Fully Abstract Closure Conversion (in the presence of state and effects)

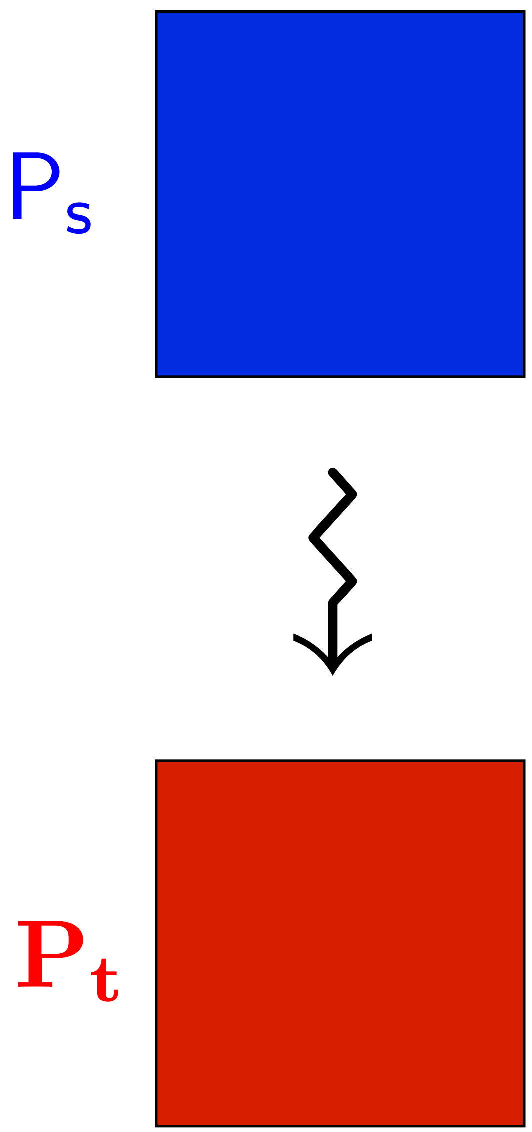
Amal Ahmed

Northeastern University

Work in progress, with Phillip Mates

Verified compilers for a multi-language world

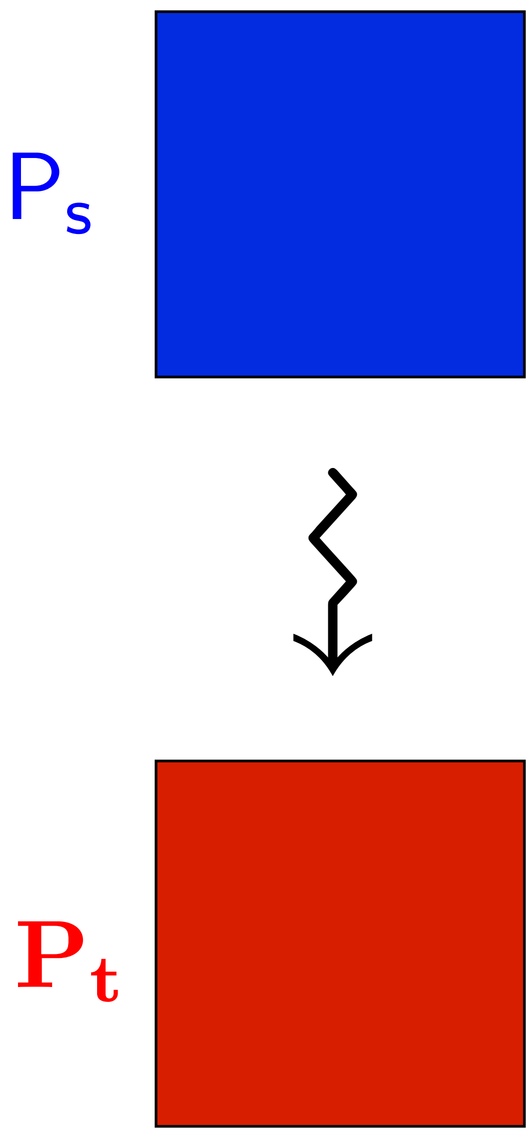
Existing work: correct compilation guarantee only applies to whole programs!



$$\Longrightarrow \quad P_s \approx \mathbf{P_t}$$

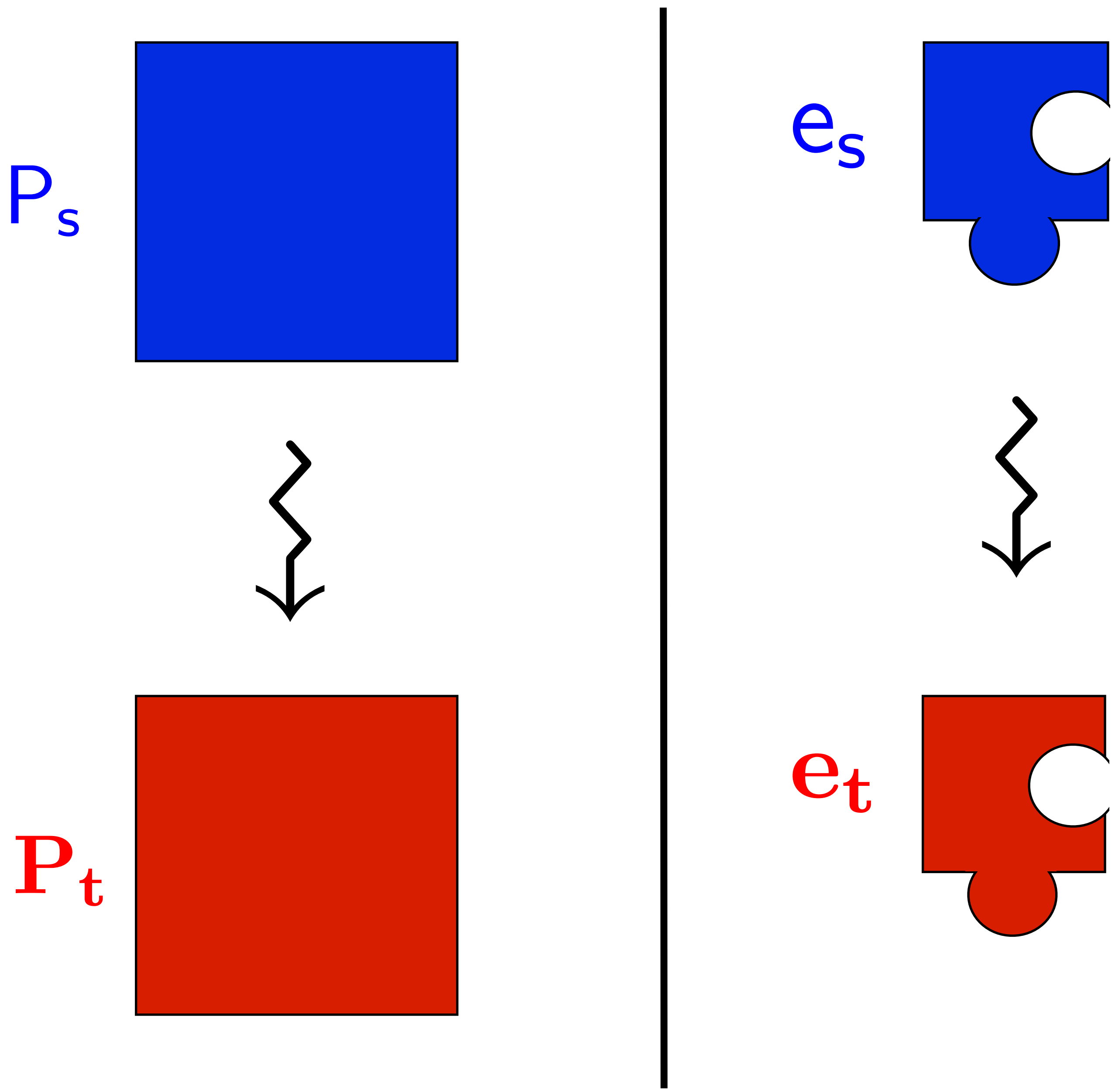
Verified compilers for a multi-language world

Existing work: correct compilation guarantee only applies to whole programs!



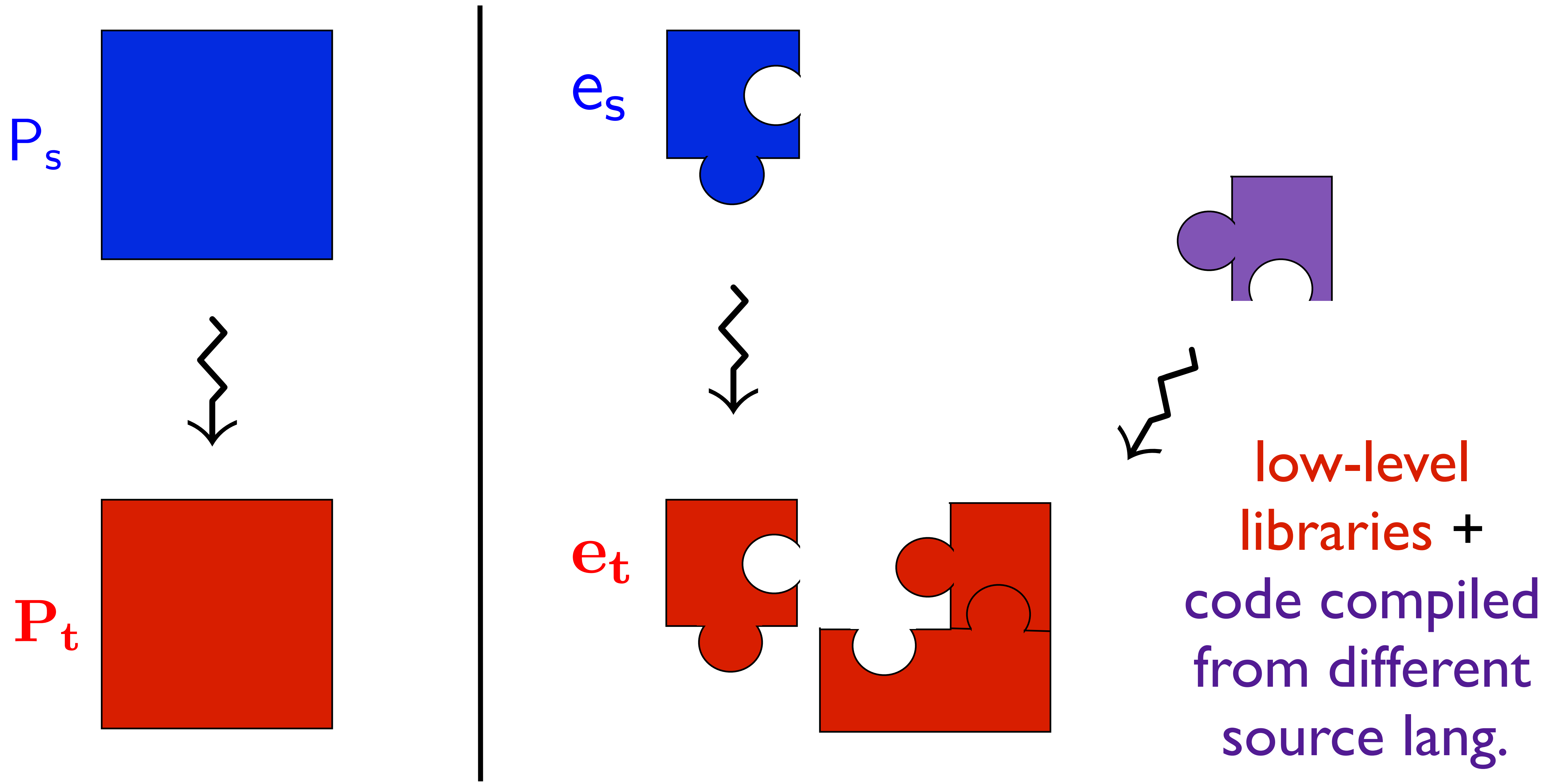
Verified compilers for a multi-language world

Existing work: correct compilation guarantee only applies to **whole** programs!



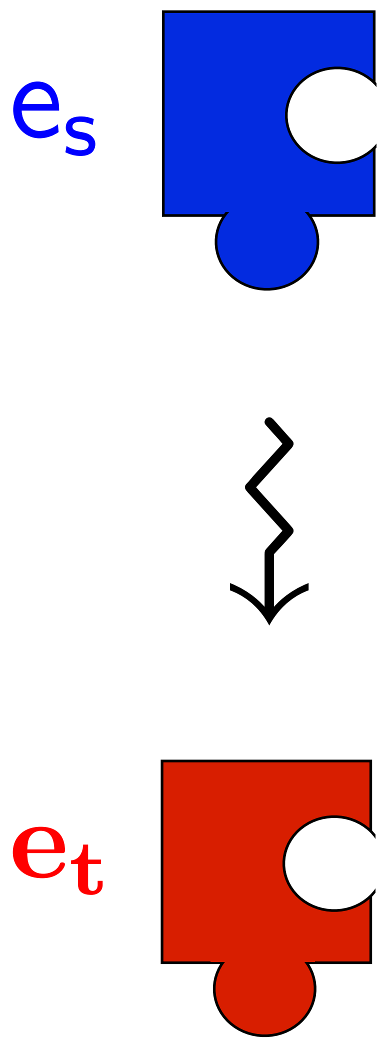
Verified compilers for a multi-language world

Existing work: correct compilation guarantee only applies to **whole** programs!



