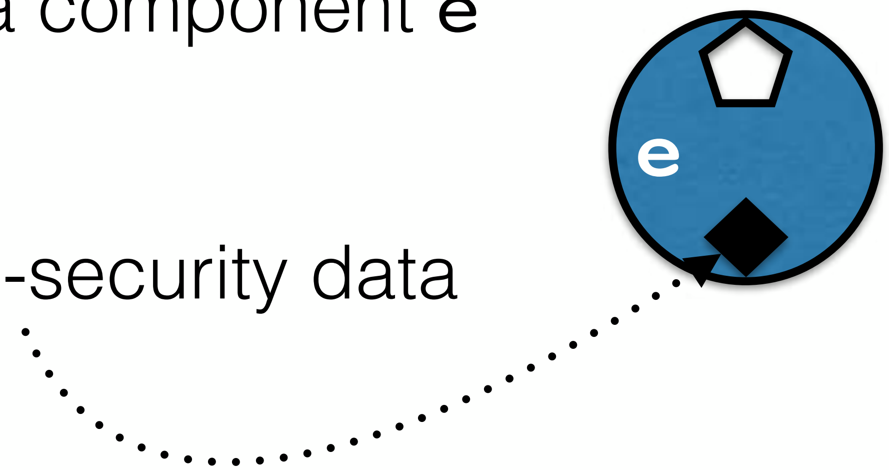
Noninterference for Free

William J. Bowman and Amal Ahmed

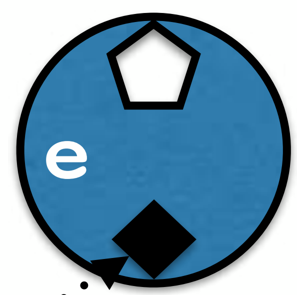
Let's write a secure program

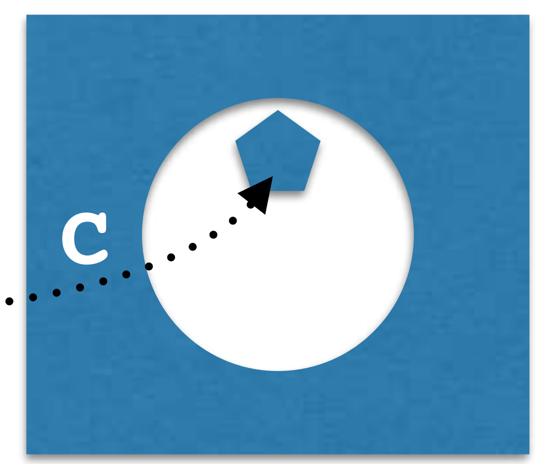
- Want to write a component **e** (browser)
- Manages high-security data (passwords)



Let’s write a secure program

- Want to write a component **e** (browser)
- Manages high-security data (passwords)
- Links with untrusted context **c** (plugins)
- **c** provides low-security inputs, reads low-security outputs

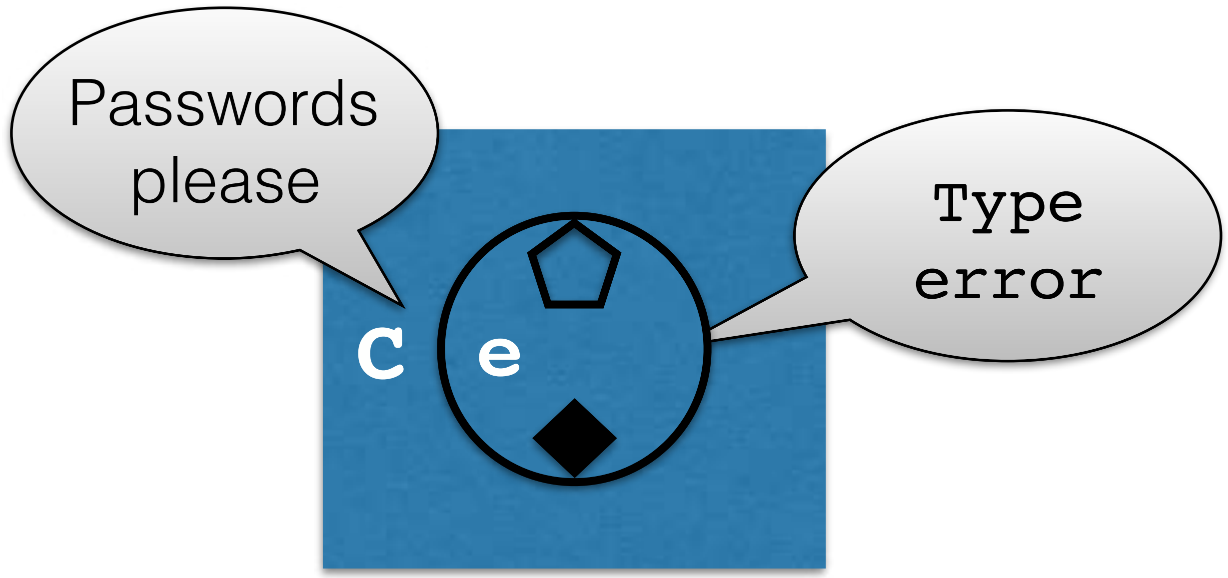




Language-based security!

Using language-based security,
we statically rule out attacks

Language-based security!

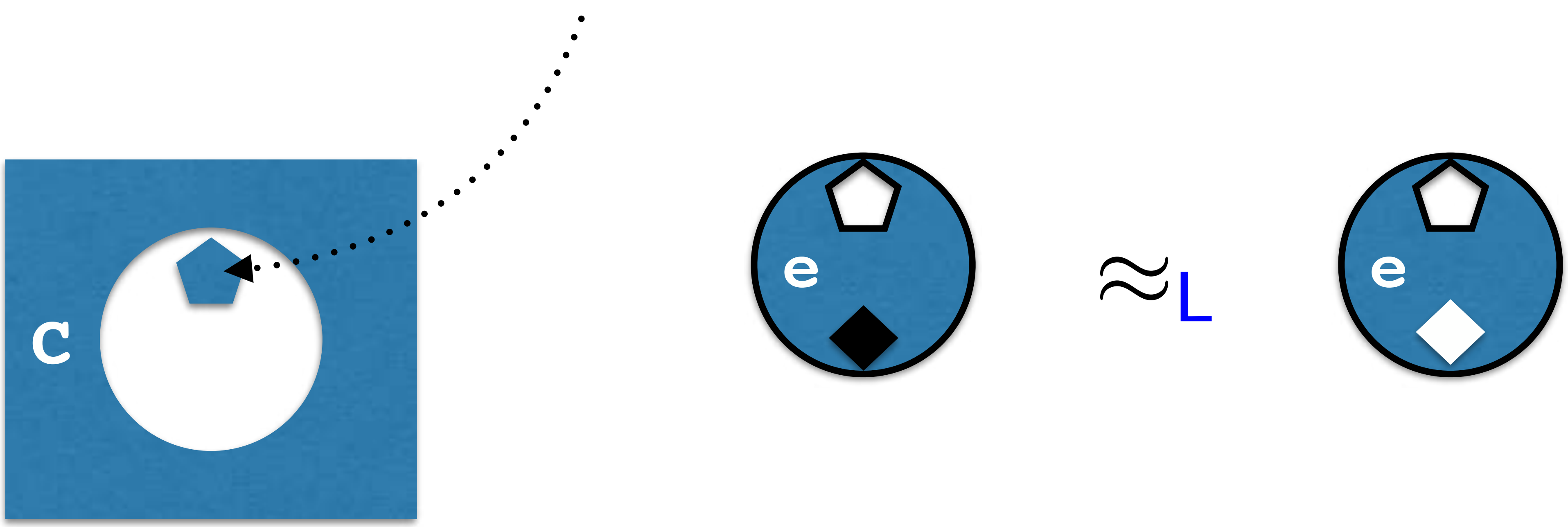


Using language-based security, we statically rule out attacks

Noninterference

Noninterference is an **equivalence property**
of any well-typed term **e**:

