The Operational Semantics of Multi-Language Systems

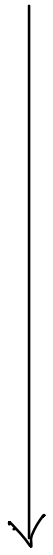
Outline

0. Motivation: How do we interop?

1. Approaching semantics (Matthews & Findler '07, '09)

2. A surprising application (Ahmed & Blume '11)

3. A major stress test (Patterson et al. '17)



"Distance" between interop-ing languages

How do we interop?

- Languages are different (even under the hood)
  - > Calling conventions
  - > Data representations
  - > Memory layouts, strategies

- So how do we use them together?
  - > "Protocols"
  - > Foreign interfaces
  - > Common runtimes

- What are the drawbacks?
  - > Glue code
  - > Lossy translations (coarse-grained)
  - > Broken abstractions (unsafe {...})
  - > Bad tooling

PROBLEM: HOW DO WE REASON ABOUT INTEROP?

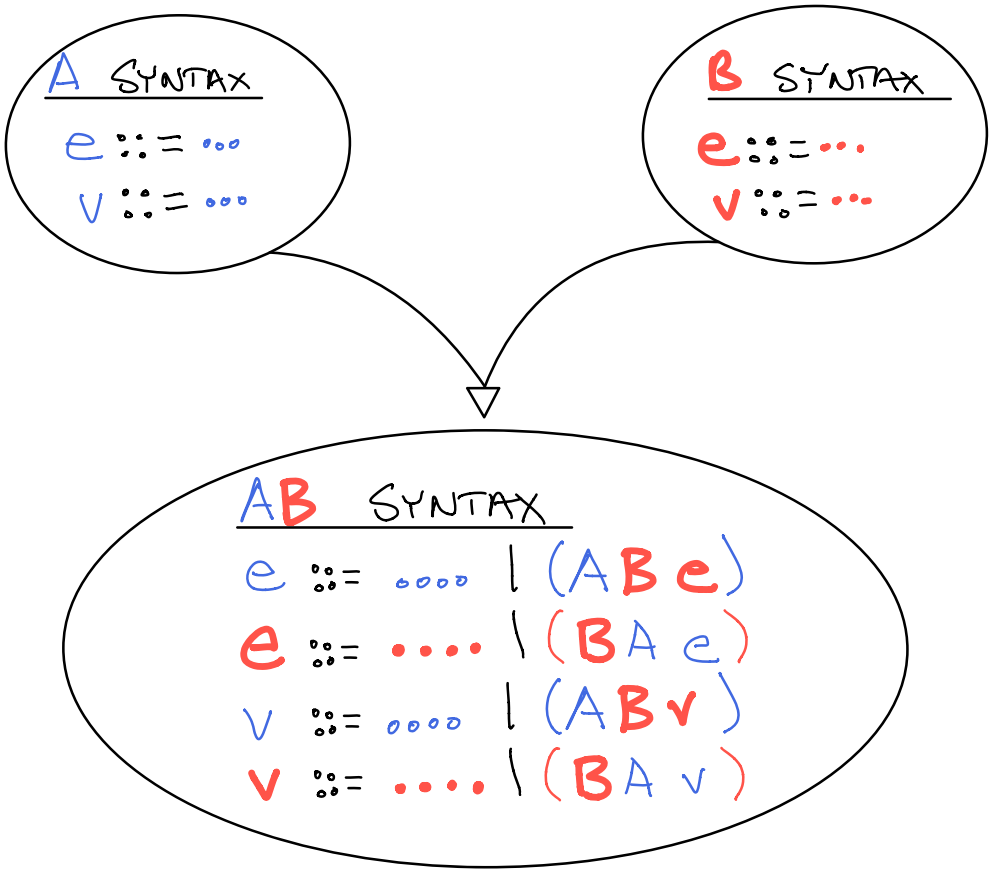
¯_(ツ)_/¯

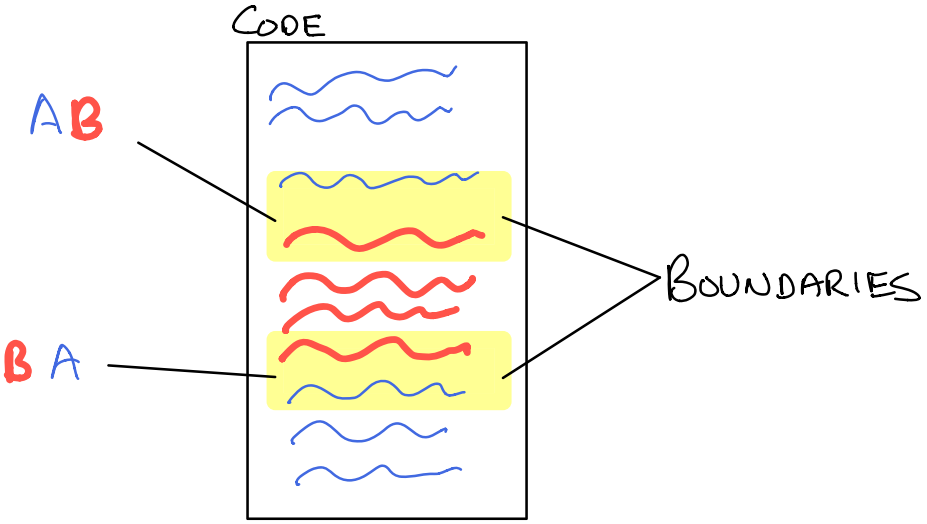
ONE APPROACH:

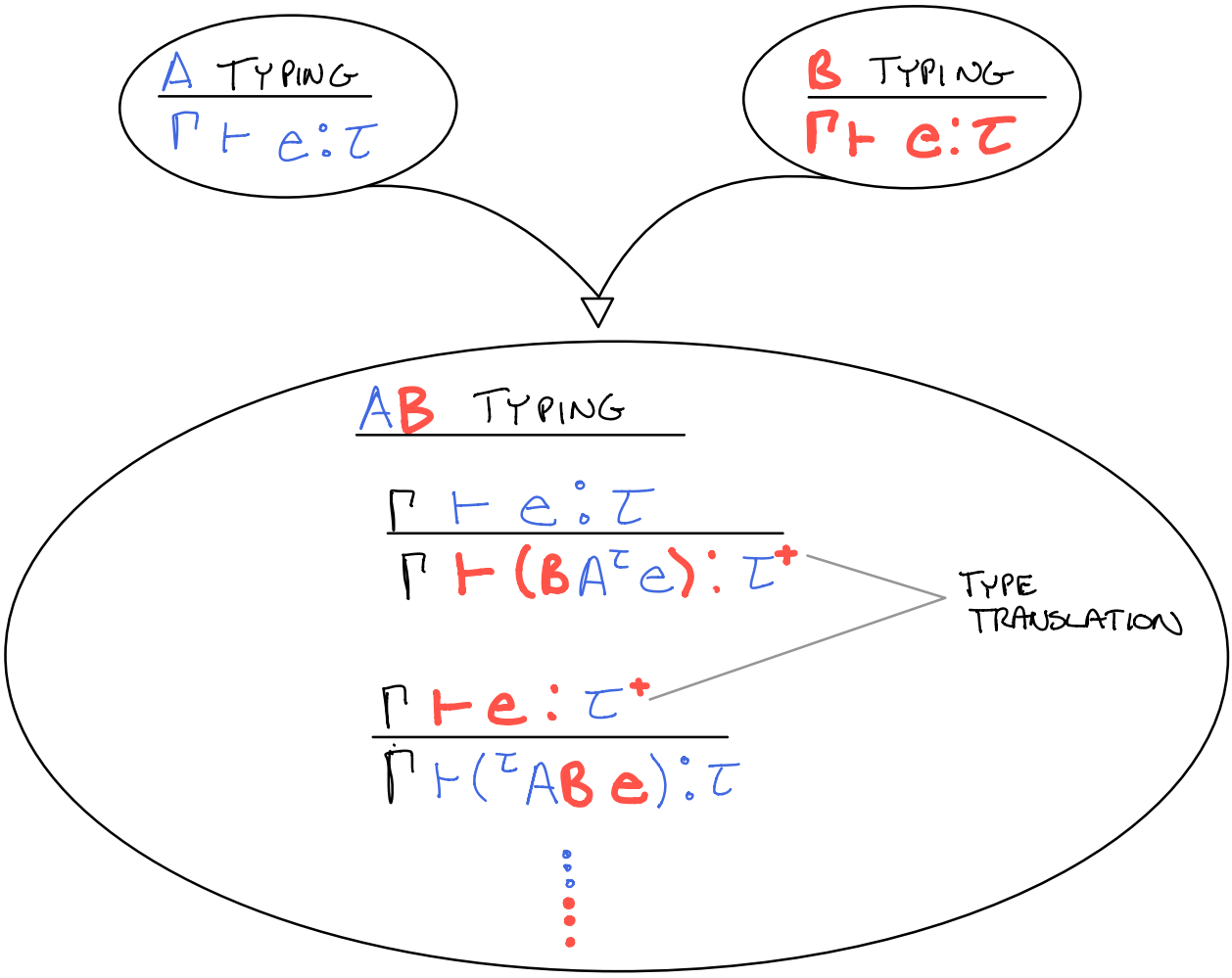
"OPERATIONAL SEMANTICS FOR MULTI-LANGUAGE PROGRAMS"
— MATTHEWS & FINDLER '07, '09