Verified compilers for a multi-language world

Correct compilation of components:

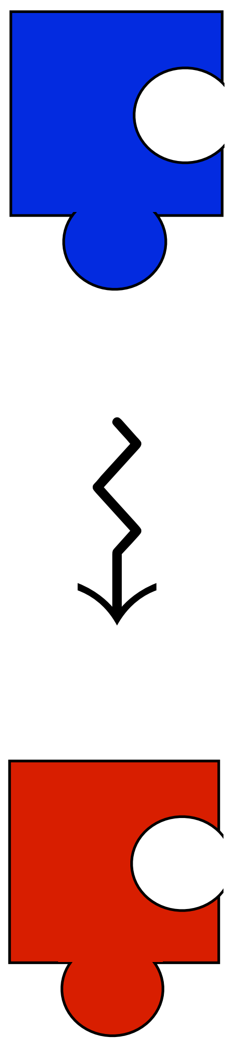


$$\Longrightarrow \quad e_s \approx e_t$$

Verified compilers for a multi-language world

Correct compilation of components:

$\mathrm{e_s}$



$\mathrm{e_t}$

$$\Longrightarrow \quad \mathrm{e_s} \approx \mathbf{e_t}$$

Define semantics of source-target interoperability:

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

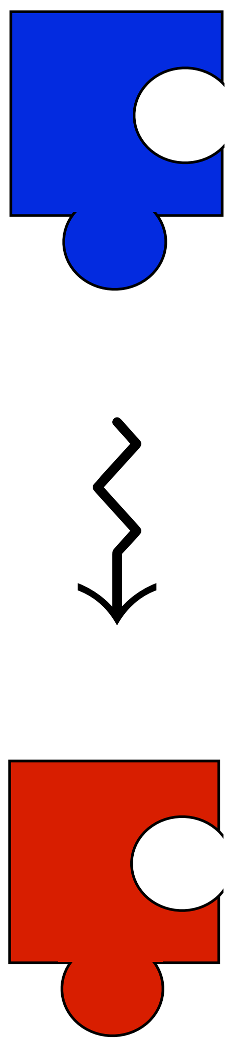
Verified compilers for a multi-language world

Correct compilation of components:

Define semantics of source-target interoperability:

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

$\mathsf{e_s}$

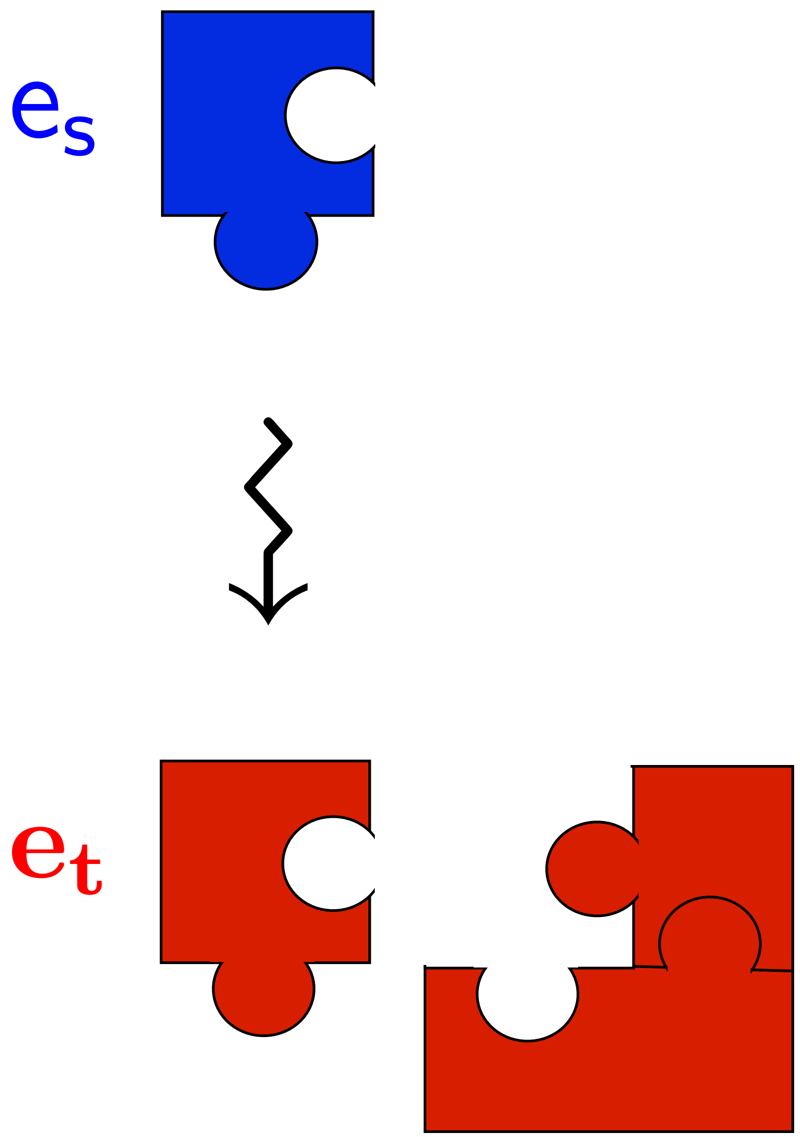


$\mathbf{e_t}$

$$\Longrightarrow \quad \mathsf{e_s} \approx \mathbf{e_t} \stackrel{\text{def}}{=} \mathsf{e_s} \approx^{ctx} \mathcal{ST}\,\mathbf{e_t}$$

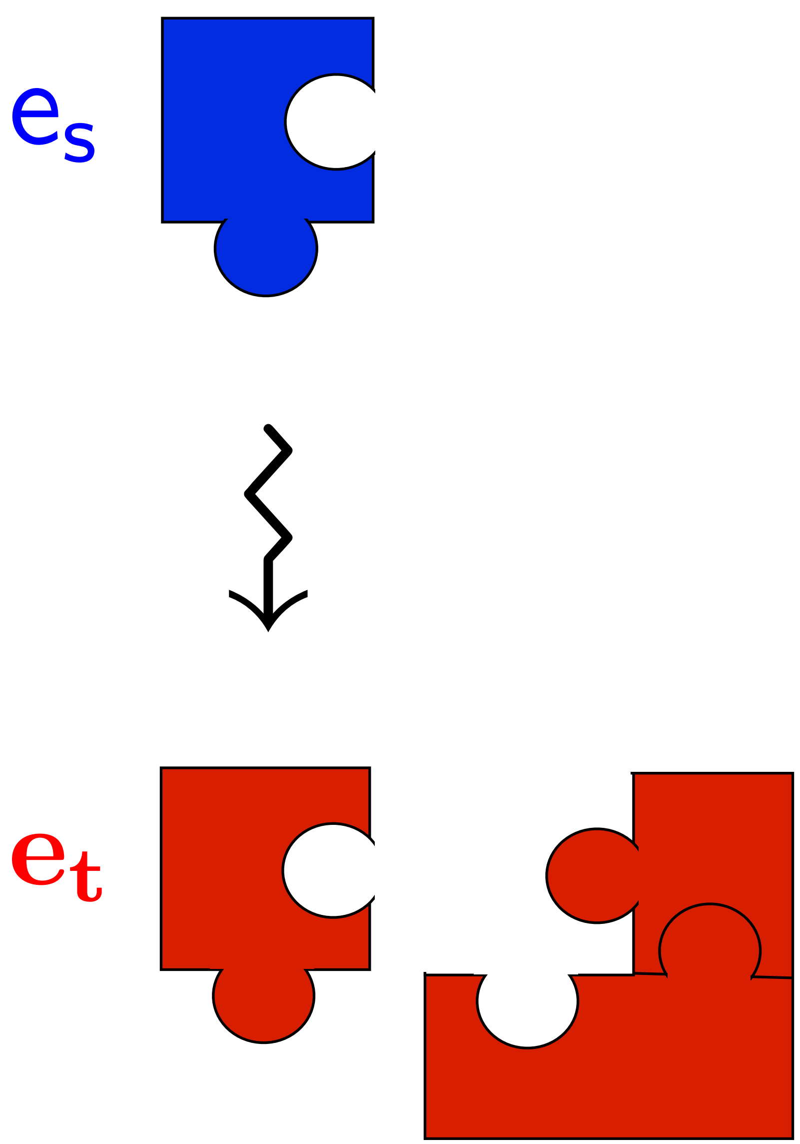
Verified compilers for a multi-language world

Secure compilation of components:



Verified compilers for a multi-language world

Secure compilation of components:

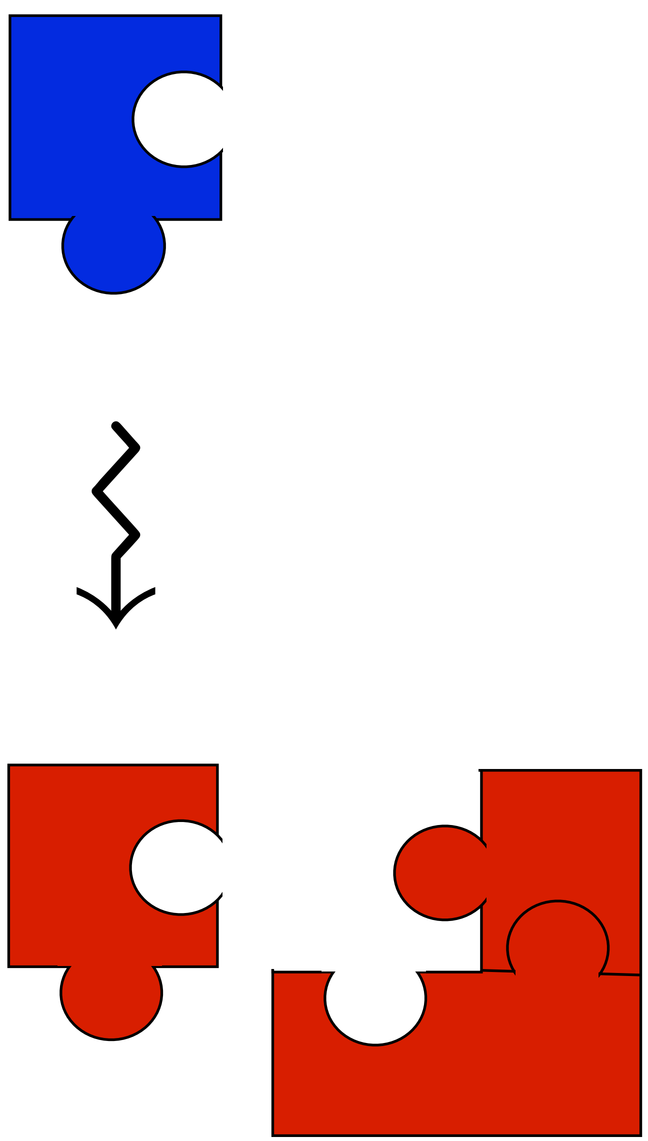


Want guarantee that e_t will remain as secure as e_s when executed in arbitrary target-level contexts

Verified compilers for a multi-language world

Secure compilation of components:

e_s



e_t

Want guarantee that e_t will remain as secure as e_s when executed in arbitrary target-level contexts

To preserve *two-run* security/reliability properties (e.g., noninterference & representation independence), compiler must preserve observational equivalence

Type-preserving compilation

$$\mathbf{e} : \tau \rightsquigarrow \mathbf{e} : \tau^{+}$$

Equivalence-preserving compilation

If $\mathsf{e}_1 : \tau \rightsquigarrow \mathbf{e}_1 : \tau^+$ and $\mathsf{e}_2 : \tau \rightsquigarrow \mathbf{e}_2 : \tau^+$ then:

$$\mathsf{e}_1 \approx_{\mathrm{S}}^{ctx} \mathsf{e}_2 : \tau \implies \mathbf{e}_1 \approx_{\mathrm{T}}^{ctx} \mathbf{e}_2 : \tau^+$$

Fully abstract compilation

If $\mathsf{e}_1 : \tau \rightsquigarrow \mathbf{e}_1 : \tau^+$ and $\mathsf{e}_2 : \tau \rightsquigarrow \mathbf{e}_2 : \tau^+$ then:

$$\mathsf{e}_1 \approx^{ctx}_{\mathrm{S}} \mathsf{e}_2 : \tau \iff \mathbf{e}_1 \approx^{ctx}_{\mathrm{T}} \mathbf{e}_2 : \tau^+$$

preserves & reflects equivalence

Security-preserving = Fully abstract

Security-preserving = Fully abstract

- If compilation is not equivalence-preserving then there exist contexts (i.e., attackers!) at target that can distinguish program fragments that cannot be distinguished by source contexts
  - C# to .NET IL compiler [Kennedy'06]: holes in full abstraction that lead to security exploits

Security-preserving = Fully abstract

- If compilation is not equivalence-preserving then there exist contexts (i.e., attackers!) at target that can distinguish program fragments that cannot be distinguished by source contexts
  - C# to .NET IL compiler [Kennedy'06]: holes in full abstraction that lead to security exploits

Our eventual goal: security-preserving compilation of dependently typed, stateful languages (HTT, F*)