Given ***same*** low-level (***public***) inputs,

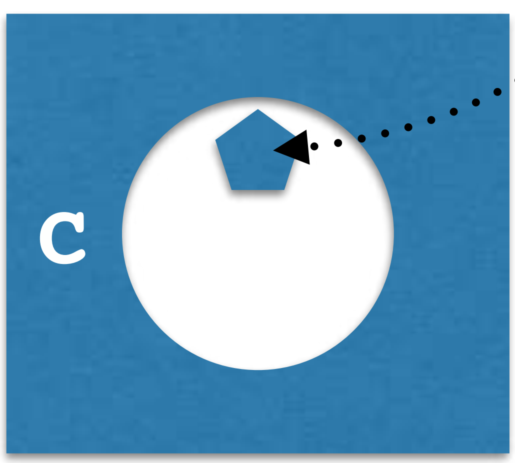


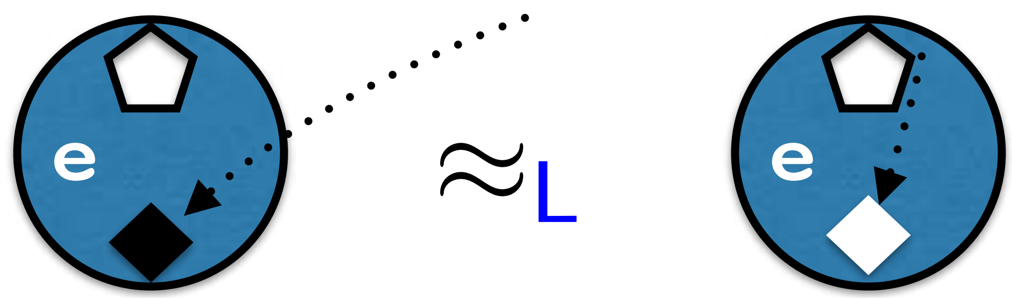
Noninterference

Noninterference is an **equivalence property** of any well-typed term **e**:

Given ***same*** low-level (***public***) inputs,

and ***different*** high-level (***private***) inputs



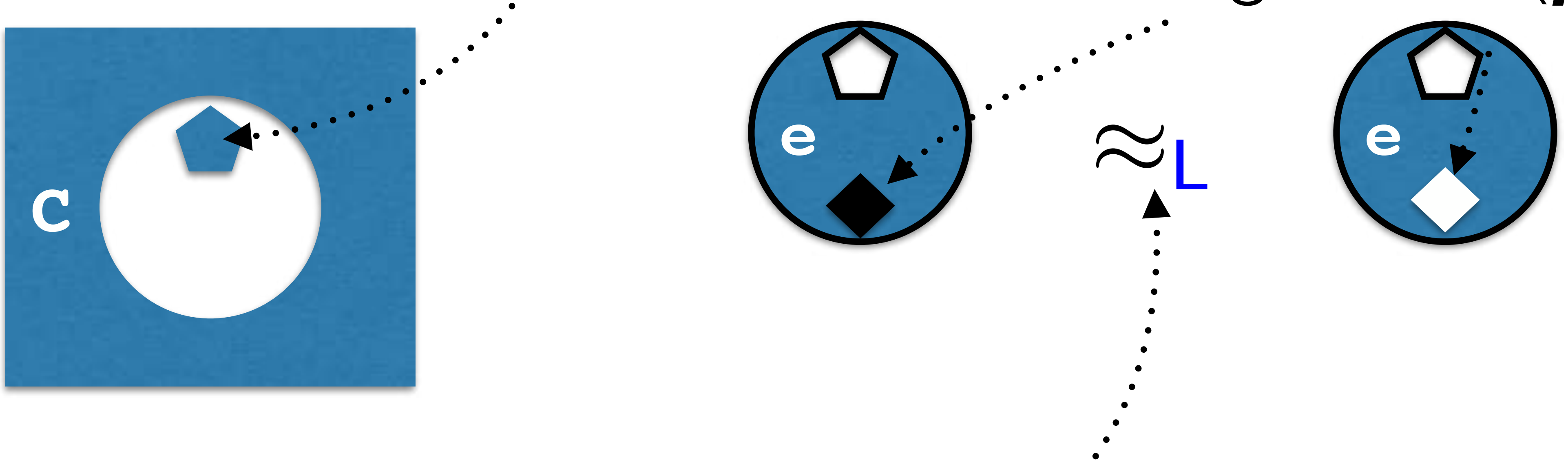


Noninterference

Noninterference is an **equivalence property**
of any well-typed term **e**:

Given ***same*** low-level (***public***) inputs,

and ***different*** high-level (***private***) inputs



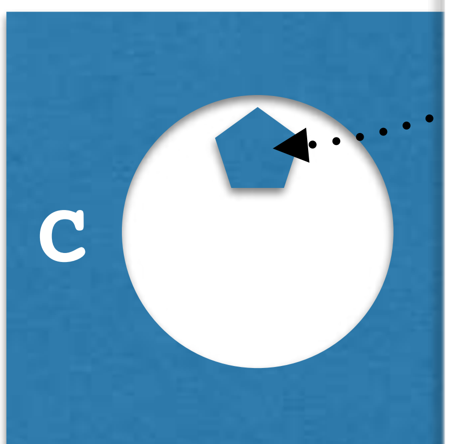
low-level outputs are indistinguishable

Noninterference

Noninterference is an **equivalence property**

Given ***same***

vate) inputs

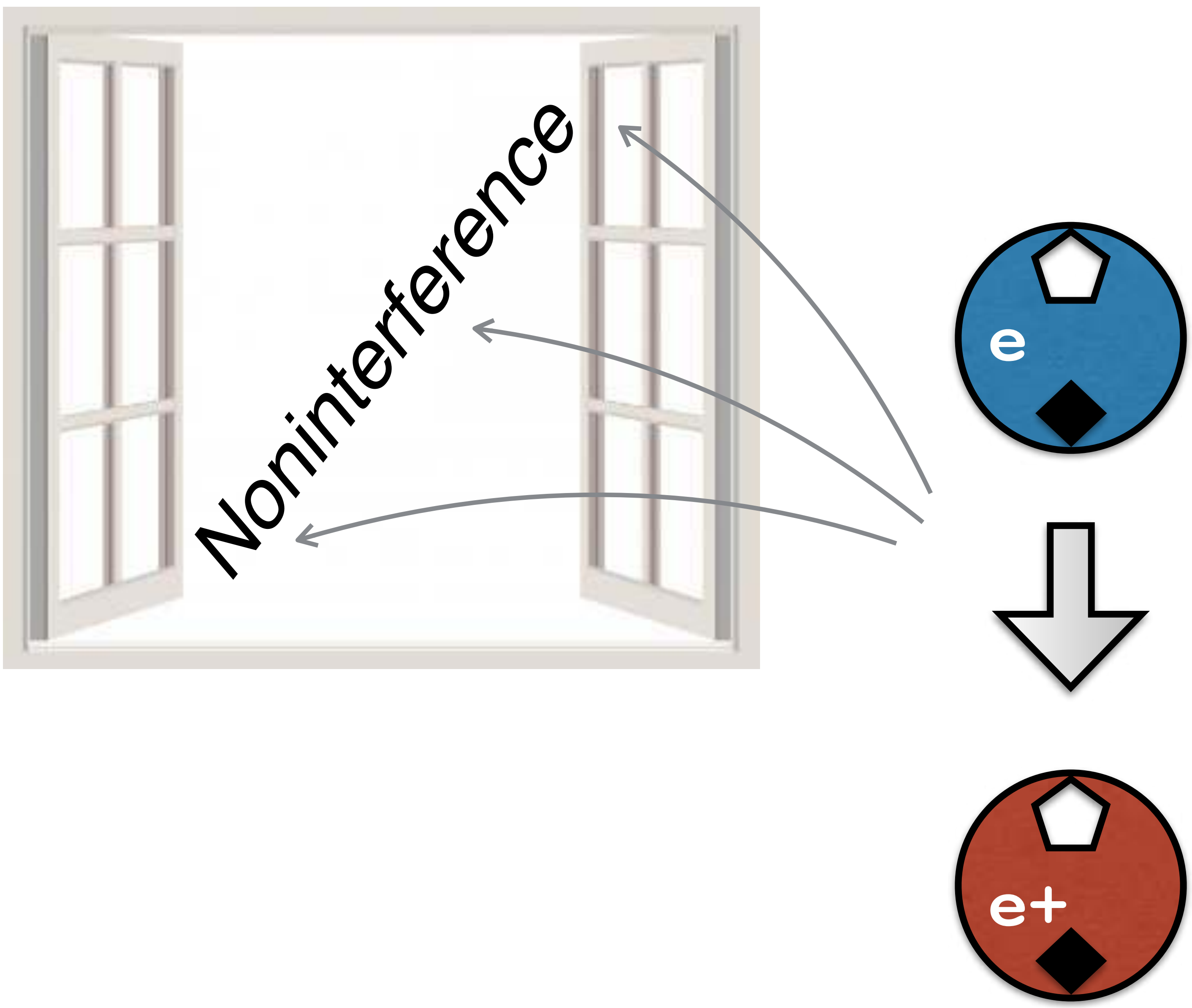


Security Solved!