MF07: The Big Picture

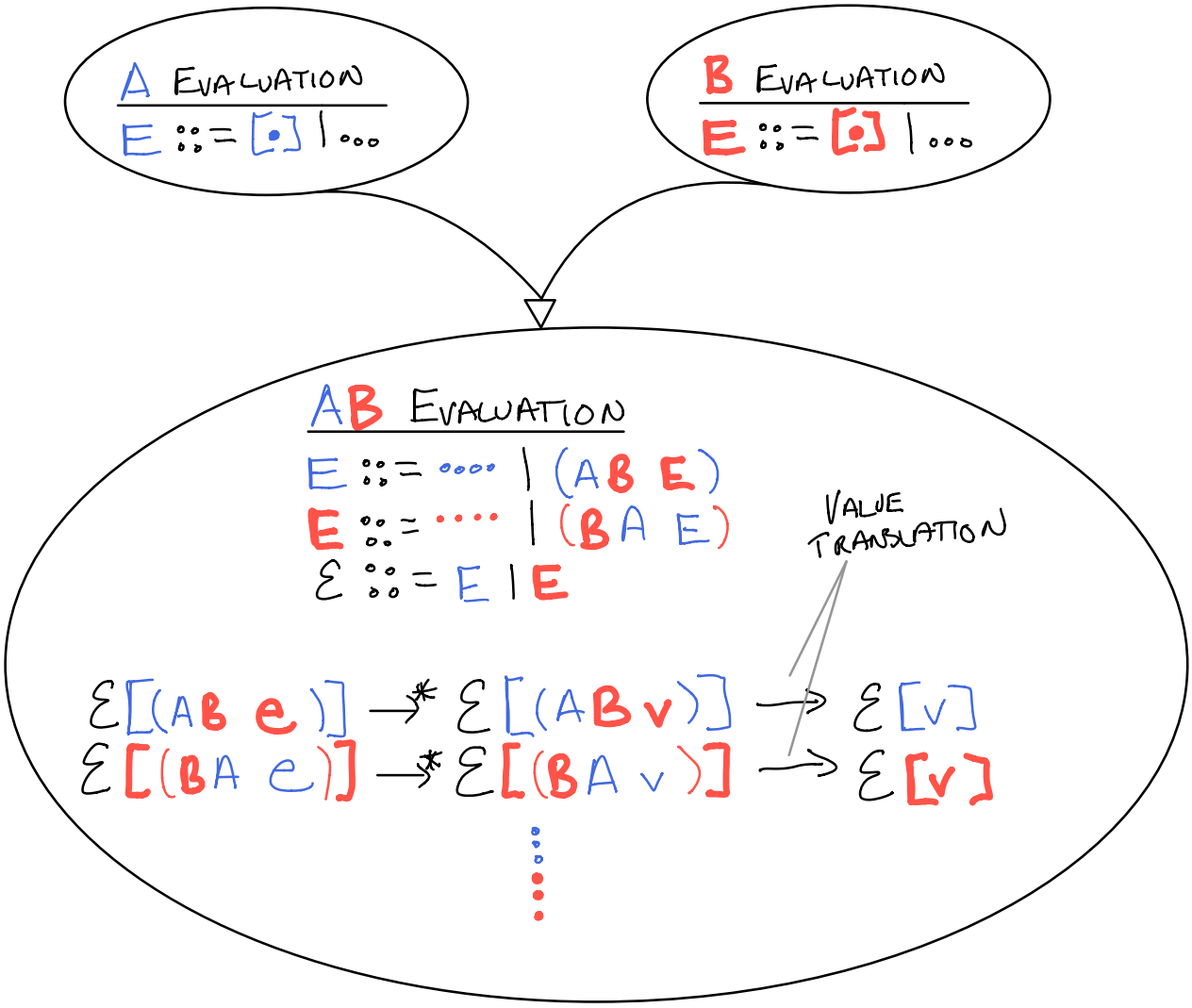
To interoperate languages A + B,



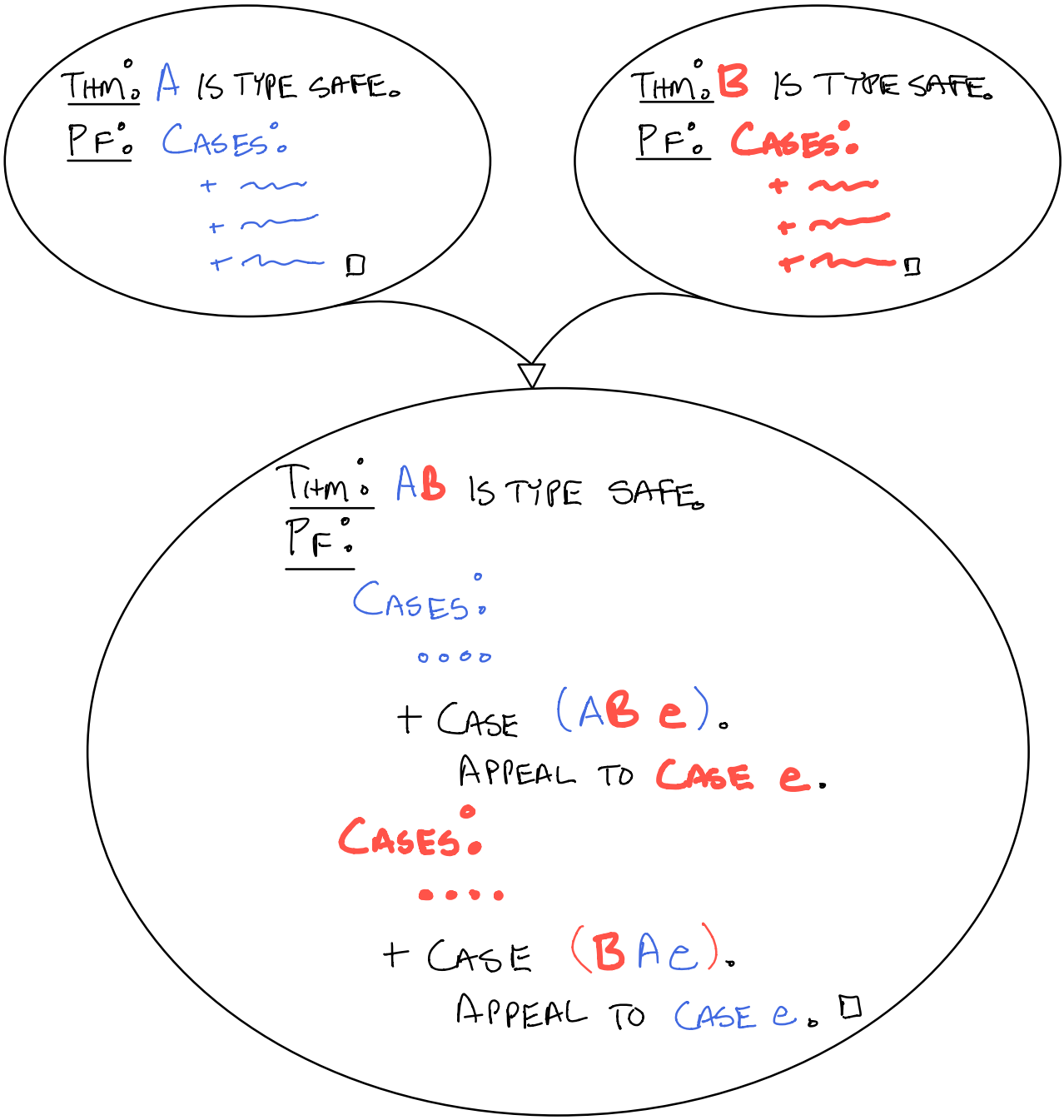




BOUNDARIES ≈ "CROSS-LANGUAGE CASTS"



"Evaluate under boundary, then translate"



- Reuse/repurpose existing meta-theoretic tools
  - > Subject reduction, logical relations, equivalence, etc
  - > Disclaimer: Not always straightforward!

MF07: ML-SCHEME (MORE LIKE STLC − λ)

$\tau^+ \triangleq \mathrm{TST}$ ("THE SCHEME TYPE")

$$\frac{\Gamma \vdash e : \mathrm{TST}}{\Gamma \vdash (?_{\mathrm{MS}}\, e) : \tau}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash (\mathrm{SM}^{\tau}\, e) : \mathrm{TST}}$$

RECALL: GRADUAL TYPING (OLEK)

- MF07 WIP TALK
- TOBIN-HOCHSTADT & FELLEISEN '06
- MF07 PUBLISHED
- MF09 PUBLISHED

WHAT HAPPENS OPERATIONALLY?

↳ MANY OPTIONS!

