mathrm{q}}.\mathbf{I}_1, \mathbf{code}[\Delta]\{\chi; \sigma\}^{\mathrm{q}}.\mathbf{I}_2) \mid \ldots\}$$



Code Generation: **A** to **T**

$\boxed{\tau^{\mathcal{T}}}$ Type translation

$$
\begin{aligned}
&\mathbf{box}\,\forall[\overline{\alpha}].(\tau_1,\ldots,\tau_n)\rightarrow\tau'^{\mathcal{T}}\\
&\quad=\mathbf{box}\,\forall[\overline{\alpha},\zeta,\epsilon].\\
&\qquad\qquad\{\mathtt{ra}:\mathbf{box}\,\forall[].\{\mathtt{r1}:\tau'^{\mathcal{T}};\zeta\}^{\epsilon};\\
&\qquad\qquad\ \tau_n{}^{\mathcal{T}}::\cdots::\tau_1{}^{\mathcal{T}}::\zeta\}^{\mathtt{ra}}
\end{aligned}
$$

Code Generation: **A** to **T**

$\boxed{\tau^{\mathcal{T}}}$ Type translation

$$\begin{aligned}&\mathbf{box}\,\forall[\overline{\alpha}].(\tau_1,\ldots,\tau_n)\to\tau'^{\mathcal{T}}\\&\quad=\mathrm{box}\,\forall[\overline{\alpha},\zeta,\epsilon].\\&\qquad\qquad\{\mathtt{ra}:\mathrm{box}\,\forall[].\{\mathtt{r1}:\tau'^{\mathcal{T}};\zeta\}^{\epsilon};\\&\qquad\qquad\ \tau_n{}^{\mathcal{T}}::\cdots::\tau_1{}^{\mathcal{T}}::\zeta\}^{\mathtt{ra}}\end{aligned}$$

$\boxed{\Psi;\Delta;\Gamma\vdash \mathbf{e}:\tau\rightsquigarrow \mathrm{e}}$

$$\Psi^{\mathcal{T}};\Delta^{\mathcal{T}};\cdot;\cdot;\Gamma^{\mathcal{T}}::\bullet;\mathrm{end}[\tau^{\mathcal{T}};\Gamma^{\mathcal{T}}::\bullet]\vdash \mathrm{e}:\tau^{\mathcal{T}};\Gamma^{\mathcal{T}}::\bullet$$

Interoperability: **A** and **T**

$$\frac{\Psi; \Delta; \Gamma; \cdot; \sigma; \mathbf{end}[\tau^{\langle \mathcal{T} \rangle}; \sigma'] \vdash \mathrm{e} : \tau^{\langle \mathcal{T} \rangle}; \sigma'}{\Psi; \Delta; \Gamma; \chi; \sigma; \mathbf{out} \vdash {}^{\tau}\mathcal{A}\mathcal{T}\mathrm{e} : \tau; \sigma'}$$

Interoperability: **A** and **T**

$$\frac{\Psi; \Delta; \Gamma; \cdot; \boldsymbol{\sigma}; \mathbf{end}[\boldsymbol{\tau}^{\langle \mathcal{T} \rangle}; \boldsymbol{\sigma}'] \vdash \mathbf{e} : \boldsymbol{\tau}^{\langle \mathcal{T} \rangle}; \boldsymbol{\sigma}'}{\Psi; \Delta; \Gamma; \boldsymbol{\chi}; \boldsymbol{\sigma}; \mathbf{out} \vdash {}^{\boldsymbol{\tau}}\mathcal{A}\mathcal{T}\mathbf{e} : \boldsymbol{\tau}; \boldsymbol{\sigma}'}$$