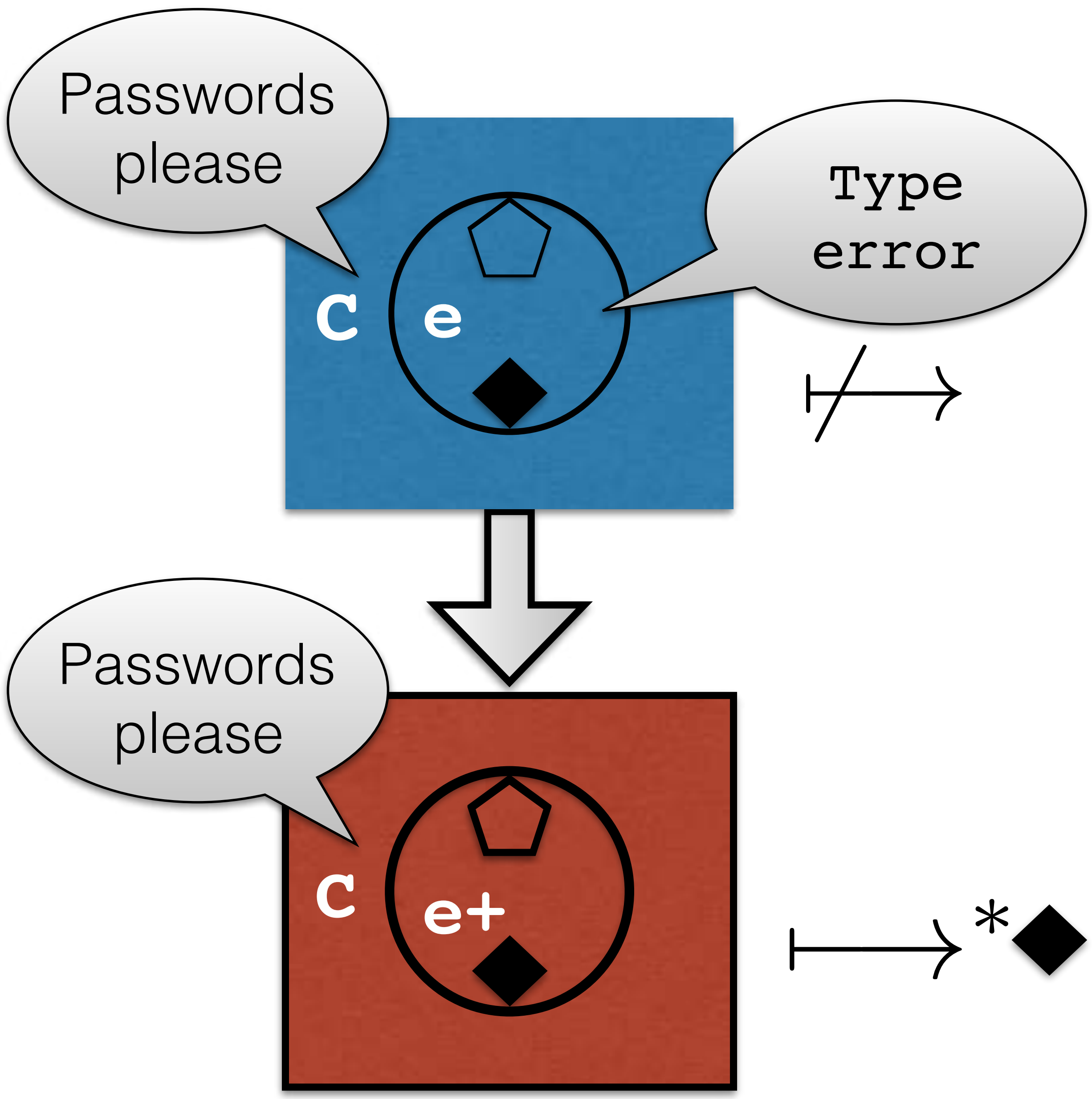
low-level outputs are indistinguishable

WRONG

Because compilers

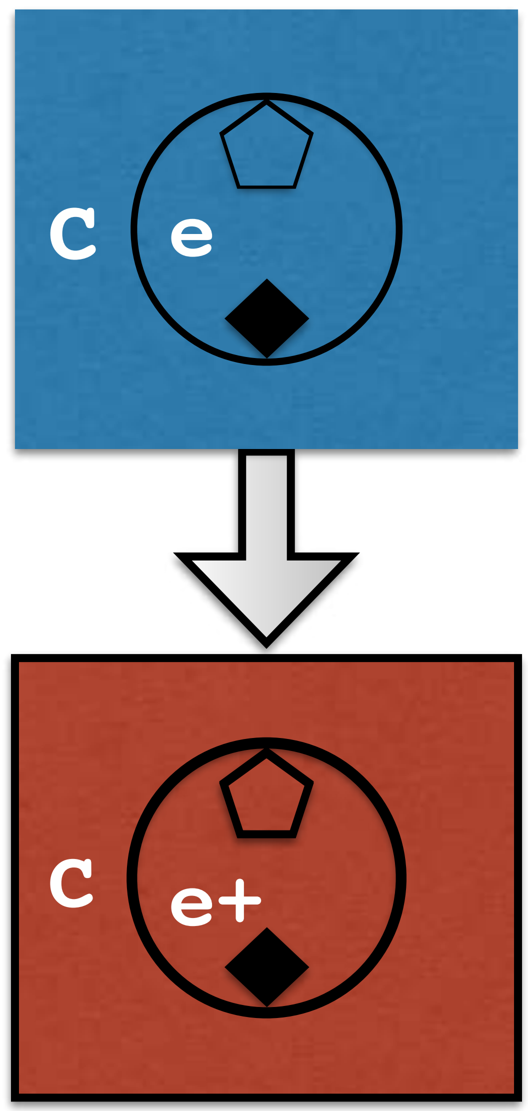


Because compilers



“Correct”