MF07: The Lump Embedding

A foreign value is a blackbox **lump**

$\mathcal{E}[({}^{L}M\,S\,v)]$ M can't touch v

$\mathcal{E}[(S\,M^{L}\,v)]$ S can't touch v

Unless it is itself a boundary

$\mathcal{E}[({}^{\tau}M\,S\,(S\,M^{\tau}\,v))] \longrightarrow \mathcal{E}[v]$

$\mathcal{E}[(S\,M^{L}\,({}^{L}M\,S\,v))] \longrightarrow \mathcal{E}[v]$

In which case boundaries may **cancel**

MF07. Foreign Apply w/ Lumps

```
fapp : L → L → L
fapp f x = ⌞M S {(S M⌞ f) (S M⌞ x)}
```

"f" "x"

Ex.

```
fapp (⌞M S add1) (⌞M S 42)
```

(β)

```
↳ ⌞M S {(S M⌞ (⌞M S add1))
        (S M⌞ (⌞M S 42))}
```

(Cancel)

```
↳ ⌞M S {add1
        (S M⌞ (⌞M S 42))}
```

(Cancel)

```
↳ ⌞M S (add1 42)
```

(β)

```
↳ ⌞M S 43
```

Actually can get really far (theoretically) just with this!

MF07: The Natural Embedding

Can we use foreign values natively?

$$\begin{array}{rl} & \mathsf{S}\,\mathsf{M}^{\mathsf{N}}\ 42 \longrightarrow 42 \\ \mathsf{N}\to\mathsf{N}\quad & \mathsf{M}\ \mathsf{S}\ \mathsf{add1} \longrightarrow \text{"}\mathsf{add1}\text{"} \end{array}$$

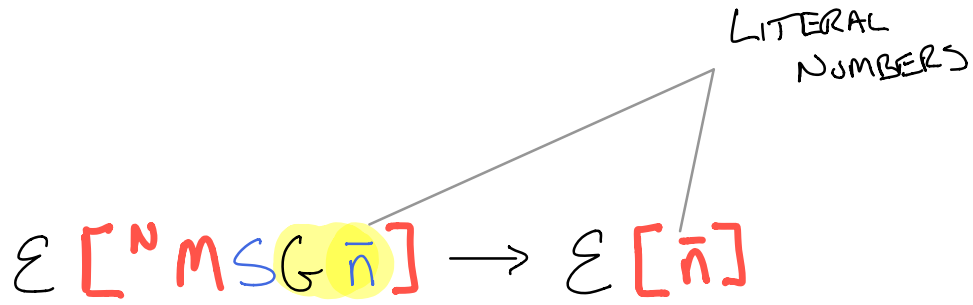
First Attempt: "Eta Expansion + Boundaries"

$$\begin{array}{l} \mathcal{E}\,[\tau_1 \to \tau_2\ \mathsf{M}\ \mathsf{S}\ f] \\ \hookrightarrow \mathcal{E}\,[\lambda(x{:}\tau_1).\ ^{\tau_2}\mathsf{M}\ \mathsf{S}\ \{f\ (\underbrace{\mathsf{S}\,\mathsf{M}^{\tau_1}\,x}_{\text{"}x\text{"}})\}] \end{array}$$

Problem: Too trusting! Where are the checks?

Recall: Higher-Order Contracts (Cameron)
Gradual Typing (Olek)

MF07. Guards - First Order



Otherwise, ℰ ⟦ⁿ MSG v ⟧ → Error!

MF07: Guards - Higher Order

$\mathcal{E}[\![\tau_1 \to \tau_2\ M\ S\ G\ \lambda x.e]\!]$ — First-order tag check

↳ $\mathcal{E}[\![\lambda (x:\tau_1).\ (\tau_2\ M\ S\ G\ \{(\lambda x.e)\ (G\ S\ M^{\tau_1}\ x)\})]\!]$

Compute v

Guard ensures $v \rightsquigarrow \mathbf{v}$ behaves like τ_2

Otherwise, $\mathcal{E}[\![\tau_1 \to \tau_2\ M\ S\ G\ v]\!] \to$ Error!

Recall: Higher-Order Contracts (Cameron)

MF07: GUARDS IN ACTION

- $\mathcal{E}[\,^{N\to N}\mathrm{MSG}(\lambda x.\#t)]$

  $\hookrightarrow \mathcal{E}[\lambda(x:N).\ ^{N}\mathrm{MSG}\{(\lambda x.\#t)\,(GSM^{N}\,x)\}]$

- (f 42)

  (β) $\hookrightarrow\ ^{N}\mathrm{MSG}\{(\lambda x.\#t)\,(GSM^{N}\,42)\}$ — As above

  (GSM^{N}) $\hookrightarrow\ ^{N}\mathrm{MSG}\{(\lambda x.\#t)\ 42\}$

  (β) $\hookrightarrow\ ^{N}\mathrm{MSG}\ \#t$

  $(^{N}\mathrm{MSG})$ $\hookrightarrow$ ERROR!