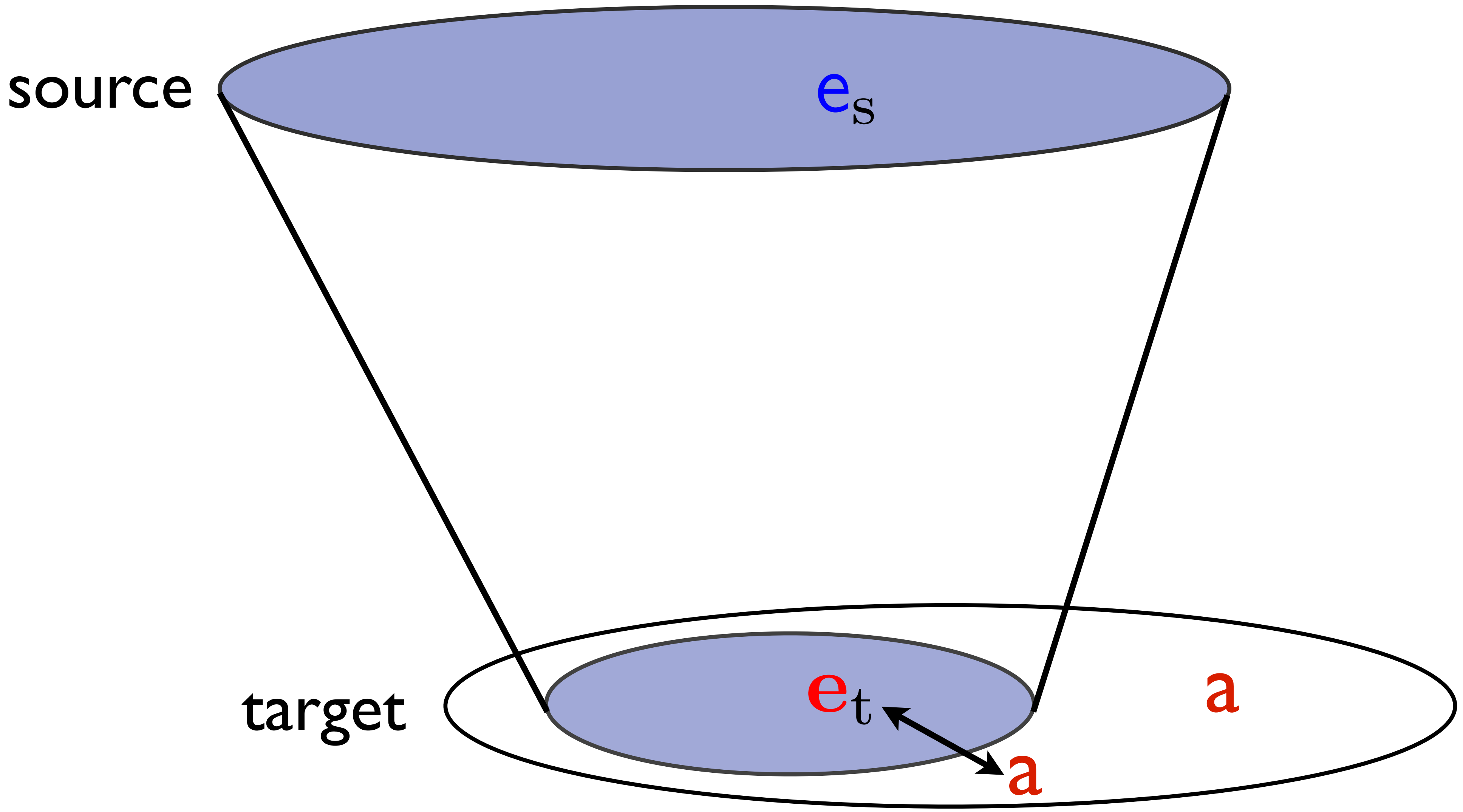
Security-preserving = Fully abstract

- If compilation is not equivalence-preserving then there exist contexts (i.e., attackers!) at target that can distinguish program fragments that cannot be distinguished by source contexts
  - C# to .NET IL compiler [Kennedy'06]: holes in full abstraction that lead to security exploits

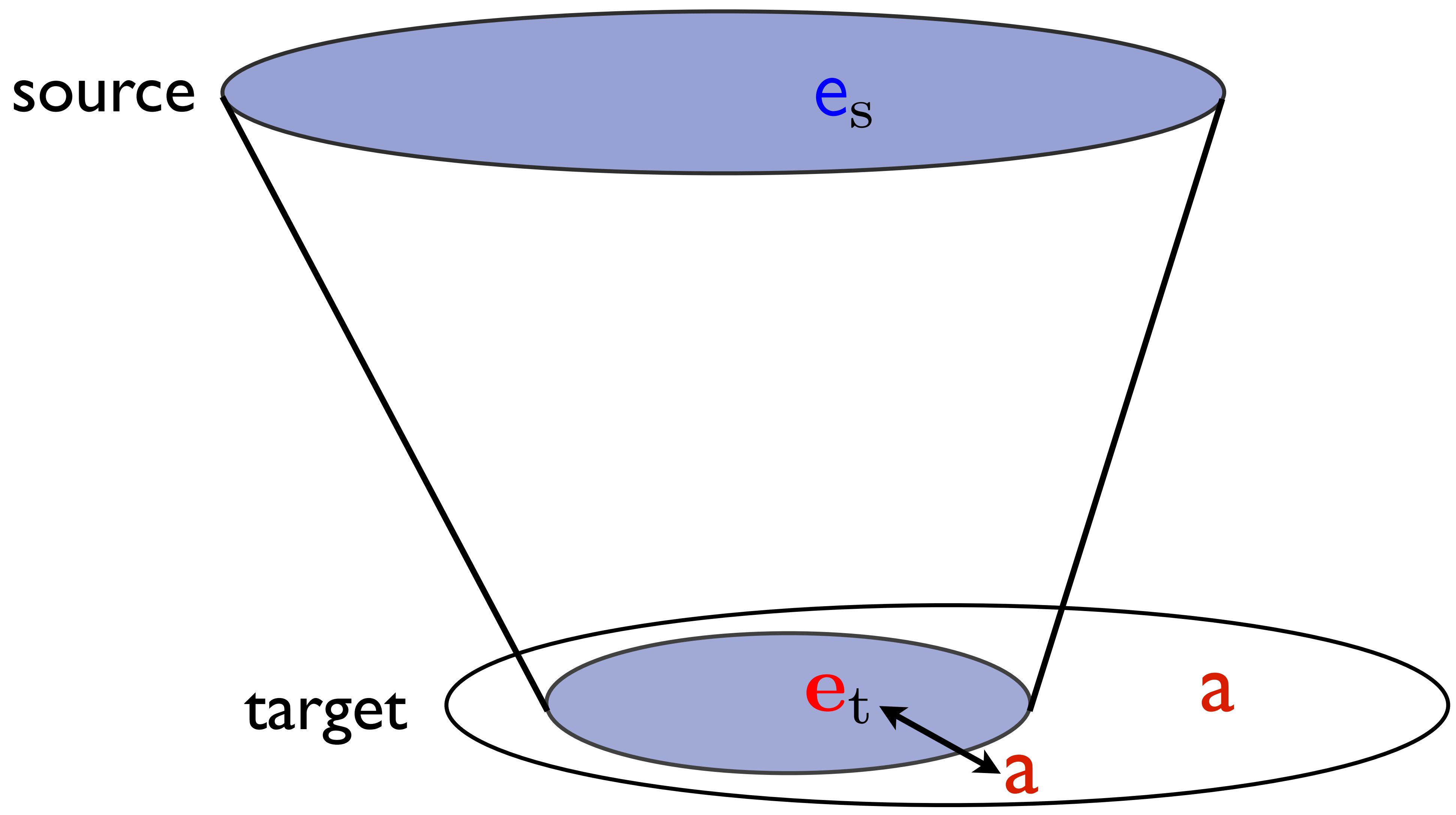
Our eventual goal: security-preserving compilation of dependently typed, stateful languages (HTT, F*)

This talk: fully abstract closure conversion of System F with mutable references

Why is full abstraction challenging?

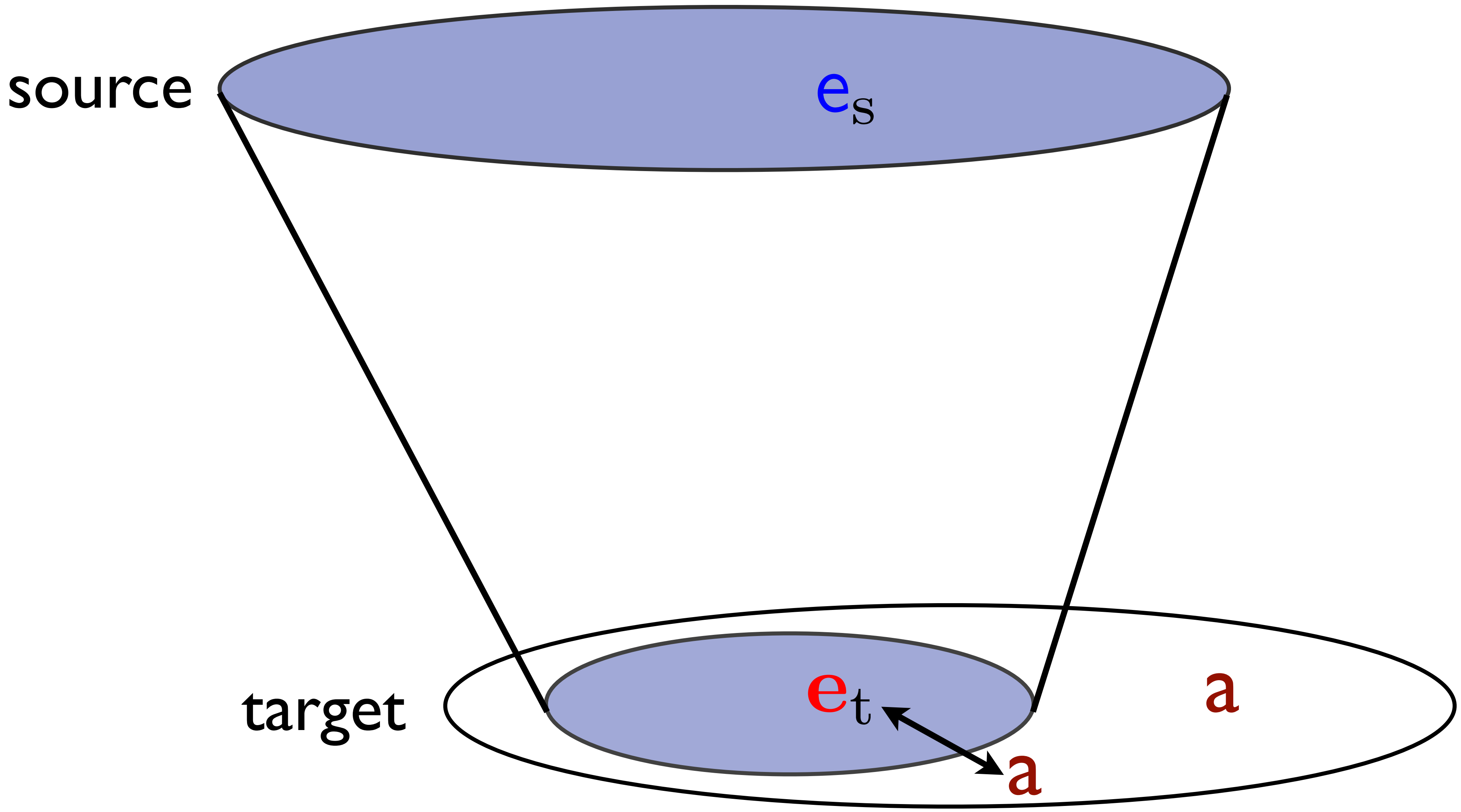


Why is full abstraction challenging?

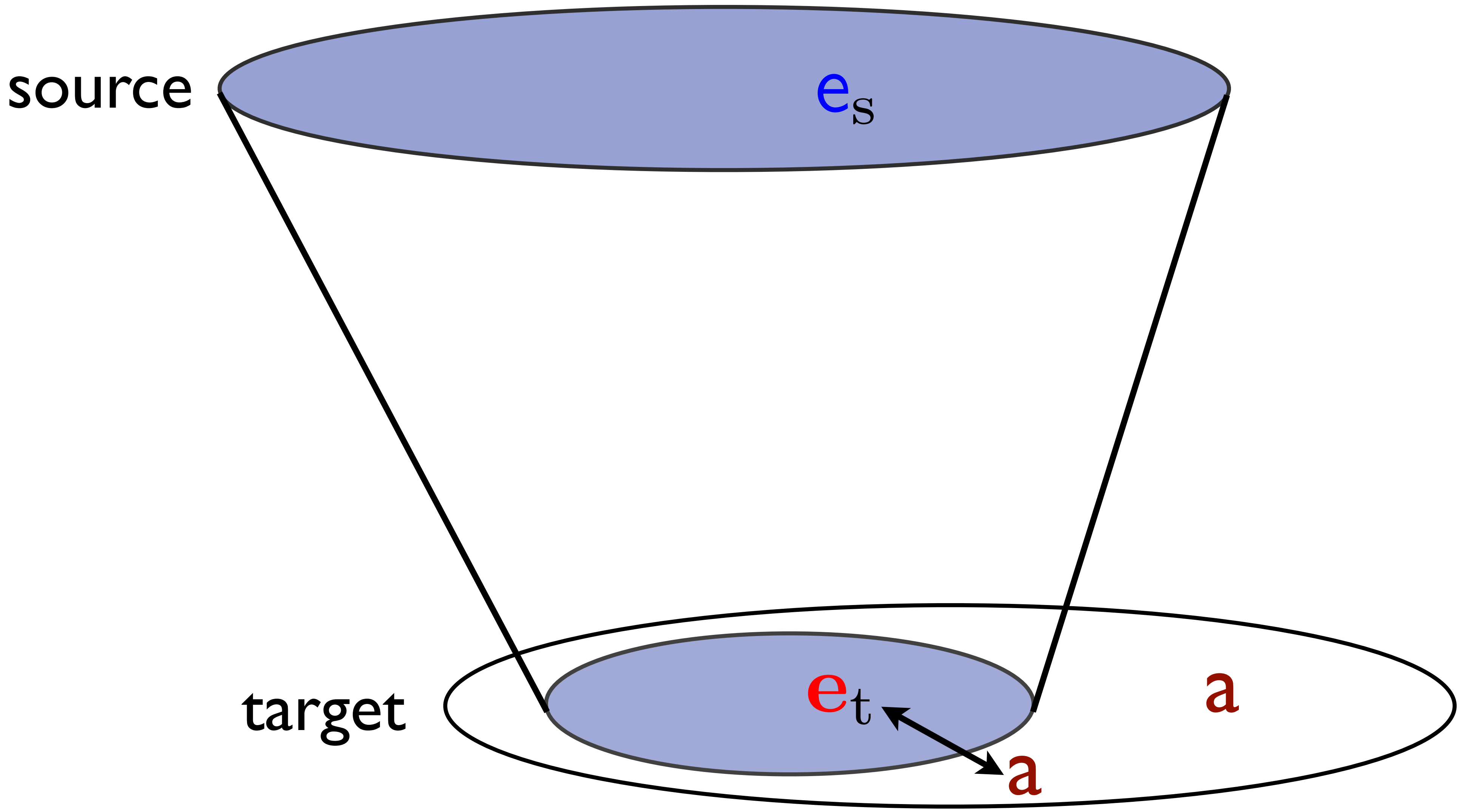


Must ensure that any a we link with behaves like some source context

Why is full abstraction challenging?