Even if the compiler is proven “correct”…

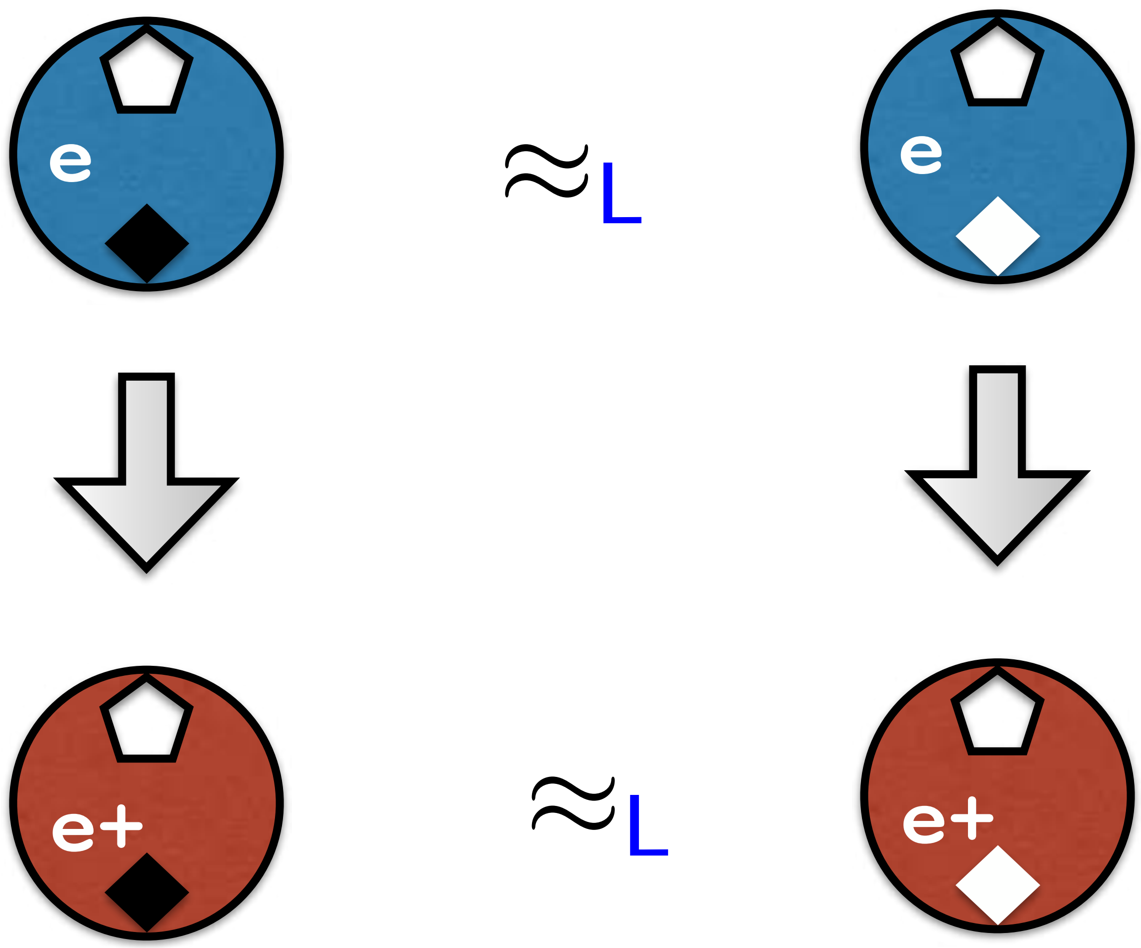


$$\longmapsto^* v$$

$$\longmapsto^* v^+$$

Equivalence Preserving

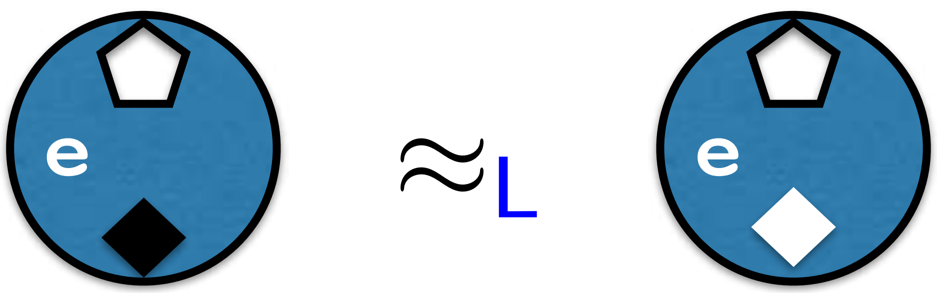
It may not preserve equivalences, e.g., noninterference.



How do we preserve noninterference?

How do we preserve noninterference?

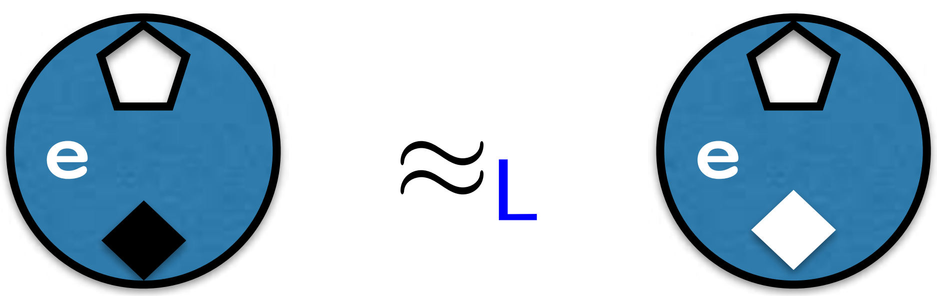
Folklore suggests noninterference can be



captured by parametricity

How do we preserve noninterference?

Folklore suggests noninterference can be



captured by parametricity

[1] Tse & Zdancewic, *Translating Dependency into Parametricity*, ICFP 2004

[2] Shikuma & Igarashi, *Proving Noninterference by a Fully Complete …*, ASIAN 2006

Languages

Source: DCC [1] → Target: System Fω

- Captures dependency analyses
  - e.g. Information-flow security
- STLC + Lattice of monads

- Parametricity
- Type constructors, i.e., higher-order polymorphism

[1] Abadi *et al.*, *The Core Calculus of Dependency,* POPL 1999

Source Language

DCC (Core Calculus of Dependency)

Monad protects
data based on label

$$\Gamma \vdash e_1 : T_\ell\, s_1$$

Source Language

DCC (Core Calculus of Dependency)

$\eta_{\mathsf{H}}\ \mathsf{true} \approx_{\mathsf{L}} \eta_{\mathsf{H}}\ \mathsf{false} : \mathsf{T_H\ bool}$

Monad protects
data based on label

$\Gamma \vdash \mathsf{e_1} : \mathsf{T}_\ell\ \mathsf{s_1}$

Source Language

DCC (Core Calculus of Dependency)

$$\frac{\Gamma \vdash e_1 : T_\ell\, s_1 \qquad \Gamma, x : s_1 \vdash e_2 : s_2}{\Gamma \vdash \mathsf{bind}\ x = e_1\ \mathsf{in}\ e_2 : s_2}$$

Term containing private data → e_1

Continuation using private data → e_2

Source Language

DCC (Core Calculus of Dependency)

Promise that result is protected

$$\frac{\Gamma \vdash e_1 : T_\ell\, s_1 \qquad \Gamma, x : s_1 \vdash e_2 : s_2 \qquad \ell \preceq s_2}{\Gamma \vdash \mathsf{bind}\, x = e_1\, \mathsf{in}\, e_2 : s_2}$$

Source Language

DCC (Core Calculus of Dependency)

Promise that result is protected

For example…

$\mathsf{L} \not\preceq \mathsf{bool}$ $\quad$ $\mathsf{H} \preceq 1$

$\ell \preceq \mathsf{s}_2$

Source Language

DCC (Core Calculus of Dependency)

Promise that result is protected

$\ell \preceq s_2$

For example…

$L \not\preceq \text{bool}$ $\quad$ $H \preceq 1$

$L \preceq T_L\ s$ $\quad$ $H \not\preceq T_L\ s$

Source Language

DCC (Core Calculus of Dependency)

Promise that result is protected

$\ell \preceq s_2$

For example…

$L \not\preceq \text{bool}$ $\quad H \preceq 1$

$L \preceq T_L\ s$ $\quad H \not\preceq T_L\ s$

$L \preceq T_H\ s$ $\quad H \preceq T_H\ s$

Source Language

DCC (Core Calculus of Dependency)

Promise that result is protected

Monad protects data based on label

$$\frac{\Gamma \vdash e_1 : T_\ell\, s_1 \qquad \Gamma, x : s_1 \vdash e_2 : s_2 \qquad \ell \preceq s_2}{\Gamma \vdash \mathsf{bind}\; x = e_1\; \mathsf{in}\; e_2 : s_2}$$

Term containing private data

Continuation using private data

Translation, Briefly

How are types translated?

$$1^+ = \mathbf{1}$$

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

$$(\mathsf{s}_1 \rightarrow \mathsf{s}_2)^+ = \mathsf{s}_1^+ \rightarrow \mathsf{s}_2^+$$

$$(\mathsf{T}_\ell\ \mathsf{s})^+ = \ ?$$

Translation, Briefly

$$\frac{\Gamma \vdash \mathsf{e}_1 : \mathsf{T}_\ell\ \mathsf{s}_1 \qquad \Gamma, \mathsf{x} : \mathsf{s}_1 \vdash \mathsf{e}_2 : \mathsf{s}_2 \qquad \ell \preceq \mathsf{s}_2}{\Gamma \vdash \mathsf{bind}\ \mathsf{x} = \mathsf{e}_1\ \mathsf{in}\ \mathsf{e}_2 : \mathsf{s}_2}$$

Idea: CPS the monad + constrain continuation result

$$(\mathsf{T}_\ell\ \mathsf{s})^+ = \forall \boldsymbol{\beta} :: *. (\mathsf{s}^+ \to \boldsymbol{\beta}) \to \boldsymbol{\beta}$$
$$\text{s.t. } \ell \preceq \boldsymbol{\beta}$$

Translation, Briefly

$$\frac{\Gamma \vdash e_1 : T_\ell\, s_1 \qquad \Gamma, x : s_1 \vdash e_2 : s_2 \qquad \ell \preceq s_2}{\Gamma \vdash \text{bind } x = e_1 \text{ in } e_2 : s_2}$$

$$(T_\ell\, s)^+ = \forall \boldsymbol{\beta}{::}{*}.\, (\llbracket \ell \preceq \boldsymbol{\beta} \rrbracket \times (s^+ \rightarrow \boldsymbol{\beta})) \rightarrow \boldsymbol{\beta}$$