Interoperability: **A** and **T**

$$\frac{\Psi; \Delta; \Gamma; \cdot; \boldsymbol{\sigma}; \mathbf{end}[\boldsymbol{\tau}^{\langle \mathcal{T} \rangle}; \boldsymbol{\sigma}'] \vdash \mathrm{e} : \boldsymbol{\tau}^{\langle \mathcal{T} \rangle}; \boldsymbol{\sigma}'}{\Psi; \Delta; \Gamma; \chi; \boldsymbol{\sigma}; \mathbf{out} \vdash {}^{\boldsymbol{\tau}}\mathcal{AT}\,\mathrm{e} : \boldsymbol{\tau}; \boldsymbol{\sigma}'}$$

$$\frac{{}^{\boldsymbol{\tau}}\mathbf{AT}(M.\mathbf{M.R(r)}, M) = (\mathsf{v}, M')}{\langle M \mid E[{}^{\boldsymbol{\tau}}\mathcal{AT}\,\mathtt{ret}\ \mathbf{end}[\boldsymbol{\tau}^{\langle \mathcal{T} \rangle}; \boldsymbol{\sigma}], \mathbf{r}] \rangle \longmapsto \langle M' \mid E[\mathsf{v}] \rangle}$$

Interoperability: **A** and **T**

$$\frac{\Psi; \Delta; \Gamma; \cdot; \boldsymbol{\sigma}; \mathbf{end}[\boldsymbol{\tau}^{\langle \mathcal{T} \rangle}; \boldsymbol{\sigma}'] \vdash \mathrm{e} : \boldsymbol{\tau}^{\langle \mathcal{T} \rangle}; \boldsymbol{\sigma}'}{\Psi; \Delta; \Gamma; \chi; \boldsymbol{\sigma}; \mathbf{out} \vdash {}^{\boldsymbol{\tau}}\mathcal{A}\mathcal{T}\mathrm{e} : \boldsymbol{\tau}; \boldsymbol{\sigma}'}$$

$$\frac{{}^{\boldsymbol{\tau}}\mathbf{AT}(M.\mathbf{M.R(r)}, M) = (\mathsf{v}, M')}{\langle M \mid E[{}^{\boldsymbol{\tau}}\mathcal{A}\mathcal{T}\mathtt{ret}\,\mathbf{end}[\boldsymbol{\tau}^{\langle \mathcal{T} \rangle}; \boldsymbol{\sigma}], \mathbf{r}]\rangle \longmapsto \langle M' \mid E[\mathsf{v}]\rangle}$$

Interoperability: **A** and **T**

$$\iota \quad ::= \cdots \mid \texttt{import}\ \mathbf{r_d}, {}^{\sigma}\mathcal{TA}^{\tau}\,\mathbf{e}$$

$$\frac{\mathbf{TA}^{\tau}(\mathsf{v}, M) = (\mathbf{w}, M')}{\langle M \mid E[\texttt{import}\ \mathbf{r_d}, {}^{\sigma'}\mathcal{TA}^{\tau}\,\mathsf{v}; \mathbf{I}]\rangle \longmapsto \langle M' \mid E[\texttt{mv}\ \mathbf{r_d}, \mathbf{w}; \mathbf{I}]\rangle}$$

$$\frac{\begin{array}{c}\sigma = \tau_0 :: \cdots :: \tau_j :: \sigma_0 \qquad \sigma' = \tau'_0 :: \cdots :: \tau'_k :: \sigma_0 \\ \Psi; \Delta, \zeta; \Gamma; \chi; (\tau_0 :: \cdots :: \tau_j :: \zeta); \mathrm{out} \vdash \mathbf{e} : \tau; (\tau'_0 :: \cdots :: \tau'_k :: \zeta) \qquad \mathbf{q} = \mathbf{i} > \mathbf{j} \text{ or } \mathbf{q} =\end{array}}{\Psi; \Delta; \Gamma; \chi; \sigma; \mathbf{q} \vdash \texttt{import}\ \mathbf{r_d}, {}^{\sigma_0}\mathcal{TA}^{\tau}\,\mathbf{e} \Rightarrow \Delta; (\mathbf{r_d} : \tau^{\mathcal{T}}); \sigma'; \mathrm{inc}(\mathbf{q}, \mathbf{k}-\mathbf{j})}$$