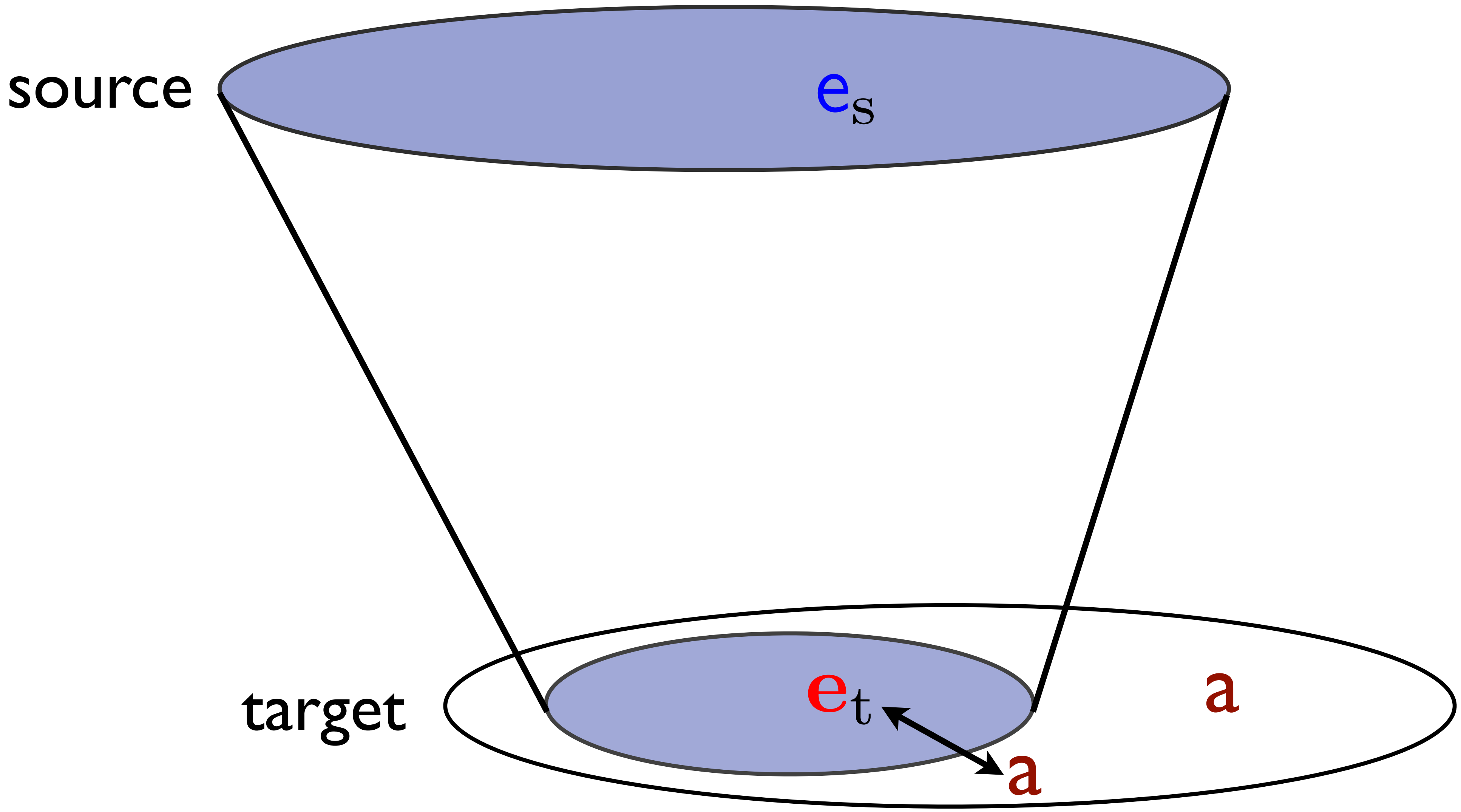


Why is full abstraction challenging?



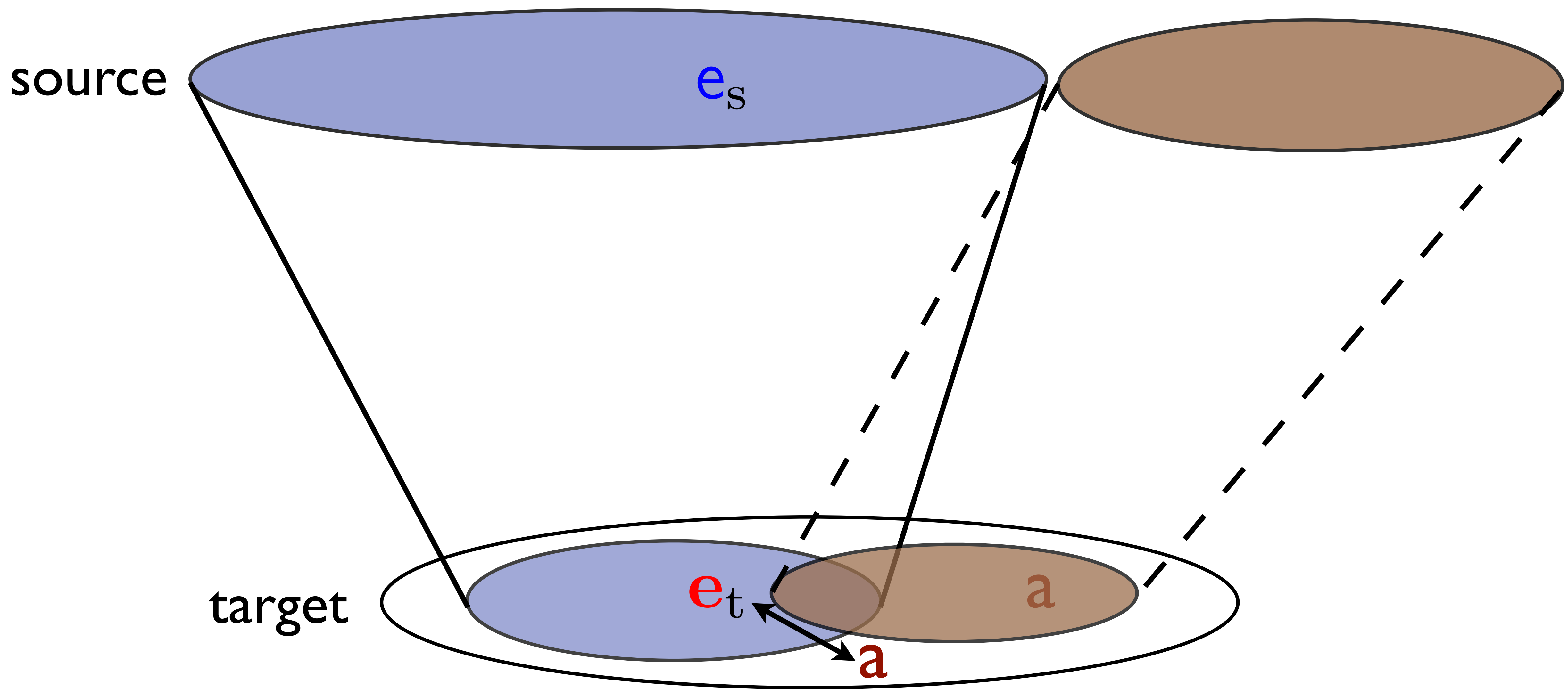
- Fix (i) Increase expressivity of source

Why is full abstraction challenging?



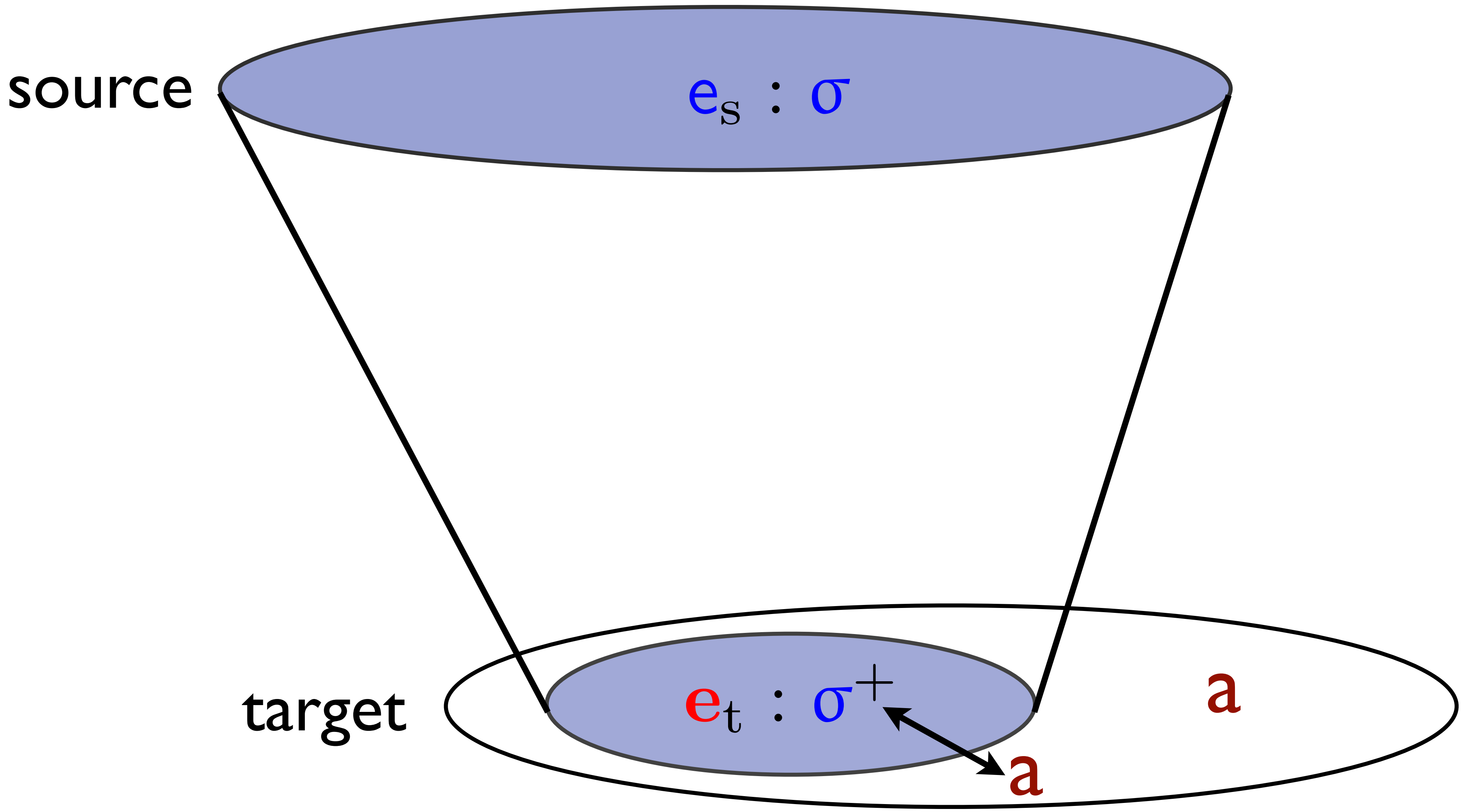
- Fix (i) Increase expressivity of source
- Fix (ii) Decrease expressivity of target

Why is full abstraction challenging?



- Fix (i) Increase expressivity of source
- Fix (ii) Decrease expressivity of target

Why is full abstraction challenging?



- Fix (i) Increase expressivity of source
- Fix (ii) Decrease expressivity of target
- Fix (iii) Change the translation: use types to rule out bad a's

Challenge of proving full abstraction

Suppose $\Gamma \vdash \mathsf{e}_1 : \tau \rightsquigarrow \mathbf{e}_1$ and $\Gamma \vdash \mathsf{e}_2 : \tau \rightsquigarrow \mathbf{e}_2$.

$$\Gamma \vdash \mathsf{e}_1 \approx^{ctx}_{\mathrm{S}} \mathsf{e}_2 : \tau$$

$$\Big\downarrow$$

$$\Gamma^{+} \vdash \mathbf{e}_1 \approx^{ctx}_{\mathrm{T}} \mathbf{e}_2 : \tau^{+}$$

Challenge of proving full abstraction

Suppose $\Gamma \vdash \mathsf{e}_1 : \tau \rightsquigarrow \mathbf{e}_1$ and $\Gamma \vdash \mathsf{e}_2 : \tau \rightsquigarrow \mathbf{e}_2$.

$$\Gamma \vdash \mathsf{e}_1 \approx^{ctx}_{\mathrm{S}} \mathsf{e}_2 : \tau$$

$$\downarrow$$

$$\Gamma^+ \vdash \mathbf{e}_1 \approx^{ctx}_{\mathrm{T}} \mathbf{e}_2 : \tau^+$$

Given:

No C_{S} can distinguish e_1, e_2

Show:

Given arbitrary C_{T}, it cannot distinguish $\mathbf{e}_1$, $\mathbf{e}_2$

Need to be able to "back-translate" C_{T} to an equivalent C_{S}

Challenge of proving full abstraction

Suppose $\Gamma \vdash \mathsf{e}_1 : \tau \rightsquigarrow \mathbf{e}_1$ and $\Gamma \vdash \mathsf{e}_2 : \tau \rightsquigarrow \mathbf{e}_2$.

$$\Gamma \vdash \mathsf{e}_1 \approx^{ctx}_{\mathrm{S}} \mathsf{e}_2 : \tau$$

$$\Downarrow$$

$$\Gamma^+ \vdash \mathbf{e}_1 \approx^{ctx}_{\mathrm{T}} \mathbf{e}_2 : \tau^+$$

"Back-translation"

What if target language is more expressive than source?