Translation, Briefly

$$\frac{\Gamma \vdash e_1 : T_\ell\ s_1 \qquad \Gamma, x : s_1 \vdash e_2 : s_2 \qquad \ell \preceq s_2}{\Gamma \vdash \mathsf{bind}\ x = e_1\ \mathsf{in}\ e_2 : s_2}$$

$$(T_\ell\ s)^+ = \forall \beta{::}{*}.\ ([\![\ell \preceq \beta]\!] \times (s^+ \to \beta)) \to \beta$$

$$[\![\ell \preceq s^+]\!] = (\alpha_\preceq\ \ \alpha_\ell\ \ s^+)$$

Translation, Briefly

$$\frac{\Gamma \vdash e_1 : T_\ell\, s_1 \qquad \Gamma, x : s_1 \vdash e_2 : s_2 \qquad \ell \preceq s_2}{\Gamma \vdash \mathsf{bind}\; x = e_1 \;\mathsf{in}\; e_2 : s_2}$$

$$(T_\ell\, s)^+ = \forall \beta{::}{*}.\,((\alpha_{\preceq}\;\; \alpha_\ell\;\; \beta) \times (s \to \beta)) \to \beta$$

$$
\begin{array}{l}
\mathbf{data}\;\; (\alpha_{\preceq}\;\; \alpha_\ell\;\; s^+)\;\; \mathbf{where} \\
\quad \mathtt{Unit\_P} :: (\alpha_{\preceq}\;\; \alpha_\ell\;\; 1) \\
\quad \ldots \\
\quad \mathtt{Monad\_P} :: [\![\ell \sqsubseteq \ell']\!] \to (\alpha_{\preceq}\;\; \alpha_\ell\;\; (T_{\ell'}\, s)^+)
\end{array}
$$

Translation Summary

$$1^+ = \mathbf{1}$$

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

$$(\mathsf{s}_1 \to \mathsf{s}_2)^+ = \mathsf{s}_1^+ \to \mathsf{s}_2^+$$

$$(\mathsf{T}_\ell\ \mathsf{s})^+ = \forall \boldsymbol{\beta}{::}{*}.\,((\boldsymbol{\alpha}_{\preceq}\ \ \boldsymbol{\alpha}_\ell\ \ \boldsymbol{\beta}) \times (\mathsf{s} \to \boldsymbol{\beta})) \to \boldsymbol{\beta}$$

Translation Summary

$$1^+ = \mathbf{1}$$

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

$$(\mathsf{s}_1 \to \mathsf{s}_2)^+ = \mathsf{s}_1^+ \to \mathsf{s}_2^+$$

$$(\mathsf{T}_\ell\ \mathsf{s})^+ = \forall \boldsymbol{\beta}{::}{*}.\ ((\boldsymbol{\alpha}_{\preceq}\ \ \boldsymbol{\alpha}_\ell\ \ \boldsymbol{\beta}) \times (\mathsf{s} \to \boldsymbol{\beta})) \to \boldsymbol{\beta}$$

Invariants:

$\vdash \mathsf{s}$ $\qquad\qquad$ $\boldsymbol{\alpha}_{\preceq}, \boldsymbol{\alpha}_\ell, \cdots \vdash \mathsf{s}^+$

Proving Noninterference Preservation

Proving Equivalence Preservation

Equiv. Preservation is Hard

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x}{:}s^{+}.e^{+} \approx \boldsymbol{\lambda x}{:}s^{+}.e'^{+}$$

Equiv. Preservation is Hard

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x}{:}s^{+}.e^{+} \approx \boldsymbol{\lambda x}{:}s^{+}.e'^{+}$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

Equiv. Preservation is Hard

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x{:}}s^{+}\boldsymbol{.}e^{+} \approx \boldsymbol{\lambda x{:}}s^{+}\boldsymbol{.}e'^{+}$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

Want to proceed as follows:

- By assumption, $\lambda x : s.\, e \approx \lambda x : s.\, e'$

Equiv. Preservation is Hard

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x}\!:\!s^{+}.e^{+} \approx \boldsymbol{\lambda x}\!:\!s^{+}.e'^{+}$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

Want to proceed as follows:

- By assumption, $\lambda x : s.\, e \approx \lambda x : s.\, e'$
- Hence, $e[\mathbf{m_1}/x] \approx e'[\mathbf{m_2}/x]$

Equiv. Preservation is Hard

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x}{:}s^{+}.e^{+} \approx \boldsymbol{\lambda x}{:}s^{+}.e'^{+}$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

Want to proceed as follows:

- By assumption, $\lambda x : s.\, e \approx \lambda x : s.\, e'$
- Hence, $e[\mathbf{m_1}/x] \approx e'[\mathbf{m_2}/x]$
- By induction

Equivalence Reflection?

How do we say that since $\mathbf{m_1} \approx \mathbf{m_2} : \mathsf{s}^+$

$$\begin{array}{ccc} \mathbf{m_1} & & \mathbf{m_2} \\ \wr & & \wr \\ \mathsf{e}_1 & & \mathsf{e}_2 : \mathsf{s} \end{array}$$

there must exist $\mathsf{e}_1 \quad \mathsf{e}_2 : \mathsf{s}$

Equivalence Reflection?

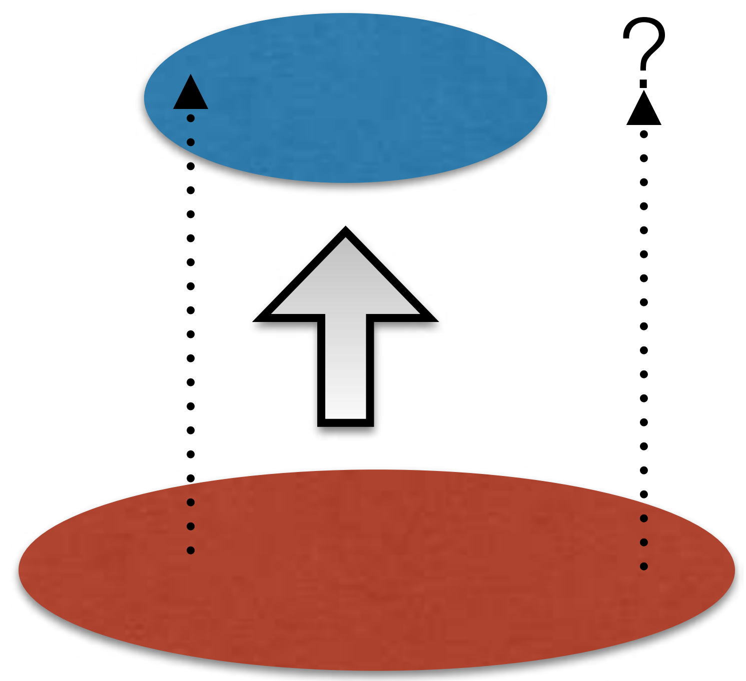
How do we say that since $\mathbf{m_1} \approx \mathbf{m_2} : \mathsf{s}^+$

$$\wr \wr \qquad \wr \wr$$

there must exist $\mathsf{e}_1 \approx \mathsf{e}_2 : \mathsf{s}$

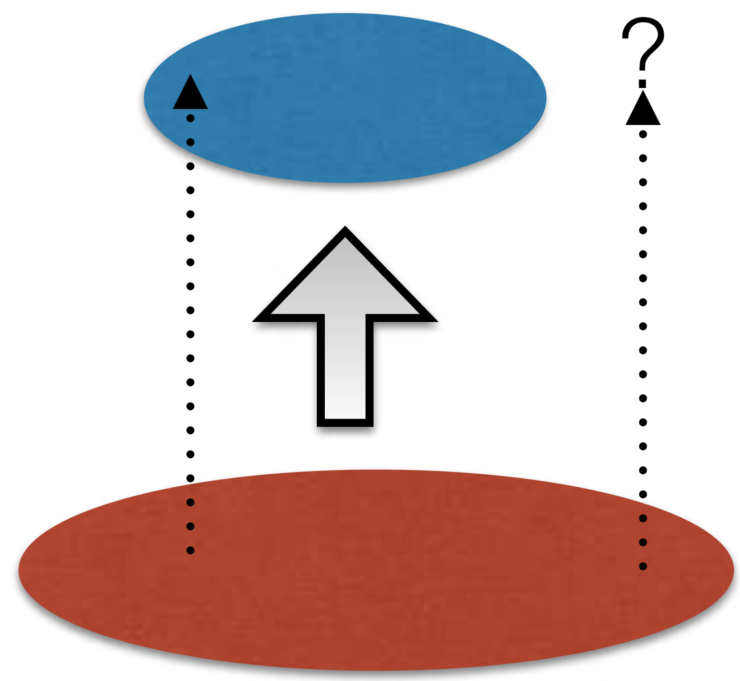
Back-translation

How can we possibly back-translate a *more expressive* target language?



Enrich Source?