MF07: Other Conversion Strategies

- So far, type-directed (e.g., $^{\tau}MS$)
- Conversion strategy can be decoupled from types!

Ex: Handle S exceptions from M

$$\mathcal{E}[{}^{N!}MS\ \bar{n}] \longrightarrow \mathcal{E}[\bar{n}]$$
$$\mathcal{E}[{}^{N!}MS\ \text{ERR}!] \longrightarrow \mathcal{E}[0]$$

0 — sentinel

More generally, define K, $\lfloor\cdot\rfloor : K \to \tau$ — "conversion strategy"

$$K ::= \tau \mid N! \mid \ldots$$
$$\lfloor N! \rfloor = N$$
$$\lfloor N \rfloor = N$$
$$\vdots$$

$$\frac{\Gamma \vdash e : T_{ST}}{\Gamma \vdash ({}^{K}MS\ e) : \lfloor K \rfloor}$$

$$\frac{\Gamma \vdash e : \lfloor K \rfloor}{\Gamma \vdash (SM^{K}\ e) : T_{ST}}$$

MF07: RECAP

BOUNDARY TERMS

$\mathcal{E}[(\lceil A \rceil_B\ e)] \rightarrow^* \mathcal{E}[(\lceil A \rceil_B\ v)] \rightarrow \mathcal{E}[v]$

- INFLUENCED GRADUAL TYPING

- STRONG CONNECTION WITH CONTRACTS:
  - DECOUPLE GUARDS FROM BOUNDARIES
    - ↳ REPLACE W/ CONTRACTS (SHOWN IN PAPER)
  - C.F. GRAY ET AL. '05

- LOTS OF CHOICES FOR BOUNDARIES
  - WHAT/HOW DO WE TRANSLATE?
  - WHAT/WHEN DO WE CHECK?

- CASE STUDY: ML-SCHEME (STLC-λ)
  - TYPING: STATIC-DYNAMIC
  - TWO HIGH-LEVEL, SURFACE LANGUAGES
  - BOTH USE DIRECT-STYLE CONTROL FLOW

"An Equivalence Preserving
CPS Translation
via Multi-Language Semantics"
— Ahmed + Blume '11

AB11: CPS

"AN EQUIVALENCE-PRESERVING CPS TRANSLATION VIA MULTI-LANGUAGE SEMANTICS"

$$(\tau_1 \to \tau_2)^+ \triangleq \tau_1^+ \times (\tau_2^+ \to \text{Ans}) \to \text{Ans}$$

Ex:

$$\lambda(f : \mathbb{1} \to \mathbb{N},\ g : \mathbb{1} \to \mathbb{N}).\ f() + g()$$

$\mathbb{I} \cdot \mathbb{I}_{CPS}$ ↓

$$\lambda(f, g : \mathbb{1} \times (\mathbb{N} \to \text{Ans}) \to \text{Ans},\ k : \mathbb{N} \to \text{Ans}).$$
$$(f()\ (\lambda(x : \mathbb{N}).\ (g()\ (\lambda(y : \mathbb{N}).\ \text{let } z = x + y \text{ in } k\ z))))$$

CONTINUATION

"RETURN" WITH CONTINUATION APPLIED TO VALUES

AB11: Equivalence Preservation

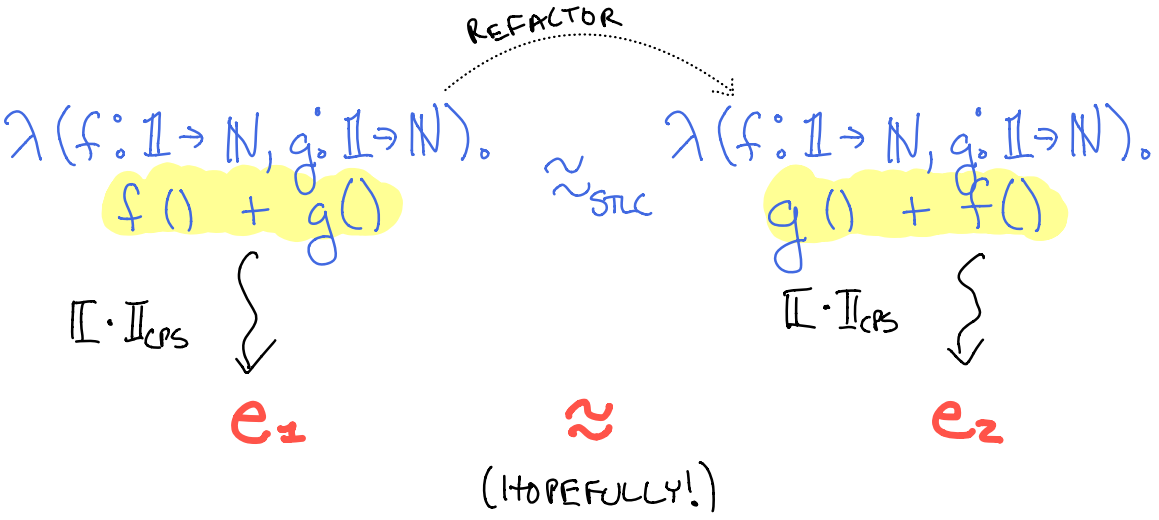
"An Equivalence-Preserving CPS Translation via Multi-Language Semantics"