Equivalence-preserving CPS from STLC to System F

[Ahmed-Blume ICFP'11]

Quick note: “same language trick”

If target happens to be no more expressive than source, use the same language: back-translation can be avoided in lieu of *wrappers* between τ and τ^+

- Closure conversion: System F with recursive types
  [Ahmed-Blume ICFP’08]
- f* (STLC with refs) to js* (encoding of JavaScript in f*)
  [Fournet et al. POPL’13]

Closure Conversion

Source

$$\tau ::= \alpha \mid \mathsf{unit} \mid \mathsf{int} \mid \forall[\overline{\alpha}].(\overline{\tau}) \rightarrow \tau \mid \mu\alpha.\tau \mid \mathsf{ref}\,\tau \mid \langle\overline{\tau}\rangle$$

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

$$\mathsf{v} ::= \mathsf{x} \mid () \mid \mathsf{n} \mid \lambda[\overline{\alpha}](\overline{\mathsf{x}:\tau}).\mathsf{e} \mid \mathsf{fold}_{\mu\alpha.\tau}\,\mathsf{v} \mid \ell \mid \langle\overline{\mathsf{v}}\rangle$$

$$\mathsf{e} ::= \mathsf{v} \mid \mathsf{v}\,[\overline{\tau}]\,\overline{\mathsf{v}} \mid \mathsf{v}\ \mathsf{p}\ \mathsf{v} \mid \mathsf{new}\,\mathsf{v} \mid \mathsf{v} := \mathsf{v} \mid !\mathsf{v} \mid \mathsf{unfold}\,\mathsf{v} \mid \pi_{\mathsf{i}}(\mathsf{v}) \mid \mathsf{let}\,\mathsf{x} = \mathsf{e}\ \mathsf{in}\ \mathsf{e} \mid \mathsf{if0}\,\mathsf{v}\ \mathsf{e}\ \mathsf{e}$$

$$\mathsf{E} ::= [\cdot] \mid \mathsf{let}\,\mathsf{x} = \mathsf{E}\ \mathsf{in}\ \mathsf{e}$$

Target

Closure Conversion

Source

$\tau \ ::= \ \alpha \mid \mathsf{unit} \mid \mathsf{int} \mid \forall[\overline{\alpha}].(\overline{\tau}) \rightarrow \tau \mid \mu\alpha.\tau \mid \mathsf{ref}\ \tau \mid \langle\overline{\tau}\rangle$

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

$\mathsf{v} \ ::= \ \mathsf{x} \mid () \mid \mathsf{n} \mid \lambda[\overline{\alpha}](\overline{\mathsf{x}:\tau}).\mathsf{e} \mid \mathsf{fold}_{\mu\alpha.\tau}\ \mathsf{v} \mid \ell \mid \langle\overline{\mathsf{v}}\rangle$

$\mathsf{e} \ ::= \ \mathsf{v} \mid \mathsf{v}\ [\overline{\tau}]\ \overline{\mathsf{v}} \mid \mathsf{v}\ \mathsf{p}\ \mathsf{v} \mid \mathsf{new}\ \mathsf{v} \mid \mathsf{v} := \mathsf{v} \mid !\mathsf{v} \mid \mathsf{unfold}\ \mathsf{v} \mid \pi_{\mathsf{i}}(\mathsf{v}) \mid \mathsf{let}\ \mathsf{x} = \mathsf{e}\ \mathsf{in}\ \mathsf{e} \mid \mathsf{if0}\ \mathsf{v}\ \mathsf{e}\ \mathsf{e}$

$\mathsf{E} \ ::= \ [\cdot] \mid \mathsf{let}\ \mathsf{x} = \mathsf{E}\ \mathsf{in}\ \mathsf{e}$

Target

$\boldsymbol{\tau} \ ::= \ \boldsymbol{\alpha} \mid \mathbf{unit} \mid \mathbf{int} \mid \forall[\overline{\boldsymbol{\alpha}}].(\overline{\boldsymbol{\tau}}) \rightarrow \boldsymbol{\tau} \mid \exists\boldsymbol{\alpha}.\boldsymbol{\tau} \mid \boldsymbol{\mu\alpha.\tau} \mid \mathbf{ref}\ \boldsymbol{\tau} \mid \langle\overline{\boldsymbol{\tau}}\rangle \mid \mathbf{cont}\ \boldsymbol{\tau}$

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

$\mathbf{v} \ ::= \ \mathbf{x} \mid () \mid \mathbf{n} \mid \boldsymbol{\lambda}[\overline{\boldsymbol{\alpha}}](\overline{\mathbf{x}:\boldsymbol{\tau}}).\mathbf{e} \mid \mathbf{pack}\ \langle\boldsymbol{\tau}, \mathbf{v}\rangle\ \mathbf{as}\ \exists\boldsymbol{\alpha}.\boldsymbol{\tau} \mid \mathbf{fold}_{\boldsymbol{\mu\alpha.\tau}}\ \mathbf{v} \mid \boldsymbol{\ell} \mid \langle\overline{\mathbf{v}}\rangle \mid \mathbf{cont}_{\boldsymbol{\tau}}\ \mathbf{E}$

$\mathbf{e} \ ::= \ \mathbf{v} \mid \mathbf{unpack}\ \langle\boldsymbol{\alpha}, \mathbf{x}\rangle = \mathbf{v}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{v}\ []\ \overline{\mathbf{v}} \mid \mathbf{v}\ [\boldsymbol{\tau}] \mid \mathbf{v}\ \mathbf{p}\ \mathbf{v} \mid \mathbf{new}\ \mathbf{v} \mid \mathbf{v} := \mathbf{v} \mid !\mathbf{v} \mid \mathbf{unfold}\ \mathbf{v} \mid \boldsymbol{\pi}_{\mathbf{i}}(\mathbf{v})$
$\quad \mid \mathbf{let}\ \mathbf{x} = \mathbf{e}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{if0}\ \mathbf{v}\ \mathbf{e}\ \mathbf{e} \mid \mathbf{call/cc}_{\boldsymbol{\tau}}(\mathbf{x}.\mathbf{e}) \mid \mathbf{throw}_{\boldsymbol{\tau}}\ \mathbf{v}\ \mathbf{to}\ \mathbf{e}$

$\mathbf{E} \ ::= \ [\cdot] \mid \mathbf{let}\ \mathbf{x} = \mathbf{E}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{throw}_{\boldsymbol{\tau}}\ \mathbf{v}\ \mathbf{to}\ \mathbf{E}$

Static & Dynamic Semantics

Source

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

$$\boxed{\langle H \mid e \rangle \longmapsto \langle H' \mid e' \rangle}$$

Target

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

$$\frac{\bullet\,; \overline{\boldsymbol{\alpha}}; \overline{\mathbf{x} : \boldsymbol{\tau}} \vdash \mathbf{e} : \boldsymbol{\tau}'}{\boldsymbol{\Psi}; \boldsymbol{\Delta}; \boldsymbol{\Gamma} \vdash \boldsymbol{\lambda}[\overline{\boldsymbol{\alpha}}](\overline{\mathbf{x} : \boldsymbol{\tau}}).\mathbf{e} : \forall[\overline{\boldsymbol{\alpha}}].(\overline{\boldsymbol{\tau}}) \to \boldsymbol{\tau}'}$$

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

$$\begin{array}{ll} \langle \mathbf{H} \mid \mathbf{E}[\mathbf{call/cc}_{\boldsymbol{\tau}}(\mathbf{x.e})] \rangle & \longmapsto \langle \mathbf{H} \mid \mathbf{E}[\mathbf{e}[\mathbf{cont}_{\boldsymbol{\tau}}\ \mathbf{E}/\mathbf{x}]] \rangle \\ \langle \mathbf{H} \mid \mathbf{E}[\mathbf{throw}_{\boldsymbol{\tau}}\ \mathbf{v}\ \mathbf{to}\ \mathbf{cont}_{\boldsymbol{\tau}'}\ \mathbf{E}'] \rangle & \longmapsto \langle \mathbf{H} \mid \mathbf{E}'[\mathbf{v}] \rangle \end{array}$$

Translation

Type translation

$$\alpha^+ = \boldsymbol{\alpha}$$

$$(\forall[\overline{\alpha}].(\overline{\tau}) \rightarrow \tau')^+ = \exists\boldsymbol{\beta}.\langle(\forall[\overline{\boldsymbol{\alpha}}].(\boldsymbol{\beta}, \overline{\tau^+}) \rightarrow \tau'^+), \boldsymbol{\beta}\rangle$$

$$\mathsf{unit}^+ = \mathbf{unit}$$

$$(\exists\alpha.\tau)^+ = \exists\boldsymbol{\alpha}.\tau^+$$

$$\mathsf{int}^+ = \mathbf{int}$$

$$(\mu\alpha.\tau)^+ = \boldsymbol{\mu\alpha}.\tau^+$$

$$(\mathsf{ref}\ \tau)^+ = \mathbf{ref}\ \tau^+$$

$$\langle\tau_1, \ldots, \tau_n\rangle^+ = \langle\tau_1{}^+, \ldots, \tau_n{}^+\rangle$$

Term translation

$$\boxed{\cdot; \Delta; \Gamma \vdash \mathsf{e} : \tau \rightsquigarrow \mathbf{e}} \quad \text{where } \cdot; \Delta^+; \Gamma^+ \vdash \mathbf{e} : \tau^+$$

Is our translation fully abstract?

$$\tau = (\mathsf{unit} \rightarrow \mathsf{unit}) \rightarrow \mathsf{int})$$

$$\mathsf{e}_1 = \mathsf{let}\ \mathsf{x} = \mathsf{new}\ 0\ \mathsf{in}$$

$$\qquad \lambda \mathsf{f}.(\mathsf{x} := 0;\ \mathsf{f}\ ();\ \mathsf{x} := 1;\ \mathsf{f}\ ();\ !\mathsf{x})$$

$$\mathsf{e}_2 = \lambda \mathsf{f}.(\mathsf{f}\ ();\ \mathsf{f}\ ();\ 1)$$

Is our translation fully abstract?

$$\tau = (\mathsf{unit} \to \mathsf{unit}) \to \mathsf{int})$$

$$\begin{aligned} \mathsf{e}_1 &= \mathsf{let}\ \mathsf{x} = \mathsf{new}\ 0\ \mathsf{in} \\ &\quad \lambda \mathsf{f}.(\mathsf{x} := 0;\ \mathsf{f}\ ();\ \mathsf{x} := 1;\ \mathsf{f}\ ();\ !\mathsf{x}) \\ \mathsf{e}_2 &= \lambda \mathsf{f}.(\mathsf{f}\ ();\ \mathsf{f}\ ();\ 1) \end{aligned}$$

$$\begin{aligned} \mathsf{C} = \ & \mathsf{let}\ \mathsf{g} = [\cdot]\ \mathsf{in}\ \mathsf{let}\ \mathsf{b} = \mathsf{new}\ \mathsf{ff}\ \mathsf{in} \\ & \mathsf{letf} = (\lambda\_.\ \mathsf{if}\ !\mathsf{b}\ \mathsf{then}\ \mathsf{call/cc}(\mathsf{k}.\ \mathsf{g}\ (\lambda\_.\ \mathsf{throw}\ ()\ \mathsf{to}\ \mathsf{k})) \\ & \qquad\qquad\qquad\qquad \mathsf{else}\ \mathsf{b} := \mathsf{tt})\ \mathsf{in} \\ & \mathsf{g}\ \mathsf{f} \end{aligned}$$

Is our translation fully abstract?

$$\tau = (\mathsf{unit} \to \mathsf{unit}) \to \mathsf{int})$$

$$\begin{aligned} \mathsf{e}_1 &= \mathsf{let}\ \mathsf{x} = \mathsf{new}\ 0\ \mathsf{in} \\ &\qquad \lambda \mathsf{f}.(\mathsf{x} := 0;\ \mathsf{f}\ ();\ \mathsf{x} := 1;\ \mathsf{f}\ ();\ !\mathsf{x}) \\ \mathsf{e}_2 &= \lambda \mathsf{f}.(\mathsf{f}\ ();\ \mathsf{f}\ ();\ 1) \end{aligned}$$

$$\begin{aligned} \mathsf{C} = \ & \mathsf{let}\ \mathsf{g} = [\cdot]\ \mathsf{in}\ \mathsf{let}\ \mathsf{b} = \mathsf{new}\ \mathsf{ff}\ \mathsf{in} \\ & \mathsf{letf} = (\lambda\_.\ \mathsf{if}\ !\mathsf{b}\ \mathsf{then}\ \mathsf{call/cc}(\mathsf{k}.\ \mathsf{g}\ (\lambda\_.\ \mathsf{throw}\ ()\ \mathsf{to}\ \mathsf{k})) \\ & \qquad\qquad\qquad\qquad\quad \mathsf{else}\ \mathsf{b} := \mathsf{tt})\ \mathsf{in} \\ & \mathsf{g}\ \mathsf{f} \end{aligned}$$

$\mathsf{C}[\mathsf{e}_1]$ returns 0
$\mathsf{C}[\mathsf{e}_2]$ returns 1

Proof of Equivalence Preservation

Suppose $\cdot; \Delta; \Gamma \vdash \mathsf{e}_1 : \tau \rightsquigarrow \mathbf{e_1}$ and $\cdot; \Delta; \Gamma \vdash \mathsf{e}_2 : \tau \rightsquigarrow \mathbf{e_2}$

$$\cdot; \Delta; \Gamma \vdash \mathsf{e}_1 \approx^{ctx}_{\mathrm{S}} \mathsf{e}_2 : \tau$$

$$\Downarrow$$

$$\cdot; \Delta^+; \Gamma^+ \vdash \mathbf{e_1} \approx^{ctx}_{\mathrm{T}} \mathbf{e_2} : \tau^+$$

Given arbitrary $\mathbf{C_T} : (\cdot; \Delta^+; \Gamma^+ \vdash \tau^+) \Rightarrow (\cdot; \cdot; \cdot \vdash \mathbf{int})$
show it cannot distinguish $\mathbf{e_1}, \mathbf{e_2}$

Suffices to be able to "back-translate" $\mathbf{e}$ of translation type to an equivalent e

“Back-translation” from T to S

$$\boxed{\cdot; \Delta; \Gamma \qquad \vdash \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}}$$

where $\Delta ::= \cdot \mid \Delta, \boldsymbol{\alpha}$
and $\Gamma ::= \cdot \mid \Gamma, \mathbf{x} : \tau^{+}$
and $\mathsf{e} \in \mathrm{S}$
and $\cdot; \Delta^{\twoheadrightarrow}; \Gamma^{\twoheadrightarrow} \vdash \mathsf{e} : \tau$