Interoperability: **A** and **T**

$$\iota \quad ::= \cdots \mid \texttt{import}\ \mathbf{r_d},{}^{\sigma}\mathcal{T}\mathcal{A}^{\tau}\,\mathbf{e}$$

$$\frac{\mathbf{TA}^{\tau}(\mathsf{v}, M) = (\mathbf{w}, M')}{\langle M \mid E[\texttt{import}\ \mathbf{r_d},{}^{\sigma'}\mathcal{T}\mathcal{A}^{\tau}\,\mathsf{v};\mathbf{I}]\rangle \longmapsto \langle M' \mid E[\texttt{mv}\ \mathbf{r_d},\mathbf{w};\mathbf{I}]\rangle}$$

$$\frac{\begin{array}{c}\sigma = \tau_0 :: \cdots :: \tau_j :: \sigma_0 \qquad \sigma' = \tau'_0 :: \cdots :: \tau'_k :: \sigma_0 \\ \Psi;\Delta,\zeta;\Gamma;\chi;(\tau_0 :: \cdots :: \tau_j :: \zeta);\mathbf{out} \vdash \mathbf{e} : \tau;(\tau'_0 :: \cdots :: \tau'_k :: \zeta) \qquad \mathbf{q} = \mathbf{i} > \mathbf{j} \text{ or } \mathbf{q} =\end{array}}{\Psi;\Delta;\Gamma;\chi;\sigma;\mathbf{q} \vdash \texttt{import}\ \mathbf{r_d},{}^{\sigma_0}\mathcal{T}\mathcal{A}^{\tau}\,\mathbf{e} \Rightarrow \Delta;(\mathbf{r_d} : \tau^{\mathcal{T}});\sigma';\mathrm{inc}(\mathbf{q},\mathbf{k}-\mathbf{j})}$$