$$e_1 \approx e_2 \Rightarrow [\![e_1]\!]^S_T \approx [\![e_2]\!]^S_T$$

Compiled terms

Why? Programmer can safely reason in S without knowing T, $[\![\cdot]\!]^S_T$

Ex: Refactoring e shouldn't break $[\![e]\!]^S_T$



AB11: The Problem

$e_1 \triangleq \lambda(f, g).\ f() + g()$

$\llbracket \cdot \rrbracket_{CPS}$

$\lambda(f, g, k).\ (f\ ()\ (\lambda x.\ (g\ ()\ (\lambda y.\ \ldots))))$

$\sim_{STLC}$

$\not\simeq$

$e_2 \triangleq \lambda(f, g).\ g() + f()$

$\llbracket \cdot \rrbracket_{CPS}$

$\lambda(f, g, k).\ (g\ ()\ (\lambda x.\ (f\ ()\ (\lambda y.\ \ldots))))$

$C_{BAD} \triangleq \lambda k.$

let $f = \lambda(\_, \_).\ k\ 1$

$g = \lambda(\_, \_).\ k\ 2$

in $[\cdot]\ f\ g\ k$

IGNORE "THEIR OWN" CONTINUATIONS

$C_{BAD}[\llbracket e_1 \rrbracket](id) \to^* 1 \not\simeq C_{BAD}[\llbracket e_2 \rrbracket](id) \to^* 2$

ABI: The Problem, More Broadly

Goal: $e_1 \approx e_2 \Rightarrow [\![e_1]\!] \approx [\![e_2]\!]$

More explicitly,

$$\forall C.\ C[e_1] \approx C[e_2] \Rightarrow \forall C.\ C[[\![e_1]\!]] \approx C[[\![e_2]\!]]$$

Universals are hard, so usually we do this:

$$\exists C.\ C[[\![e_1]\!]] \not\approx C[[\![e_2]\!]] \Rightarrow \exists C.\ C[e_1] \not\approx C[e_2]$$

To do so, define Back Translation $C \twoheadrightarrow C$

Problem: if T is more powerful than S, back translation isn't always possible!

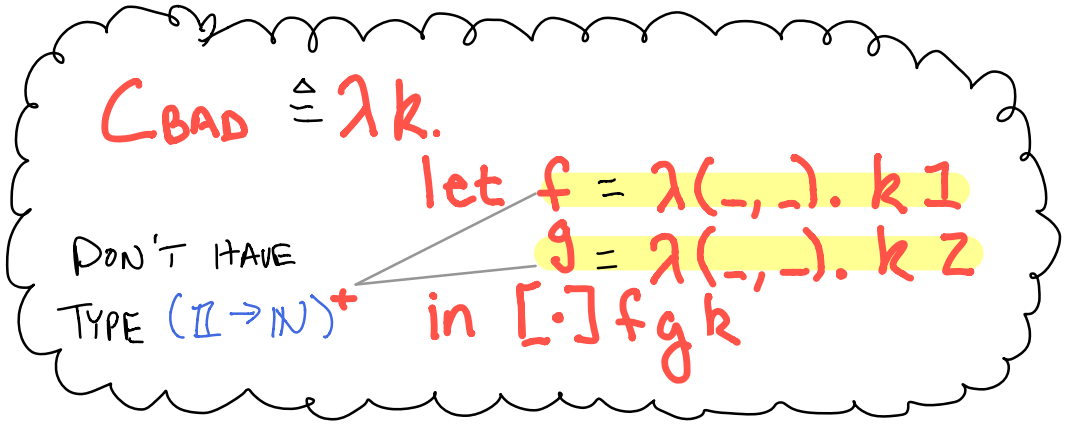
Ex: C_{BAD} can't be back translated to STLC

AB11: The Solution

Idea. Change type translation τ^+ so that $[\![e]\!]$ doesn't type check in bad contexts C