"Back-translation" from T to S

$$\cdot; \Delta; \Gamma \vdash \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}$$

where $\Delta ::= \cdot \mid \Delta, \boldsymbol{\alpha}$
and $\Gamma ::= \cdot \mid \Gamma, \mathbf{x} : \tau^{+}$
and $\mathsf{e} \in S$
and $\cdot; \Delta^{\twoheadrightarrow}; \Gamma^{\twoheadrightarrow} \vdash \mathsf{e} : \tau$

$$\frac{}{\cdot; \Delta; \Gamma \vdash \mathbf{()} : \mathsf{unit}^{+} \twoheadrightarrow \mathsf{()}} \qquad \frac{}{\cdot; \Delta; \Gamma \vdash \mathbf{n} : \mathsf{int}^{+} \twoheadrightarrow \mathsf{n}}$$

$$\frac{\mathbf{x} : \tau^{+} \in \Gamma}{\cdot; \Delta; \Gamma \vdash \mathbf{x} : \tau^{+} \twoheadrightarrow \mathsf{x}}$$

Back-translation: values

$$\frac{\cdot;\Delta;\Gamma \quad \vdash^{+} \mathbf{v} : \tau^{+}[\mu\alpha.\tau^{+}/\boldsymbol{\alpha}] \twoheadrightarrow \mathsf{v}'}{\cdot;\Delta;\Gamma \quad \vdash^{+} \mathbf{fold}_{\mu\alpha.\tau^{+}}\ \mathbf{v} : \mu\alpha.\tau^{+} \twoheadrightarrow \mathsf{fold}_{\mu\alpha.\tau}\ \mathsf{v}'}$$

$$\frac{\cdot;\Delta;\Gamma \quad \vdash \mathbf{v_1} : \tau_1^{+} \twoheadrightarrow \mathsf{v}_1' \qquad \ldots \qquad \cdot;\Delta;\Gamma \quad \vdash \mathbf{v_n} : \tau_{\mathsf{n}}^{+} \twoheadrightarrow \mathsf{v}_{\mathsf{n}}'}{\cdot;\Delta;\Gamma \quad \vdash \langle \mathbf{v_1}, \ldots, \mathbf{v_n} \rangle : \langle \tau_1, \ldots, \tau_{\mathsf{n}} \rangle^{+} \twoheadrightarrow \langle \mathsf{v}_1', \ldots, \mathsf{v}_{\mathsf{n}}' \rangle}$$

Back-translation: values (pack)

$$(\tau^{+})[\hat{\boldsymbol{\tau}}/\boldsymbol{\alpha}]$$

$$\frac{\cdot;\Delta;\Gamma \vdash \mathbf{v} : ?^{+} \twoheadrightarrow \mathsf{v}'}{\cdot;\Delta;\Gamma \vdash \mathbf{pack}\ \langle\hat{\boldsymbol{\tau}}, \mathbf{v}\rangle\ \mathbf{as}\ \exists\alpha.\tau^{+} : \exists\alpha.\tau^{+} \twoheadrightarrow}$$

Back-translation: values (pack)

$$(\tau^{+})[\hat{\boldsymbol{\tau}}/\boldsymbol{\alpha}]$$

$$\downarrow$$

$$\frac{\cdot;\Delta;\Gamma \quad \vdash \mathbf{v} : \quad ? \quad {}^{+} \twoheadrightarrow \mathsf{v}'}{\cdot;\Delta;\Gamma \quad \vdash \mathbf{pack}\ \langle\hat{\boldsymbol{\tau}}, \mathbf{v}\rangle\ \mathbf{as}\ \exists\alpha.\tau^{+} : \exists\alpha.\tau^{+} \twoheadrightarrow \mathsf{pack}\ \langle ?, \mathsf{v}'\rangle\ \mathsf{as}\ \exists\alpha.\tau}$$

Back-translation: values (pack)

$$(\tau^{+})[\hat{\boldsymbol{\tau}}/\boldsymbol{\alpha}]$$

$$\downarrow$$

$$\frac{\cdot; \Delta; \Gamma \vdash \mathbf{v} : \tau[\hat{\tau}/\alpha]^{+} \twoheadrightarrow \mathsf{v}'}{\cdot; \Delta; \Gamma \vdash \mathbf{pack}\ \langle \hat{\boldsymbol{\tau}}, \mathbf{v} \rangle\ \mathbf{as}\ \exists \alpha.\tau^{+} : \exists \alpha.\tau^{+} \twoheadrightarrow \mathsf{pack}\ \langle \mathbf{?}, \mathsf{v}' \rangle\ \mathsf{as}\ \exists \alpha.\tau}$$

Back-translation: values (pack)

$$(\tau^{+})[\hat{\boldsymbol{\tau}}/\boldsymbol{\alpha}]$$

$$\downarrow$$

$$\frac{\cdot; \Delta; \Gamma \vdash \mathbf{v} : \tau[\hat{\tau}/\alpha]^{+} \twoheadrightarrow \mathsf{v}'}{\cdot; \Delta; \Gamma \vdash \mathbf{pack}\ \langle \hat{\boldsymbol{\tau}}, \mathbf{v} \rangle\ \mathbf{as}\ \exists\alpha.\tau^{+} : \exists\alpha.\tau^{+} \twoheadrightarrow \mathsf{pack}\ \langle \hat{\tau}, \mathsf{v}' \rangle\ \mathsf{as}\ \exists\alpha.\tau}$$

Back-translation: values (pack)

$$(\tau^{+})[\hat{\boldsymbol{\tau}}/\boldsymbol{\alpha}]$$

$$\downarrow$$

$$\frac{\cdot; \Delta; \Gamma \vdash \mathbf{v} : \tau[\hat{\tau}/\alpha]^{+} \twoheadrightarrow \mathsf{v}'}{\cdot; \Delta; \Gamma \vdash \mathbf{pack}\ \langle \hat{\boldsymbol{\tau}}, \mathbf{v} \rangle\ \mathbf{as}\ \exists\alpha.\tau^{+} : \exists\alpha.\tau^{+} \twoheadrightarrow \mathsf{pack}\ \langle \hat{\tau}, \mathsf{v}' \rangle\ \mathsf{as}\ \exists\alpha.\tau}$$

Need to require that witness type ($\hat{\boldsymbol{\tau}}$) of any package of type $\exists\alpha.\tau^{+}$ must be a translation type

Back-translation: values (pack)

$$(\tau^{+})[\hat{\boldsymbol{\tau}}/\boldsymbol{\alpha}]$$

$$\frac{\hat{\boldsymbol{\tau}}^{\twoheadrightarrow} = \hat{\tau} \qquad \cdot;\Delta;\Gamma \qquad \vdash \mathbf{v} : \tau[\hat{\tau}/\alpha]^{+} \twoheadrightarrow \mathsf{v}'}{\cdot;\Delta;\Gamma \qquad \vdash \mathbf{pack}\ \langle\hat{\boldsymbol{\tau}}, \mathbf{v}\rangle\ \mathbf{as}\ \exists\alpha.\tau^{+} : \exists\alpha.\tau^{+} \twoheadrightarrow \mathsf{pack}\ \langle\hat{\tau}, \mathsf{v}'\rangle\ \mathsf{as}\ \exists\alpha.\tau}$$

Need to require that witness type ($\hat{\boldsymbol{\tau}}$) of any package of type $\exists\alpha.\tau^{+}$ must be a translation type

Back-translation: values (pack)

$$(\tau^{+})[\hat{\boldsymbol{\tau}}/\boldsymbol{\alpha}]$$

$$\downarrow$$

$$\frac{\hat{\boldsymbol{\tau}}^{\twoheadrightarrow} = \hat{\tau} \qquad \cdot;\Delta;\Gamma \qquad \vdash \mathbf{v} : \tau[\hat{\tau}/\alpha]^{+} \twoheadrightarrow \mathsf{v}'}{\cdot;\Delta;\Gamma \qquad \vdash \mathbf{pack}\ \langle \hat{\boldsymbol{\tau}}, \mathbf{v} \rangle\ \mathbf{as}\ \exists\alpha.\tau^{+} : \exists\alpha.\tau^{+} \twoheadrightarrow \mathsf{pack}\ \langle \hat{\tau}, \mathsf{v}' \rangle\ \mathsf{as}\ \exists\alpha.\tau}$$

Need to require that witness type ($\hat{\boldsymbol{\tau}}$) of any package of type $\exists\alpha.\tau^{+}$ must be a translation type

Fix the type translation! Add "trans" kinds $\diamond$ to target and kinding judgment that says all translation types have kind $\diamond$

Fixing type translation...

$$\mathbf{s} ::= \diamond \mid \star$$

$$\boldsymbol{\tau} ::= \boldsymbol{\alpha} \mid \mathbf{unit} \mid \mathbf{int} \mid \forall[\overline{\boldsymbol{\alpha} :: \mathbf{s}}].(\overline{\boldsymbol{\tau}}) \to \boldsymbol{\tau} \mid \exists \boldsymbol{\alpha} :: \mathbf{s}.\boldsymbol{\tau} \mid \boldsymbol{\mu\alpha.\tau} \mid \mathbf{ref}\ \boldsymbol{\tau} \mid \langle \overline{\boldsymbol{\tau}} \rangle \mid \mathbf{cont}\ \boldsymbol{\tau}$$

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

$$\mathbf{v} ::= \mathbf{x} \mid () \mid \mathbf{n} \mid \boldsymbol{\lambda}[\overline{\boldsymbol{\alpha} :: \mathbf{s}}](\overline{\mathbf{x} : \boldsymbol{\tau}}).\mathbf{e} \mid \mathbf{pack}\ \langle \boldsymbol{\tau}, \mathbf{v} \rangle\ \mathbf{as}\ \exists \boldsymbol{\alpha} :: \mathbf{s}.\boldsymbol{\tau} \mid \mathbf{fold}_{\boldsymbol{\mu\alpha.\tau}}\ \mathbf{v} \mid \boldsymbol{\ell} \mid \langle \overline{\mathbf{v}} \rangle \mid \mathbf{cont}_{\boldsymbol{\tau}}\ \mathbf{E}$$

$$\mathbf{e} ::= \mathbf{v} \mid \mathbf{unpack}\ \langle \boldsymbol{\alpha}, \mathbf{x} \rangle = \mathbf{v}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{v}\,[]\,\overline{\mathbf{v}} \mid \mathbf{v}\,[\boldsymbol{\tau}] \mid \mathbf{v}\ \mathbf{p}\ \mathbf{v} \mid \mathbf{new}\ \mathbf{v} \mid \mathbf{v} := \mathbf{v} \mid !\mathbf{v} \mid \mathbf{unfold}\ \mathbf{v} \mid \boldsymbol{\pi}_{\mathbf{i}}(\mathbf{v})$$

$$\mid \mathbf{let}\ \mathbf{x} = \mathbf{e}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{if0}\ \mathbf{v}\ \mathbf{e}\ \mathbf{e} \mid \mathbf{call/cc}_{\boldsymbol{\tau}}(\mathbf{x}.\mathbf{e}) \mid \mathbf{throw}_{\boldsymbol{\tau}}\ \mathbf{v}\ \mathbf{to}\ \mathbf{e}$$

$$\mathbf{E} ::= [\cdot] \mid \mathbf{let}\ \mathbf{x} = \mathbf{E}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{throw}_{\boldsymbol{\tau}}\ \mathbf{v}\ \mathbf{to}\ \mathbf{E}$$

Fixing type translation...

$$\mathbf{s} \;::=\; \diamond \mid \star$$

$$\boldsymbol{\tau} \;::=\; \boldsymbol{\alpha} \mid \mathbf{unit} \mid \mathbf{int} \mid \forall[\overline{\boldsymbol{\alpha}::\mathbf{s}}].(\overline{\boldsymbol{\tau}}) \to \boldsymbol{\tau} \mid \exists \boldsymbol{\alpha}::\mathbf{s}.\boldsymbol{\tau} \mid \boldsymbol{\mu\alpha.\tau} \mid \mathbf{ref}\ \boldsymbol{\tau} \mid \langle \overline{\boldsymbol{\tau}} \rangle \mid \mathbf{cont}\ \boldsymbol{\tau}$$

$$\mathbf{p} \;::=\; + \mid - \mid *$$

$$\mathbf{v} \;::=\; \mathbf{x} \mid () \mid \mathbf{n} \mid \boldsymbol{\lambda}[\overline{\boldsymbol{\alpha}::\mathbf{s}}](\overline{\mathbf{x}:\boldsymbol{\tau}}).\mathbf{e} \mid \mathbf{pack}\ \langle \boldsymbol{\tau}, \mathbf{v} \rangle\ \mathbf{as}\ \exists \boldsymbol{\alpha}::\mathbf{s}.\boldsymbol{\tau} \mid \mathbf{fold}_{\boldsymbol{\mu\alpha.\tau}}\ \mathbf{v} \mid \boldsymbol{\ell} \mid \langle \overline{\mathbf{v}} \rangle \mid \mathbf{cont}_{\boldsymbol{\tau}}\ \mathbf{E}$$

$$\mathbf{e} \;::=\; \mathbf{v} \mid \mathbf{unpack}\ \langle \boldsymbol{\alpha}, \mathbf{x} \rangle = \mathbf{v}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{v}\ []\ \overline{\mathbf{v}} \mid \mathbf{v}\ [\boldsymbol{\tau}] \mid \mathbf{v}\ \mathbf{p}\ \mathbf{v} \mid \mathbf{new}\ \mathbf{v} \mid \mathbf{v} := \mathbf{v} \mid !\mathbf{v} \mid \mathbf{unfold}\ \mathbf{v} \mid \boldsymbol{\pi}_{\mathbf{i}}(\mathbf{v})$$
$$\mid \mathbf{let}\ \mathbf{x} = \mathbf{e}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{if0}\ \mathbf{v}\ \mathbf{e}\ \mathbf{e} \mid \mathbf{call/cc}_{\boldsymbol{\tau}}(\mathbf{x}.\mathbf{e}) \mid \mathbf{throw}_{\boldsymbol{\tau}}\ \mathbf{v}\ \mathbf{to}\ \mathbf{e}$$

$$\mathbf{E} \;::=\; [\cdot] \mid \mathbf{let}\ \mathbf{x} = \mathbf{E}\ \mathbf{in}\ \mathbf{e} \mid \mathbf{throw}_{\boldsymbol{\tau}}\ \mathbf{v}\ \mathbf{to}\ \mathbf{E}$$

$$\alpha^{+} = \boldsymbol{\alpha}$$
$$\mathsf{unit}^{+} = \mathbf{unit}$$
$$\mathsf{ref}\ \tau^{+} = \mathbf{ref}\ \tau^{+}$$
$$\langle \tau_1, \ldots, \tau_n \rangle^{+} = \langle \tau_1{}^{+}, \ldots, \tau_n{}^{+} \rangle$$

$$\forall[\overline{\alpha}].(\overline{\tau}) \to \tau'^{+} = \exists \boldsymbol{\beta}::\diamond.\ \langle (\forall[\overline{\boldsymbol{\alpha}::\diamond}].(\boldsymbol{\beta}, \overline{\tau^{+}}) \to \tau'^{+}), \boldsymbol{\beta} \rangle$$
$$\exists \alpha.\tau^{+} = \exists \boldsymbol{\alpha}::\diamond.\ \tau^{+}$$
$$\mu\alpha.\tau^{+} = \boldsymbol{\mu\alpha}.\tau^{+}$$
$$\mathsf{int}^{+} = \mathbf{int}$$