Other Issues

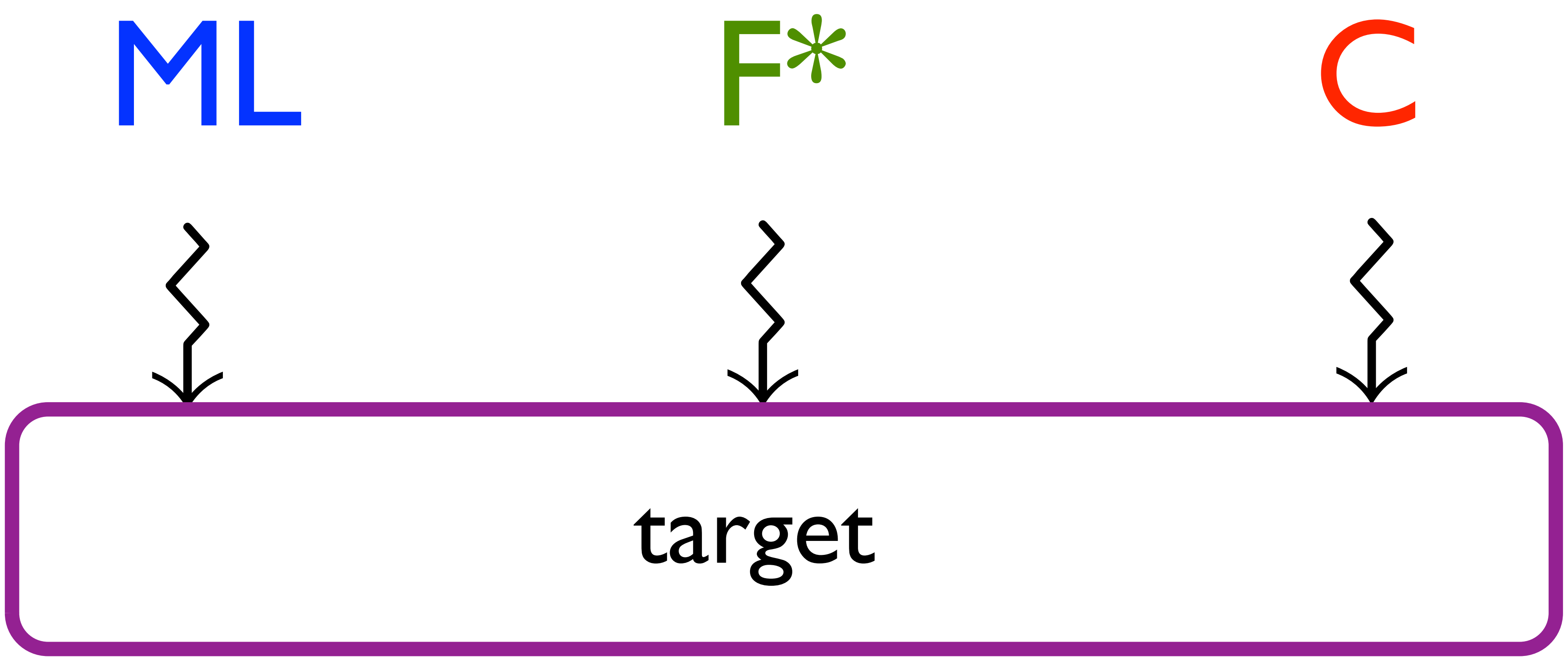
Contexts of **FCAT**

- plugging **T** context with a component is subtle

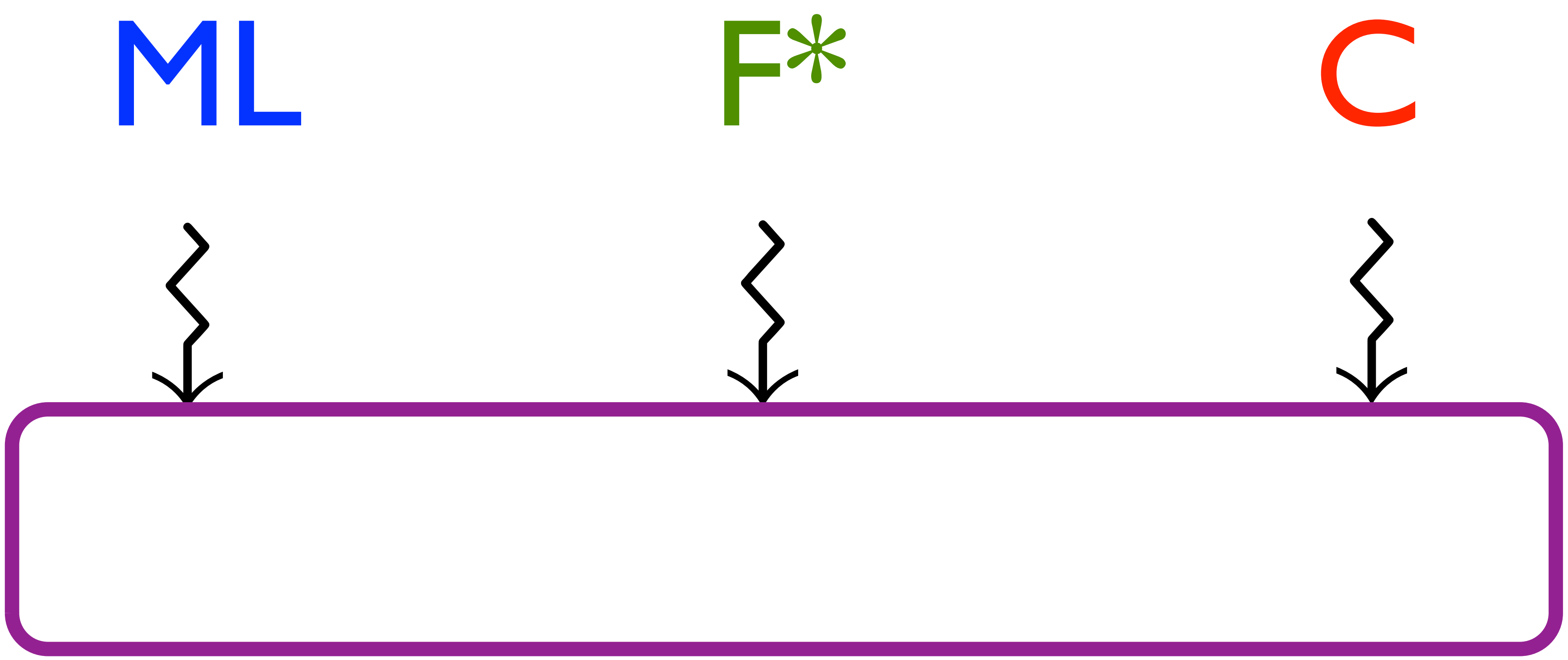
$$\begin{aligned} \mathbf{C} &::= (\mathbf{C_I}, \mathbf{H}) \mid (\mathbf{I}, \mathbf{C_H}) \\ \mathbf{C_I} &::= [\cdot] \mid \iota; \mathbf{C_I} \mid \texttt{import}\ \mathbf{r_d}, {}^{\sigma}\mathcal{T}\mathcal{A}^{\tau}\ \mathsf{C}; \mathbf{I} \\ \mathbf{C_H} &::= \mathbf{C_H}, \ell \mapsto \mathbf{h} \mid \mathbf{H}, \ell \mapsto \mathbf{code}[\Delta]\{\chi; \sigma\}^{\mathbf{q}}.\mathbf{C_I} \end{aligned}$$