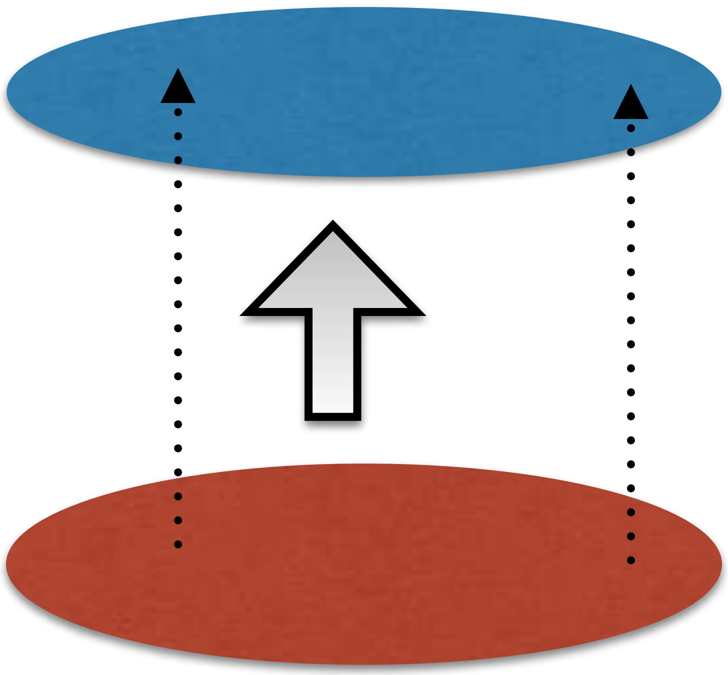
How can we possibly back-translate
a *more expressive* target language?



We could enrich the source

Enrich Source?

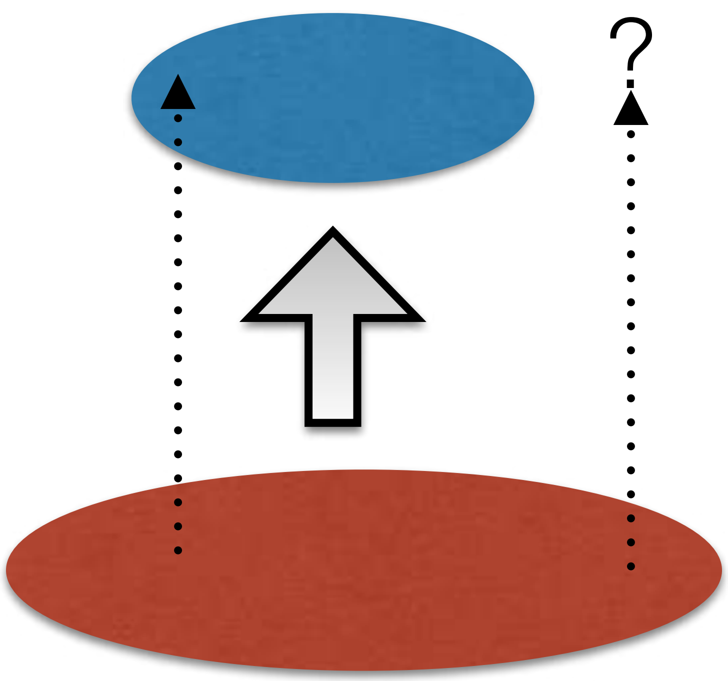
How can we possibly back-translate
a *more expressive* target language?



We could enrich the source

Impoverish Target?

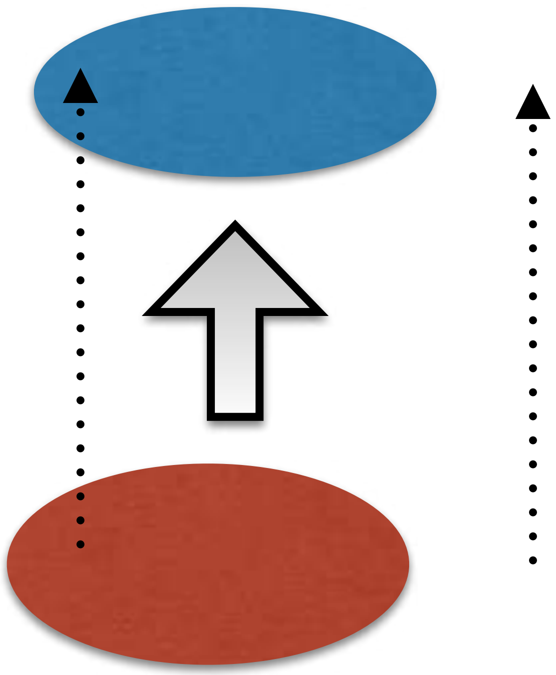
How can we possibly back-translate
a *more expressive* target language?



Or impoverish the target

Impoverish Target?

How can we possibly back-translate
a *more expressive* target language?

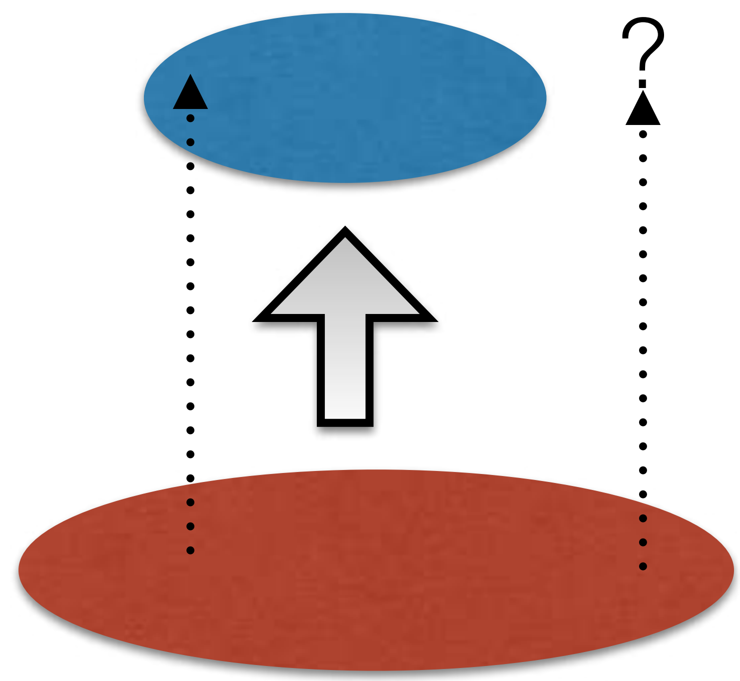


Or impoverish the target

None of the above

How can we possibly back-translate
a *more expressive* target language?

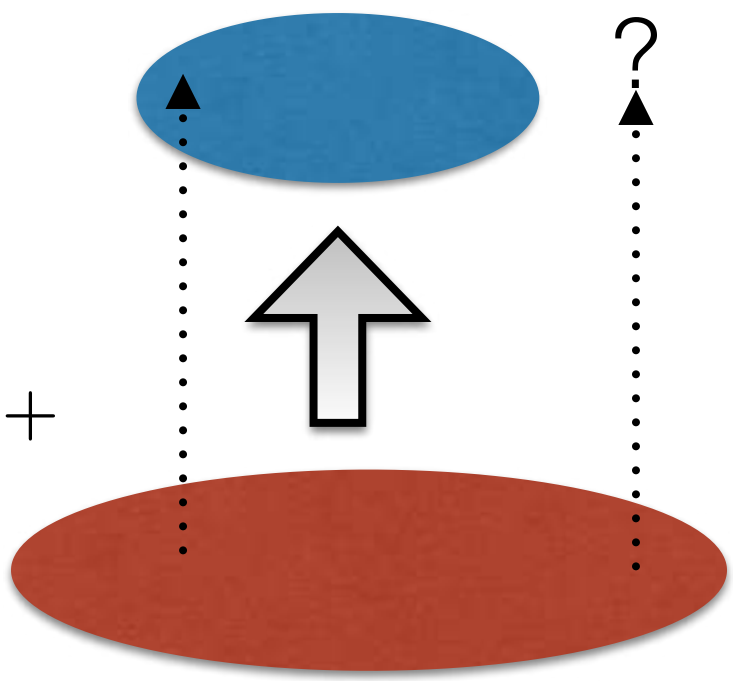
Neither is satisfying



Be clever

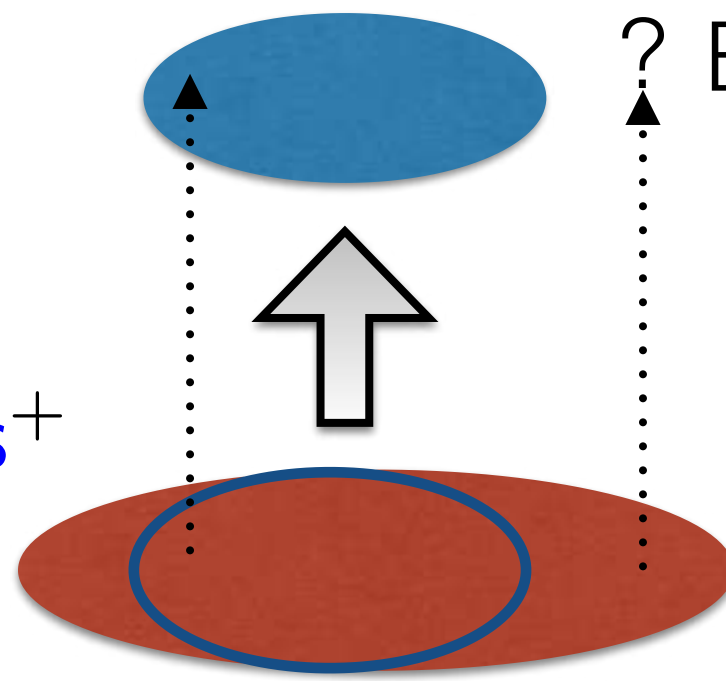
How can we possibly back-translate
a *more expressive* target language?