Now our translation *is* fully abstract

$$\tau = (\mathsf{unit} \to \mathsf{unit}) \to \mathsf{int})$$

$$\mathsf{e}_1 = \mathsf{let\ x = new\ 0\ in}$$
$$\lambda \mathsf{f}.(\mathsf{x} := 0;\ \mathsf{f}\ ();\ \mathsf{x} := 1;\ \mathsf{f}\ ();\ !\mathsf{x})$$

$$\mathsf{e}_2 = \lambda \mathsf{f}.(\mathsf{f}\ ();\ \mathsf{f}\ ();\ 1)$$

g wants $(\mathsf{unit} \to \mathsf{unit})^{+}$

$$\mathsf{C} = \mathsf{let\ g} = [\cdot]\ \mathsf{in\ let\ b = new\ ff\ in}$$
$$\mathsf{letf} = (\lambda\_.\ \mathsf{if}\ !\mathsf{b}\ \mathsf{then\ call/cc}(\mathsf{k}.\ \mathsf{g}\ (\lambda\_.\ \mathsf{throw}\ ()\ \mathsf{to\ k}))$$
$$\mathsf{else\ b} := \mathsf{tt})\ \mathsf{in}$$
$$\mathsf{g\ f}$$

Now our translation *is* fully abstract

$\tau = (\mathsf{unit} \to \mathsf{unit}) \to \mathsf{int})$

$\mathsf{e}_1 = \mathsf{let}\ \mathsf{x} = \mathsf{new}\ 0\ \mathsf{in}$

$\qquad \lambda \mathsf{f}.(\mathsf{x} := 0;\ \mathsf{f}\ ();\ \mathsf{x} := 1;\ \mathsf{f}\ ();\ !\mathsf{x})$

$\mathsf{e}_2 = \lambda \mathsf{f}.(\mathsf{f}\ ();\ \mathsf{f}\ ();\ 1)$

g wants $(\mathsf{unit} \to \mathsf{unit})^{+}$

$\mathsf{C} = \mathsf{let}\ \mathsf{g} = [\cdot]\ \mathsf{in}\ \mathsf{let}\ \mathsf{b} = \mathsf{new}\ \mathsf{ff}\ \mathsf{in}$

$\qquad \mathsf{let} \mathsf{f} = (\lambda\_.\ \mathsf{if}\ !\mathsf{b}\ \mathsf{then}\ \mathsf{call/cc}(\mathsf{k}.\ \mathsf{g}\ (\lambda\_.\ \mathsf{throw}\ ()\ \mathsf{to}\ \mathsf{k}))$

$\qquad\qquad \mathsf{else}\ \mathsf{b} := \mathsf{tt})\ \mathsf{in}$

$\qquad \mathsf{g}\ \mathsf{f}$

$\mathbf{pack}\ \langle \mathbf{cont}\ \boldsymbol{\tau},\ \boldsymbol{\lambda}(\mathbf{z}, \_).\ \mathbf{throw}\ ()\ \mathbf{to}\ \boldsymbol{\pi}_1\ \mathbf{z} \rangle$

Back-translation: values (pack-closure)

$$\frac{\begin{array}{c} \mathbf{v}_\exists = \mathbf{pack}\ \langle \boldsymbol{\tau}_{\mathbf{env}}, \langle \mathbf{v}, \mathbf{v}_{\mathbf{env}} \rangle\rangle\ \mathbf{as}\ \exists \boldsymbol{\alpha}' :: \diamond .\ \langle (\forall[\overline{\boldsymbol{\alpha} :: \diamond}].(\boldsymbol{\alpha}', \overline{\tau}^{+}) \rightarrow \tau'^{+}), \boldsymbol{\alpha}' \rangle \\ \mathbf{v} = \boldsymbol{\lambda}[\overline{\boldsymbol{\alpha} :: \diamond}](\mathbf{z} : \tau_{\mathsf{env}}{}^{+}, \overline{\mathbf{x} : \tau^{+}}).\mathbf{e} \qquad \boldsymbol{\tau}_{\mathbf{env}} \overset{\twoheadrightarrow}{=} \tau_{\mathsf{env}} \\ \cdot; \Delta; \Gamma \vdash \mathbf{v}_{\mathbf{env}} : \tau_{\mathsf{env}}{}^{+} \twoheadrightarrow \mathsf{v}'_{\mathsf{env}} \qquad \cdot; \overline{\boldsymbol{\alpha} :: \diamond}; \mathbf{z} : \tau_{\mathsf{env}}{}^{+}, \overline{\mathbf{x} : \tau^{+}} | \cdot; \cdot \vdash \mathbf{e} : \tau'^{+} \twoheadrightarrow \mathsf{e}' \end{array}}{\cdot; \Delta; \Gamma \vdash^{+} \mathbf{v}_\exists : (\forall[\overline{\alpha}].(\overline{\tau}) \rightarrow \tau')^{+} \twoheadrightarrow \lambda[\overline{\alpha}](\overline{\mathsf{x} : \tau}).\mathsf{let}\,\mathsf{z} = \mathsf{v}'_{\mathsf{env}}\ \mathsf{in}\ \mathsf{e}'}$$

Back-translation: trans subterms (easy)

$$\frac{\cdot;\Delta;\Gamma \vdash \mathbf{v} : \mathsf{int}^{+} \twoheadrightarrow \mathsf{v}' \qquad \cdot;\Delta;\Gamma \vdash \mathbf{e_1} : \tau^{+} \twoheadrightarrow \mathsf{e}_1' \qquad \cdot;\Delta;\Gamma \vdash \mathbf{e_2} : \tau^{+} \twoheadrightarrow \mathsf{e}_2'}{\cdot;\Delta;\Gamma \vdash \mathbf{if0\ v\ e_1\ e_2} : \tau^{+} \twoheadrightarrow \mathsf{if0}\,\mathsf{v}'\ \mathsf{e}_1'\ \mathsf{e}_2'}$$

$$\frac{\cdot;\Delta;\Gamma \vdash \mathbf{e_1} : \tau_1{}^{+} \twoheadrightarrow \mathsf{e}_1' \qquad \cdot;\Delta;\Gamma,\mathbf{x}:\tau_1{}^{+} \vdash \mathbf{e_2} : \tau_2{}^{+} \twoheadrightarrow \mathsf{e}_2'}{\cdot;\Delta;\Gamma \vdash \mathbf{let\,x = e_1\ in\ e_2} : \tau_2{}^{+} \twoheadrightarrow \mathsf{let}\,\mathsf{x} = \mathsf{e}_1'\ \mathsf{in}\ \mathsf{e}_2'}$$

Back-translation: trans subterms (easy)

$$\frac{\cdot;\Delta;\Gamma \vdash \mathbf{v} : \exists\alpha.\tau'^{+} \twoheadrightarrow \mathsf{v'} \qquad \cdot;\Delta,\boldsymbol{\alpha}::\diamond;\Gamma,\mathbf{x}:\tau'^{+} \vdash \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e'}}{\cdot;\Delta;\Gamma \vdash \mathbf{unpack}\ \langle\boldsymbol{\alpha},\mathbf{x}\rangle = \mathbf{v}\ \mathbf{in}\ \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{unpack}\ \langle\alpha,\mathsf{x}\rangle = \mathsf{v'}\ \mathsf{in}\ \mathsf{e'}}$$

$$\frac{\cdot;\Delta;\Gamma \vdash \mathbf{v} : \mu\alpha.\tau^{+} \twoheadrightarrow \mathsf{v'}}{\cdot;\Delta;\Gamma \vdash \mathbf{unfold}\ \mathbf{v} : \tau^{+}[\mu\alpha.\tau^{+}/\boldsymbol{\alpha}] \twoheadrightarrow \mathsf{unfold}\ \mathsf{v'}}$$

$$\frac{\cdot;\Delta;\Gamma \vdash \mathbf{v} : \tau^{+} \twoheadrightarrow \mathsf{v'}}{\cdot;\Delta;\Gamma \vdash^{+} \mathbf{new}\ \mathbf{v} : \mathsf{ref}\ \tau^{+} \twoheadrightarrow \mathsf{new}\ \mathsf{v'}} \qquad \frac{\cdot;\Delta;\Gamma \vdash^{+} \mathbf{v} : \mathsf{ref}\ \tau^{+} \twoheadrightarrow \mathsf{v'}}{\cdot;\Delta;\Gamma \vdash \mathbf{!v} : \tau^{+} \twoheadrightarrow \mathsf{!v'}}$$

$$\frac{\cdot;\Delta;\Gamma \vdash \mathbf{v_1} : \mathsf{ref}\ \tau^{+} \twoheadrightarrow \mathsf{v'_1} \qquad \cdot;\Delta;\Gamma \vdash \mathbf{v_2} : \tau^{+} \twoheadrightarrow \mathsf{v'_2}}{\cdot;\Delta;\Gamma \vdash^{+} \mathbf{v_1} := \mathbf{v_2} : \mathsf{unit}^{+} \twoheadrightarrow \mathsf{v'_1} := \mathsf{v'_2}}$$

Back-translation: trans subterms

$$\frac{}{\cdot;\Delta;\Gamma,\mathbf{x}:(\forall[\overline{\alpha}].(\overline{\tau'})\to\tau'')^{+}\cdot \quad \vdash \mathbf{unpack}\,\langle\boldsymbol{\beta},\mathbf{y}\rangle=\mathbf{x}\ \mathbf{in}\ \mathbf{e}:\tau^{+}\twoheadrightarrow \mathsf{e}'}$$

Back-translation: trans subterms

$$\frac{}{\cdot;\Delta;\Gamma,\mathbf{x}:(\forall[\overline{\alpha}].(\overline{\tau'})\rightarrow\tau'')^{+}\mid\quad\mathbf{\Phi}\vdash\mathbf{unpack}\,\langle\boldsymbol{\beta},\mathbf{y}\rangle=\mathbf{x}\ \mathbf{in}\ \mathbf{e}:\tau^{+}\twoheadrightarrow\mathsf{e}'}$$

Back-translation: trans subterms

$$\frac{\begin{array}{c}\boldsymbol{\Phi'} = \boldsymbol{\Phi}, (\boldsymbol{\beta_{env}} :: \diamond, \mathbf{x_f} : \forall[\overline{\boldsymbol{\alpha} :: \diamond}].(\boldsymbol{\beta_{env}}, \tau'^{+}) \rightarrow \tau''^{+}, \mathbf{x_{env}} : \boldsymbol{\beta_{env}}, \mathsf{x}) \\ \cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi'} \vdash \mathbf{e}[\langle \mathbf{x_f}, \mathbf{x_{env}} \rangle / \mathbf{y}] : \tau^{+} \twoheadrightarrow \mathsf{e}'\end{array}}{\cdot; \Delta; \Gamma, \mathbf{x} : (\forall[\overline{\alpha}].(\overline{\tau'}) \rightarrow \tau'')^{+}| \quad \boldsymbol{\Phi} \vdash \mathbf{unpack}\ \langle \boldsymbol{\beta}, \mathbf{y} \rangle = \mathbf{x}\ \mathbf{in}\ \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}'}$$

Back-translation: trans subterms

$$\frac{\begin{array}{c}\mathbf{\Phi'} = \mathbf{\Phi}, (\boldsymbol{\beta}_{\mathbf{env}} :: \diamond, \mathbf{x_f} : \forall[\overline{\boldsymbol{\alpha} :: \diamond}].(\boldsymbol{\beta}_{\mathbf{env}}, \tau'^{+}) \rightarrow \tau''^{+}, \mathbf{x_{env}} : \boldsymbol{\beta}_{\mathbf{env}}, \mathsf{x}) \\ \cdot; \Delta; \Gamma| \quad \mathbf{\Phi'} \vdash \mathbf{e}[\langle \mathbf{x_f}, \mathbf{x_{env}} \rangle / \mathbf{y}] : \tau^{+} \twoheadrightarrow \mathsf{e}'\end{array}}{\cdot; \Delta; \Gamma, \mathbf{x} : (\forall[\overline{\alpha}].(\overline{\tau'}) \rightarrow \tau'')^{+} | \quad \mathbf{\Phi} \vdash \mathbf{unpack}\ \langle \boldsymbol{\beta}, \mathbf{y} \rangle = \mathbf{x}\ \mathbf{in}\ \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}'}$$

$$\frac{\boldsymbol{\tau_0} = \tau_0 \qquad \tau^{+} = \tau''\overline{[\tau_0/\alpha]}^{+} \qquad \dfrac{(\boldsymbol{\beta}_{\mathbf{env}} :: \diamond, \mathbf{x_f} : \forall[\overline{\boldsymbol{\alpha} :: \diamond}].(\boldsymbol{\beta}_{\mathbf{env}}, \tau'^{+}) \rightarrow \tau''^{+}, \mathbf{x_{env}} : \boldsymbol{\beta}_{\mathbf{env}}, \mathsf{x}) \in \mathbf{\Phi}}{\cdot; \Delta; \Gamma| \quad \mathbf{\Phi} \vdash \mathbf{v} : \tau'\overline{[\tau_0/\alpha]}^{+} \twoheadrightarrow \mathsf{v}}}{\cdot; \Delta; \Gamma| \quad \mathbf{\Phi} \vdash ((\mathbf{x_f}\ [\overline{\boldsymbol{\tau_0}}])\ []\ \langle \mathbf{x_{env}}, \overline{\mathbf{v}} \rangle) : \tau^{+} \twoheadrightarrow \mathsf{x}\ [\overline{\tau_0}]\ \overline{\mathsf{v}}}$$