Logical Relation for **FCAT** .... nontrivial!

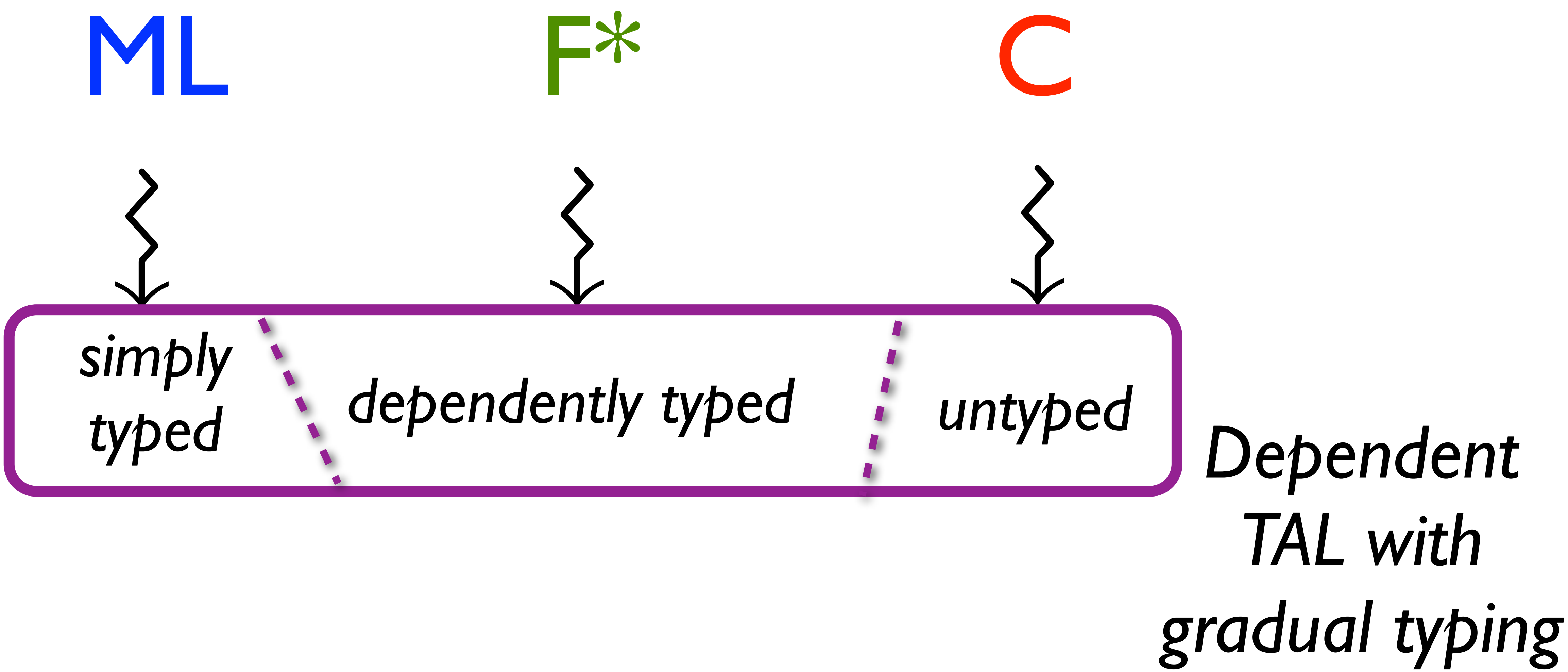
Stepping Back... where's this going?