Recall:
Assume $\mathbf{m_1} \approx \mathbf{m_2} : \mathsf{s}^+$



Be clever

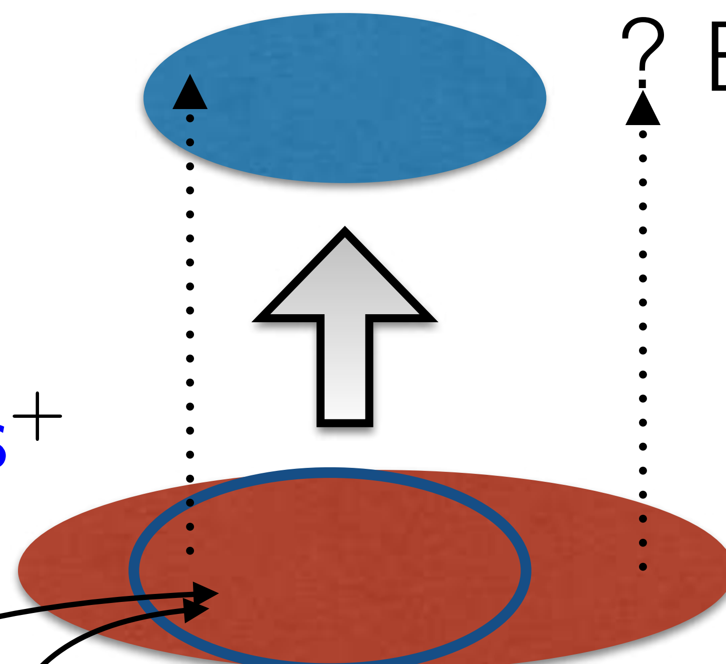
How can we possibly back-translate
a *more expressive* target language?



Recall:
Assume $\mathbf{m_1} \approx \mathbf{m_2} : \mathrm{s}^+$

Be clever

How can we possibly back-translate
a *more expressive* target language?



? By using types.

Recall:
Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

e.g.

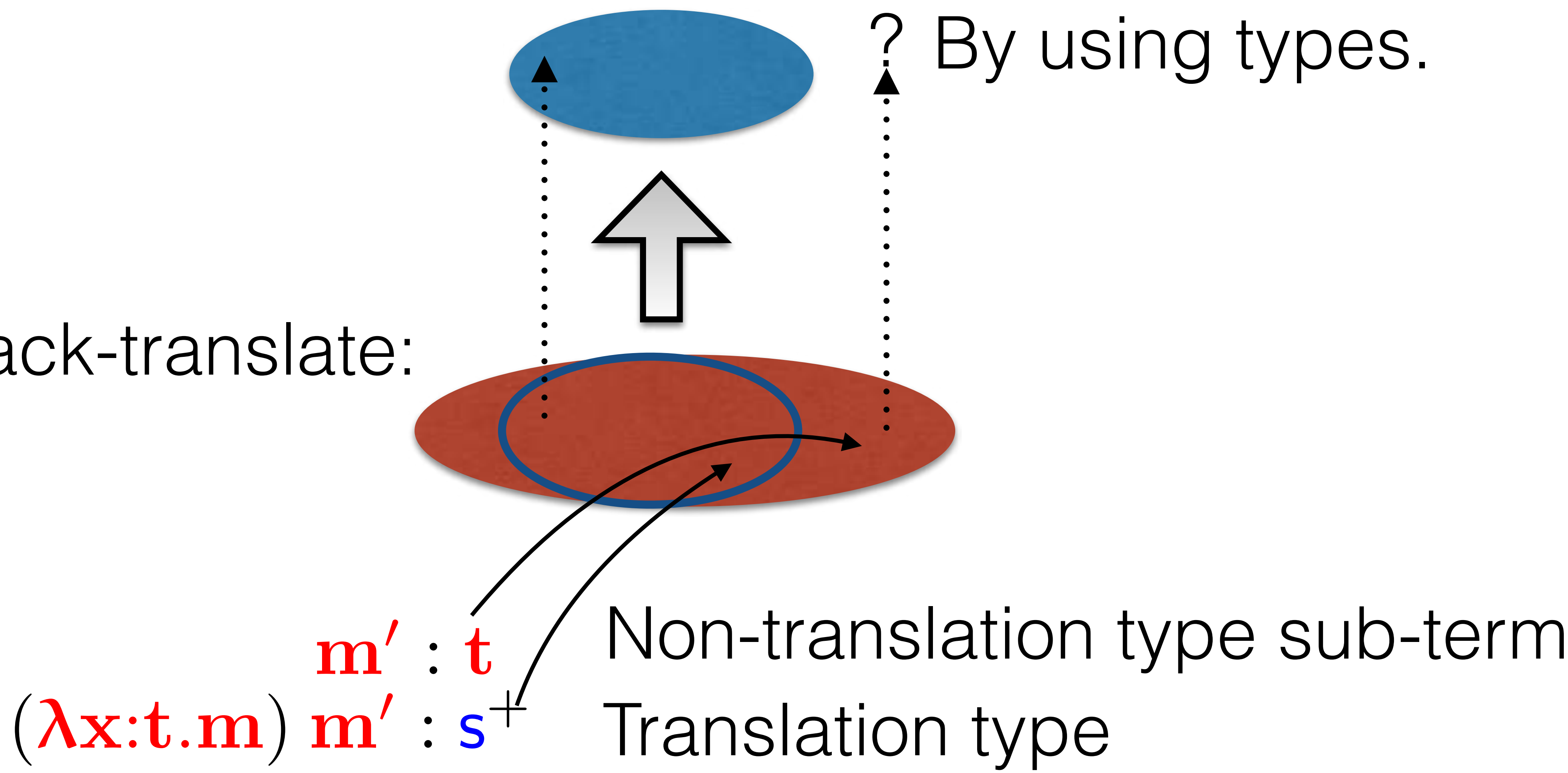
$$\mathbf{x} : s_1^{+} \vdash \mathbf{m} : s_2^{+} \quad \cdots\cdots\blacktriangleright \quad x : s_1 \vdash e : s_2$$

$$\boldsymbol{\lambda}\mathbf{x}{:}s_1^{+}.\mathbf{m} : s_1^{+} \rightarrow s_2^{+} \quad \cdots\cdots\blacktriangleright \quad \lambda x : s_1 . e : s_1 \rightarrow s_2$$

Be clever

How can we possibly back-translate
a *more expressive* target language?