Back-translation: non-trans subterms

$$\frac{\mathbf{v_0} = (\boldsymbol{\lambda}[\overline{\boldsymbol{\alpha} :: \mathbf{s}}](\overline{\mathbf{x} : \boldsymbol{\tau}'}).\mathbf{e})\ [\overline{\boldsymbol{\tau}''}] \qquad \cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[\overline{\boldsymbol{\tau}''/\boldsymbol{\alpha}}][\overline{\mathbf{v}/\mathbf{x}}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash \mathbf{v_0}\ []\ \overline{\mathbf{v}} : \tau^{+} \twoheadrightarrow \mathsf{e}'}$$

Back-translation: non-trans subterms

$$\frac{\mathbf{v_0} = (\boldsymbol{\lambda}[\overline{\boldsymbol{\alpha} :: \mathbf{s}}](\overline{\mathbf{x} : \boldsymbol{\tau}'}).\mathbf{e})\ [\overline{\boldsymbol{\tau}''}] \qquad \cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[\overline{\boldsymbol{\tau}''/\boldsymbol{\alpha}}][\overline{\mathbf{v}/\mathbf{x}}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash \mathbf{v_0}\ []\ \overline{\mathbf{v}} : \tau^{+} \twoheadrightarrow \mathsf{e}'}$$

$$\frac{}{\cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash \mathbf{let}\ \mathbf{x} = (\mathbf{new}\ \mathbf{v})\ \mathbf{in}\ \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}'} \ (\nexists \tau'.\tau'^{+} = \boldsymbol{\tau}')$$

Back-translation: non-trans subterms

$$\frac{\mathbf{v_0} = (\boldsymbol{\lambda}[\overline{\boldsymbol{\alpha} :: \mathbf{s}}](\overline{\mathbf{x} : \boldsymbol{\tau'}}).\mathbf{e})\ [\overline{\boldsymbol{\tau''}}] \qquad \cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash_\Omega \mathbf{e}[\overline{\boldsymbol{\tau''}/\boldsymbol{\alpha}}][\overline{\mathbf{v}/\mathbf{x}}] : \tau^+ \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash \mathbf{v_0}\ []\ \overline{\mathbf{v}} : \tau^+ \twoheadrightarrow \mathsf{e}'}$$

$$\frac{}{\cdot; \Delta; \Gamma|\mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let}\ \mathbf{x} = (\mathbf{new}\ \mathbf{v})\ \mathbf{in}\ \mathbf{e} : \tau^+ \twoheadrightarrow \mathsf{e}'}\ (\nexists \tau'.\tau'^+ = \boldsymbol{\tau'})$$

Back-translation: non-trans subterms

$$\frac{\mathbf{v_0} = (\boldsymbol{\lambda}[\overline{\boldsymbol{\alpha} :: \mathbf{s}}](\overline{\mathbf{x} : \boldsymbol{\tau}'}).\mathbf{e})\ [\overline{\boldsymbol{\tau}''}] \qquad \cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash_\Omega \mathbf{e}[\overline{\boldsymbol{\tau}''/\boldsymbol{\alpha}}][\overline{\mathbf{v}/\mathbf{x}}] : \tau^+ \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma| \quad \boldsymbol{\Phi} \vdash \mathbf{v_0}\ []\ \overline{\mathbf{v}} : \tau^+ \twoheadrightarrow \mathsf{e}'}$$

$$\frac{\boldsymbol{\ell} : \boldsymbol{\tau}' \notin \mathrm{dom}(\mathbf{H}) \qquad \cdot; \Delta; \Gamma|\mathbf{H}[\boldsymbol{\ell} : \boldsymbol{\tau}' \mapsto \mathbf{v}]; \boldsymbol{\Phi} \vdash_\Omega \mathbf{e}[\boldsymbol{\ell}/\mathbf{x}] : \tau^+ \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma|\mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let}\ \mathbf{x} = (\mathbf{new}\ \mathbf{v})\ \mathbf{in}\ \mathbf{e} : \tau^+ \twoheadrightarrow \mathsf{e}'}\ (\nexists \tau'.\tau'^+ = \boldsymbol{\tau}')$$

Back-translation: non-trans subterms

$$\frac{\mathbf{v_0} = (\boldsymbol{\lambda}[\overline{\boldsymbol{\alpha} :: \mathbf{s}}](\overline{\mathbf{x} : \boldsymbol{\tau}'}).\mathbf{e})\ [\overline{\boldsymbol{\tau}''}] \qquad \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[\overline{\boldsymbol{\tau}''/\boldsymbol{\alpha}}][\overline{\mathbf{v}/\mathbf{x}}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{v_0}\ []\ \overline{\mathbf{v}} : \tau^{+} \twoheadrightarrow \mathsf{e}'}$$

$$\frac{\boldsymbol{\ell} : \boldsymbol{\tau}' \notin \mathrm{dom}(\mathbf{H}) \qquad \cdot; \Delta; \Gamma | \mathbf{H}[\boldsymbol{\ell} : \boldsymbol{\tau}' \mapsto \mathbf{v}]; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[\boldsymbol{\ell}/\mathbf{x}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x = (new\ v)\ in\ e} : \tau^{+} \twoheadrightarrow \mathsf{e}'} \ (\nexists \tau'. \tau'^{+} = \boldsymbol{\tau}')$$

Back-translation: non-trans subterms

$$\frac{\mathbf{v_0} = (\boldsymbol{\lambda}[\overline{\boldsymbol{\alpha} :: \mathbf{s}}](\overline{\mathbf{x} : \boldsymbol{\tau'}}).\mathbf{e})\ [\overline{\boldsymbol{\tau''}}] \qquad \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[\overline{\boldsymbol{\tau''}/\boldsymbol{\alpha}}][\overline{\mathbf{v}/\mathbf{x}}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{v_0}\ []\ \overline{\mathbf{v}} : \tau^{+} \twoheadrightarrow \mathsf{e}'}$$

$$\frac{\boldsymbol{\ell} : \boldsymbol{\tau'} \notin \mathrm{dom}(\mathbf{H}) \qquad \cdot; \Delta; \Gamma | \mathbf{H}[\boldsymbol{\ell} : \boldsymbol{\tau'} \mapsto \mathbf{v}]; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[\boldsymbol{\ell}/\mathbf{x}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x = (new\ v)\ in\ e} : \tau^{+} \twoheadrightarrow \mathsf{e}'} \ (\nexists \tau'.\tau'^{+} = \boldsymbol{\tau'})$$

$$\frac{\mathbf{H}(\boldsymbol{\ell} : \boldsymbol{\tau'}) = \mathbf{v} \qquad \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[\mathbf{v}/\mathbf{x}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x = !}\boldsymbol{\ell}\ \mathbf{in\ e} : \tau^{+} \twoheadrightarrow \mathsf{e}'} \ (\nexists \tau'.\tau'^{+} = \boldsymbol{\tau'})$$

$$\frac{\boldsymbol{\ell} : \boldsymbol{\tau'} \in \mathrm{dom}(\mathbf{H}) \qquad \cdot; \Delta; \Gamma | \mathbf{H}[\boldsymbol{\ell} : \boldsymbol{\tau'} \mapsto \mathbf{v}]; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[()/\mathbf{x}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x = (}\boldsymbol{\ell}\ \mathbf{:= v)\ in\ e} : \tau^{+} \twoheadrightarrow \mathsf{e}'} \ (\nexists \tau'.\tau'^{+} = \boldsymbol{\tau'})$$

Back-translation: non-trans subterms

$$\frac{\boldsymbol{\Psi_{H}}; \Delta, \boldsymbol{\Delta_{\Phi}}; \Gamma, \boldsymbol{\Gamma_{\Phi}} \vdash \mathbf{v} : \boldsymbol{\tau'} \qquad \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e}[\mathbf{v}/\mathbf{x}] : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x = v\ in\ e} : \tau^{+} \twoheadrightarrow \mathsf{e}'} \ (\nexists \tau'.\tau'^{+} = \boldsymbol{\tau'})$$

$$\frac{\begin{array}{c}\boldsymbol{\Psi_{H}}; \Delta, \boldsymbol{\Delta_{\Phi}}; \Gamma, \boldsymbol{\Gamma_{\Phi}} \vdash \mathbf{e_2} : \boldsymbol{\tau'} \\ \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x_1 = e_1\ in\ (let\ x_2 = e_2\ in\ e_3)} : \tau^{+} \twoheadrightarrow \mathsf{e}\end{array}}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x_2 = (let\ x_1 = e_1\ in\ e_2)\ in\ e_3} : \tau^{+} \twoheadrightarrow \mathsf{e}} \ (\nexists \tau'.\tau'^{+} = \boldsymbol{\tau'})$$

$$\frac{\begin{array}{cc}\boldsymbol{\Psi_{H}}; \Delta, \boldsymbol{\Delta_{\Phi}}; \Gamma, \boldsymbol{\Gamma_{\Phi}} \vdash \mathbf{e_1}, \mathbf{e_2} : \boldsymbol{\tau'} & \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{v} : \mathsf{int}^{+} \twoheadrightarrow \mathsf{v}' \\ \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x = e_1\ in\ e_3} : \tau^{+} \twoheadrightarrow \mathsf{e}_1' & \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x = e_2\ in\ e_3} : \tau^{+} \twoheadrightarrow \mathsf{e}_2'\end{array}}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{let\ x = (if0\ v\ e_1\ e_2)\ in\ e_3} : \tau^{+} \twoheadrightarrow \mathsf{if0}\ \mathsf{v}'\ \mathsf{e}_1'\ \mathsf{e}_2'} \ (\nexists \tau'.\tau'^{+} = \boldsymbol{\tau'})$$

Back-translation: well-foundedness!

$$\frac{\boldsymbol{\Psi}_{\mathbf{H}}; \Delta, \boldsymbol{\Delta}_{\boldsymbol{\Phi}}; \Gamma, \boldsymbol{\Gamma}_{\boldsymbol{\Phi}} \vdash \mathbf{e} \approx^{ctx} \boldsymbol{\Omega} : \tau^{+}}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e} : \tau^{+} \twoheadrightarrow \Omega}$$

$$\frac{\boldsymbol{\Psi}_{\mathbf{H}}; \Delta, \boldsymbol{\Delta}_{\boldsymbol{\Phi}}; \Gamma, \boldsymbol{\Gamma}_{\boldsymbol{\Phi}} \vdash \mathbf{e} \not\approx^{ctx}_{M+C} \boldsymbol{\Omega} : \tau^{+} \qquad \cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \boldsymbol{\Phi} \vdash_{\Omega} \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}'}$$

Back-translation: well-foundedness!

$$\frac{\mathbf{\Psi_H}; \Delta, \mathbf{\Delta_\Phi}; \Gamma, \mathbf{\Gamma_\Phi} \vdash \mathbf{e} \approx^{ctx} \mathbf{\Omega} : \tau^{+}}{\cdot; \Delta; \Gamma | \mathbf{H}; \mathbf{\Phi} \vdash_{\Omega} \mathbf{e} : \tau^{+} \twoheadrightarrow \Omega}$$

$$\frac{\mathbf{\Psi_H}; \Delta, \mathbf{\Delta_\Phi}; \Gamma, \mathbf{\Gamma_\Phi} \vdash \mathbf{e} \not\approx^{ctx}_{M+C} \mathbf{\Omega} : \tau^{+} \qquad \cdot; \Delta; \Gamma | \mathbf{H}; \mathbf{\Phi} \vdash \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \mathbf{\Phi} \vdash_{\Omega} \mathbf{e} : \tau^{+} \twoheadrightarrow \mathsf{e}'}$$

Intuition: have an “oracle” that checks, after every partial evaluation step, if the term is equivalent to $\mathbf{\Omega}$

Back-translation: well-foundedness!

$$\frac{\mathbf{\Psi_H}; \Delta, \mathbf{\Delta_\Phi}; \Gamma, \mathbf{\Gamma_\Phi} \vdash \mathbf{e} \approx^{ctx} \mathbf{\Omega} : \tau^+}{\cdot; \Delta; \Gamma | \mathbf{H}; \mathbf{\Phi} \vdash_\Omega \mathbf{e} : \tau^+ \twoheadrightarrow \Omega}$$

$$\frac{\mathbf{\Psi_H}; \Delta, \mathbf{\Delta_\Phi}; \Gamma, \mathbf{\Gamma_\Phi} \vdash \mathbf{e} \not\approx^{ctx}_{M+C} \mathbf{\Omega} : \tau^+ \qquad \cdot; \Delta; \Gamma | \mathbf{H}; \mathbf{\Phi} \vdash \mathbf{e} : \tau^+ \twoheadrightarrow \mathsf{e}'}{\cdot; \Delta; \Gamma | \mathbf{H}; \mathbf{\Phi} \vdash_\Omega \mathbf{e} : \tau^+ \twoheadrightarrow \mathsf{e}'}$$

Intuition: have an “oracle” that checks, after every partial evaluation step, if the term is equivalent to $\mathbf{\Omega}$

Prove backtranslation is well-founded using a logical relation.

Back-translation: call/cc, throw

Same intuition as for heap effects.

- Rules maintain "state" -- i.e., current continuation E
- for call/cc and throw subterms, do partial evaluation
- current continuation E is reset to empty when we go under a lambda

Takeaways

- Advanced languages like HTT and F* are ideal for verifying security properties alongside development of code
- Need correct and secure compilers to ensure that source-level guarantees are preserved at the target level
- To build realistic fully abstract compilers, we need proof techniques (back-translation)

- Main idea: use types/type-translation to ensure compiled code will only be run in well-behaved target contexts
- If type-translation is right, back-translation will work!

Questions?