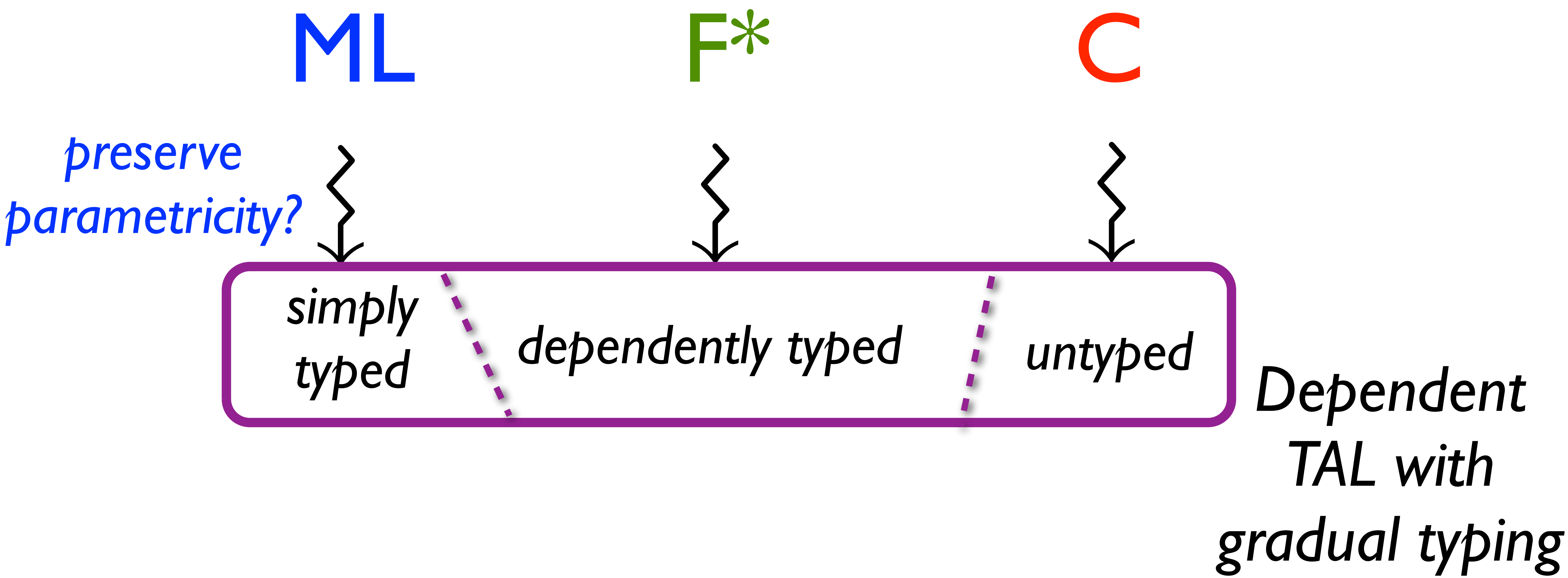
Stepping Back... where's this going?