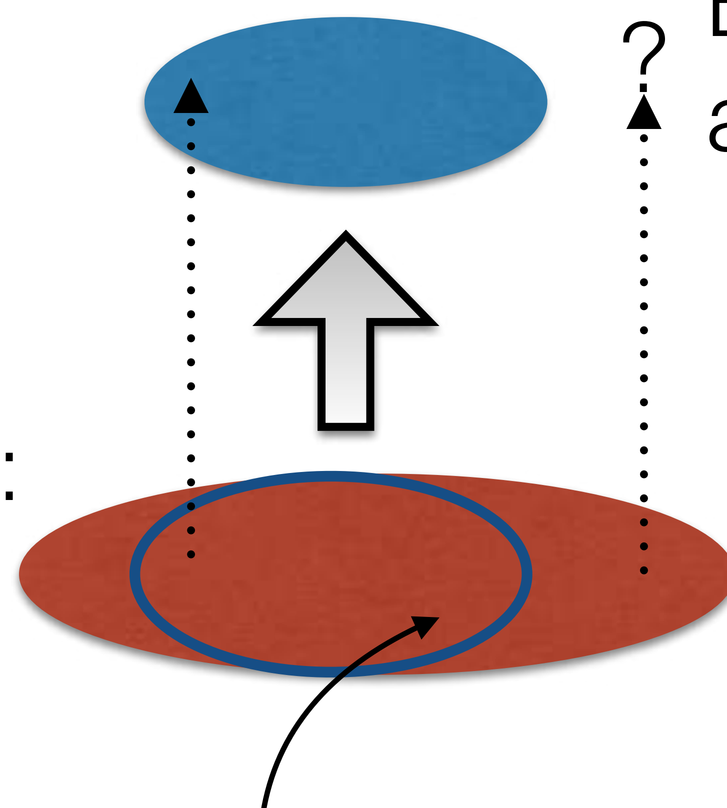
Can we back-translate:



Be cleverer

How can we possibly back-translate
a *more expressive* target language?

By using types
and partial evaluation.



Can we back-translate:

Not by induction

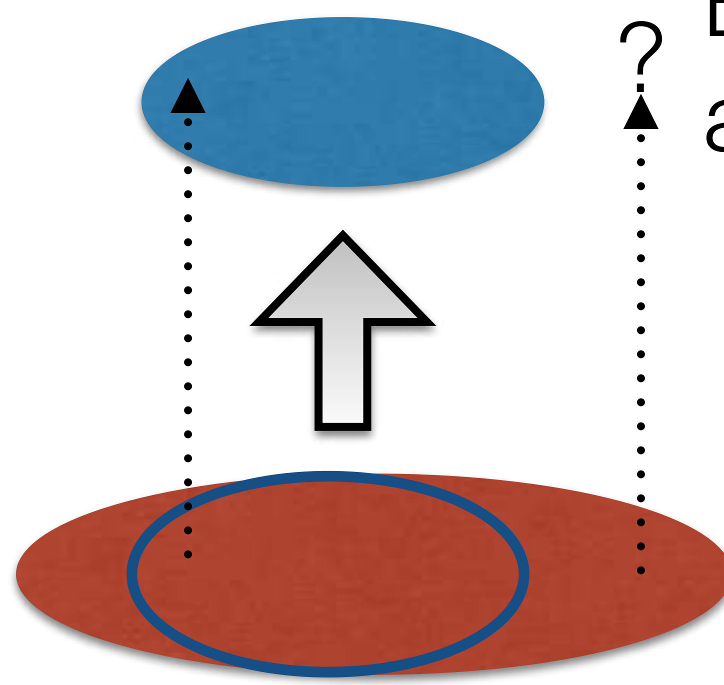
$(\lambda x{:}t.m)\ m' \longmapsto m'' : s^+$

$m' : t$ is no more

Be clevererer

How can we possibly back-translate
a *more expressive* target language?

By using types
and partial evaluation.



Not by induction

Therefore, must prove all terms are back-translatable.

Proof of Equiv. Preservation

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x{:}}s^+\boldsymbol{.}e^+ \approx \boldsymbol{\lambda x{:}}s^+\boldsymbol{.}e'^+$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^+$

Show $e^+[\mathbf{m_1}/\mathbf{x}] \approx e'^+[\mathbf{m_2}/\mathbf{x}]$

How the proof proceeds:

Proof of Equiv. Preservation

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x}{:}s^{+}.e^{+} \approx \boldsymbol{\lambda x}{:}s^{+}.e'^{+}$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

How the proof proceeds:

- Back-translate $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$ to $e_1 \approx e_2 : s$

Proof of Equiv. Preservation

To *show*

$$\lambda x:s.\,e \approx \lambda x:s.\,e'$$

implies

$$\boldsymbol{\lambda x}{:}s^{+}.e^{+} \approx \boldsymbol{\lambda x}{:}s^{+}.e'^{+}$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

How the proof proceeds:

- Back-translate $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$ to $e_1 \approx e_2 : s$
- By assumption, $\lambda x:s.\,e \approx \lambda x:s.\,e'$

Proof of Equiv. Preservation

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x}{:}s^{+}.e^{+} \approx \boldsymbol{\lambda x}{:}s^{+}.e'^{+}$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

How the proof proceeds:

- Back-translate $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$ to $e_1 \approx e_2 : s$
- By assumption, $\lambda x : s.\, e \approx \lambda x : s.\, e'$
- Hence, $e[e_1/x] \approx e'[e_2/x]$

Proof of Equiv. Preservation

To *show*

$$\lambda x : s.\, e \approx \lambda x : s.\, e'$$

implies

$$\boldsymbol{\lambda x}{:}s^{+}.e^{+} \approx \boldsymbol{\lambda x}{:}s^{+}.e'^{+}$$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

How the proof proceeds:

- Back-translate $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$ to $e_1 \approx e_2 : s$
- By assumption, $\lambda x : s.\, e \approx \lambda x : s.\, e'$
- Hence, $e[e_1/x] \approx e'[e_2/x]$
- By induction. QED.

Proof of Equiv. Preservation

To *show*

$\lambda x : s.\, e \approx \lambda x : s.\, e'$

implies

$\lambda x{:}s^+.e^+ \approx \lambda x$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^+$

Show $e^+[\mathbf{m_1}/\mathbf{x}] \approx e'^+[\mathbf{m_2}/\mathbf{x}]$

roceeds:

$_1 \approx \mathbf{m_2} : s^+$

$: s.\, e \approx \lambda x : s.\, e'$

$e'[e_2/x]$

.

EXCEPT

Proof of Equiv. Preservation

To *show*

$\lambda x : s.\, e \approx \lambda x : s.\, e'$

implies

$\lambda \mathbf{x} : s^{+} .\, e^{+} \approx \lambda \mathbf{x}$

Assume $\mathbf{m_1} \approx \mathbf{m_2} : s^{+}$

Show $e^{+}[\mathbf{m_1}/\mathbf{x}] \approx e'^{+}[\mathbf{m_2}/\mathbf{x}]$

oes:

$\mathbf{{}_1} \approx \mathbf{m_2} : s^{+}$

$x : s.\, e \approx \lambda x : s.\, e'$

$e'[e_2/x]$

.

EXCEPT

If you don't understand logical relations, you can stop listening for the next 4 slides.

“Open” Logical Relations

Typically, logical relations are defined on closed terms/types.

Again recall: Assume $\mathbf{m_1} \approx \mathbf{m_2} : \mathsf{s}^+$

And: $\boldsymbol{\alpha}_{\preceq}, \boldsymbol{\alpha}_{\ell}, \cdots \vdash \mathsf{s}^+$

But translation types are only well-formed when open.

“Open” Logical Relations

Again recall: Assume $\mathbf{m_1} \approx \mathbf{m_2} : \mathsf{s}^+$

And: $\boldsymbol{\alpha}_{\preceq}, \boldsymbol{\alpha}_{\ell}, \cdots \vdash \mathsf{s}^+$

i.e., to even *state* this assumption, the logical relation *must* leave these type variables open.

"Open" Logical Relations

~~$\mathbf{m_1} \approx \mathbf{m_2} : \mathsf{s}^{+}$~~

building on [1], we define $\mathbf{m_1} \approx^{\mathbf{\Sigma}} \mathbf{m_2} : \mathsf{s}^{+}$

$\mathbf{\Sigma} =$

```
data (α_⪯  α_ℓ  s⁺) where
  Unit_P :: (α_⪯  α_ℓ  1)
  ...
  Monad_P :: (α_⪯  α_ℓ  (T_ℓ s)⁺)
```

[1] Zhao *et al.*, *Relational Parametricity for Linear System F*, APLAS 2010

QED! (ish)

$$\sout{\mathbf{m_1} \approx \mathbf{m_2} : \mathsf{s}^+}$$

$$\mathbf{m_1} \approx^{\boldsymbol{\Sigma}} \mathbf{m_2} : \mathsf{s}^+$$

$$\boldsymbol{\Sigma} =$$

```
data (α_⪯ α_ℓ s⁺) where
  Unit_P :: (α_⪯ α_ℓ 1)
  ...
  Monad_P :: (α_⪯ α_ℓ (T_ℓ s)⁺)
```

$$[\![\boldsymbol{\Sigma}]\!]_{\mathsf{L}} = ?$$

Conclusion

- Language-based reasoning requires better compilers
- We have developed techniques for such compilers (specifically, for noninterference preservation)

https://www.williamjbowman.com/papers#niforfree