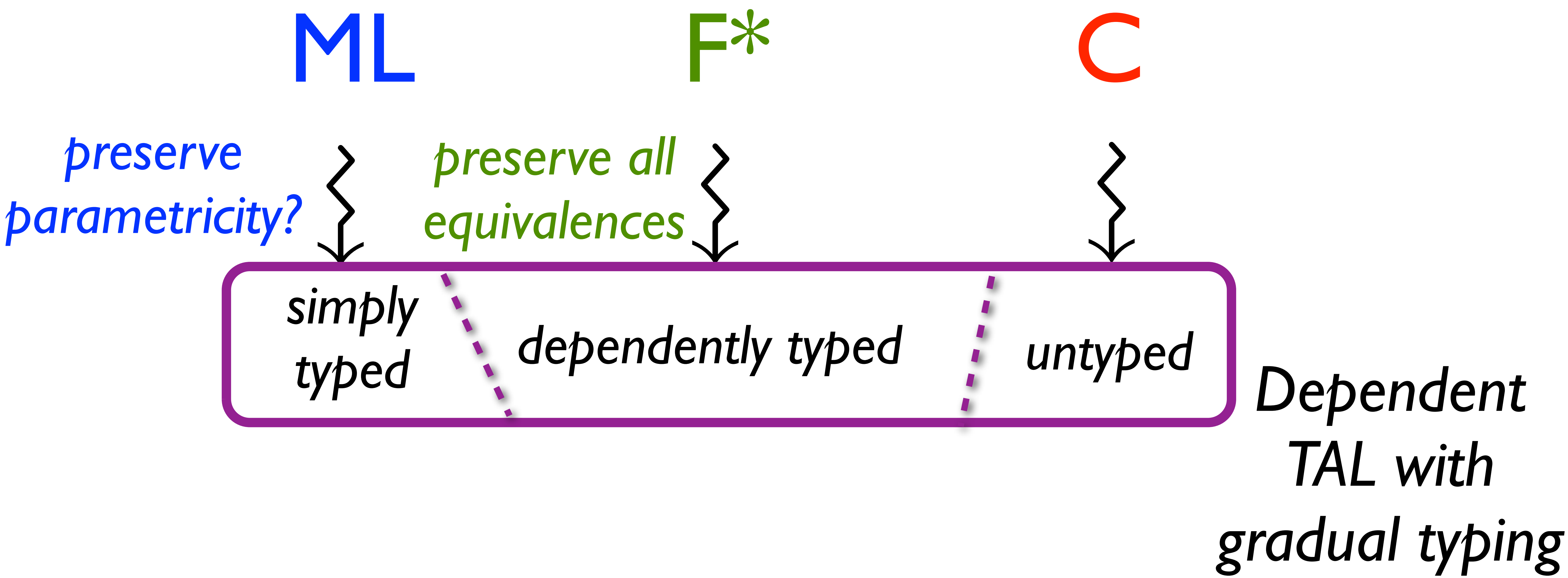
Stepping Back... where's this going?



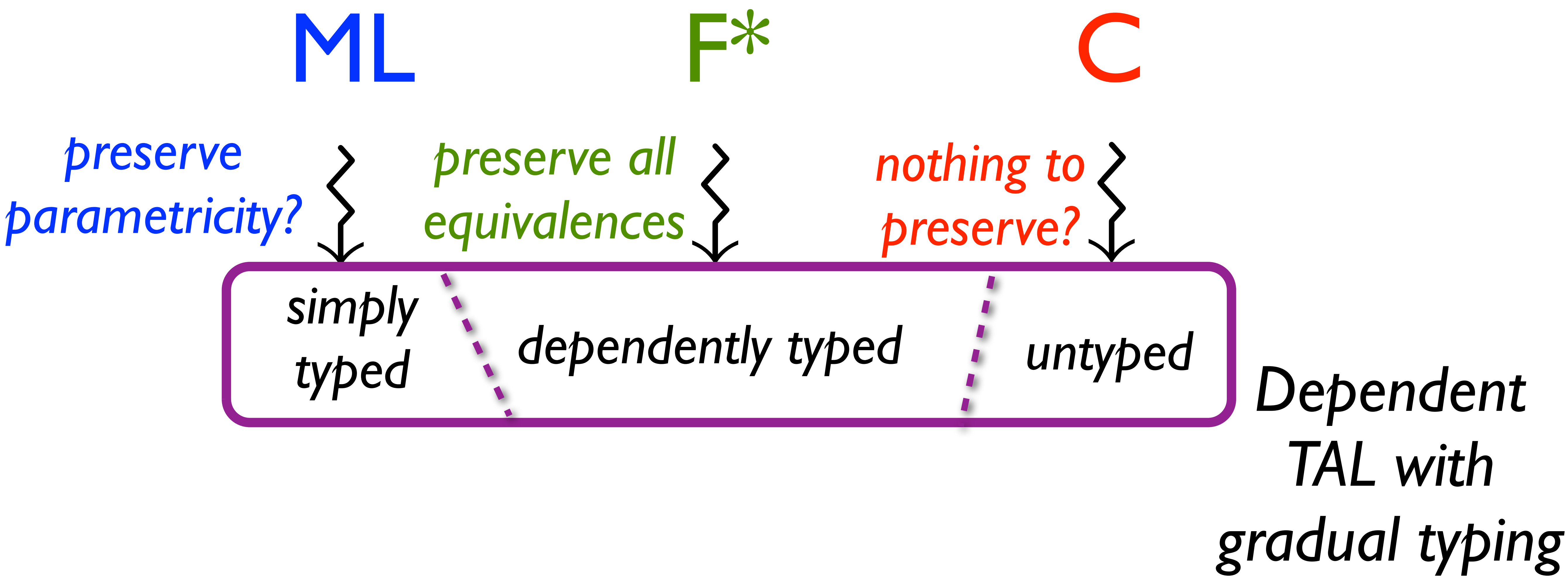
Stepping Back... where's this going?



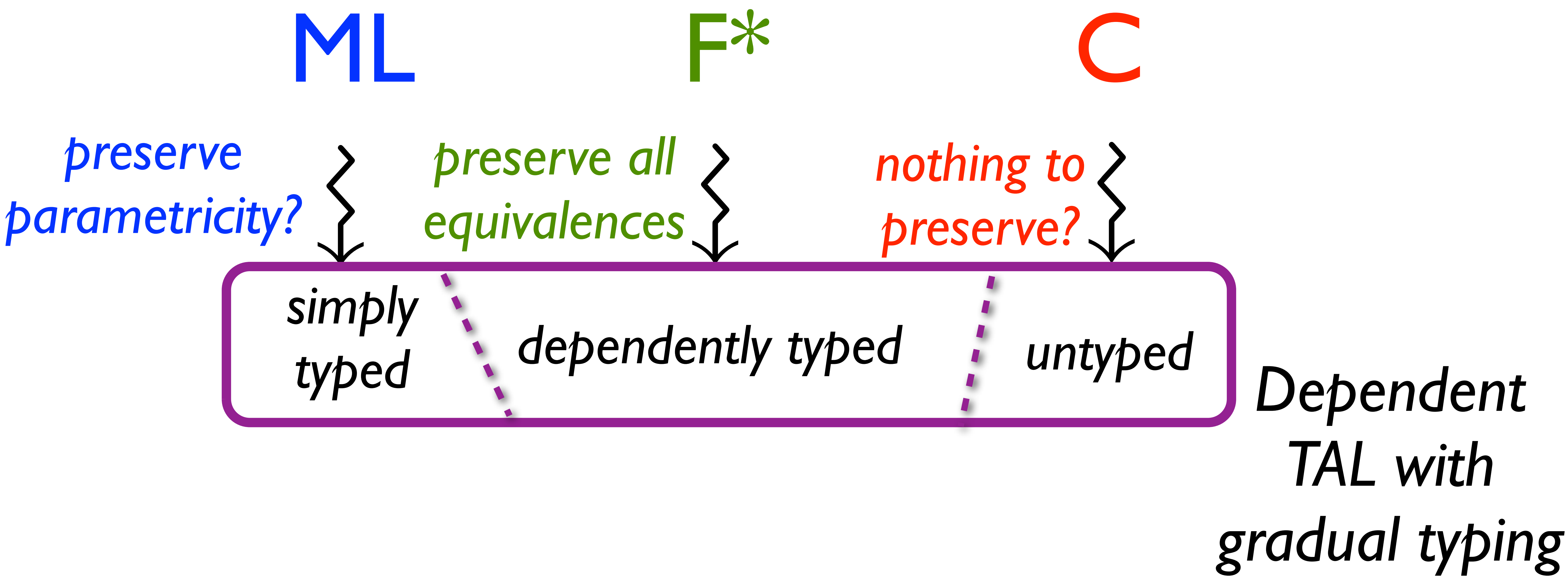
Stepping Back... where's this going?



Stepping Back... where's this going?



Stepping Back... where's this going?



It's about principled language interoperability!

Conclusions

Correct compilation of components, not just whole programs

- it's a language interoperability problem!

Multi-language approach:

- works for multi-pass compilers
- supports linking with target code of arbitrary provenance
- an opportunity to study principled interoperability
- interoperability semantics provides a specification of when source and target are related
- but have to get all the languages to fit together!

Questions?