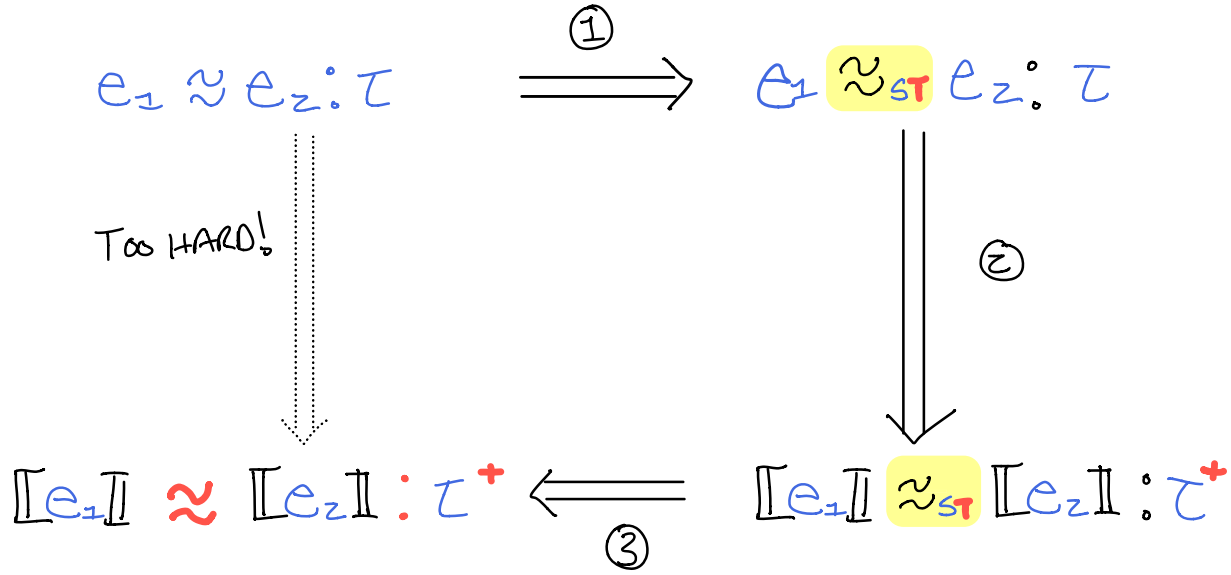
$$(\tau_1 \to \tau_2)^+ \triangleq \forall\alpha.\, \tau_1^+ \times (\tau_2^+ \to \alpha) \to \alpha$$

Intuition. Ensure that continuation $\tau_2^+ \to \alpha$ is actually used — no other way to return α!



AB11: Proof by Multi-Language

"An Equivalence-Preserving CPS Translation via Multi-Language Semantics"



1. Back translation
2. "Compiler correctness"
3. "Easy" because $T \subset ST$

- ST logical relation simpler than cross-language
- Isolating 1 + 2 critical
- Boundary cancellation required

AB11: TS Boundary

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash (\mathsf{TS}^{\tau}\, e) : \tau^{+}}$$

RECALL: $(\tau_1 \to \tau_2)^{+} \triangleq \forall \alpha.\, \tau_1^{+} \times (\tau_2^{+} \to \alpha) \to \alpha$

SYNTACTIC RESTR. ENFORCES CTRL FLOW

$$\mathcal{E}[\![\, \mathsf{let}\ y = \mathsf{TS}^{\tau_1 \to \tau_2} f\ \mathsf{in}\ e \,]\!]$$

$$\hookrightarrow \mathcal{E}[\![\, e[y \mapsto f] \,]\!]$$

WHERE $f = \lambda[\alpha](x : \tau_1^{+}, k : \tau_2^{+} \to \alpha).$
$\mathsf{let}\ z : \tau_2^{+} = \mathsf{TS}^{\tau_2}\,(f\ ({}^{\tau_1}\mathsf{ST}\, x))$
$\mathsf{in}\ k\ z$

AB11: ST BOUNDARY

$$\frac{\Gamma \vdash e : \tau^+}{\Gamma \vdash ({}^{\tau}\mathsf{S}\mathsf{T}\, e) : \tau}$$

RECALL: $\forall \alpha.\ \tau_1^+ \times (\tau_2^+ \to \alpha) \to \alpha = (\tau_1 \to \tau_2)^+$

$\mathcal{E}\,[{}^{\tau_1 \to \tau_2}\mathsf{S}\mathsf{T}\ f]$

$\hookrightarrow \mathcal{E}\,[\,\lambda(x{:}\tau_1).\ {}^{\tau_2}\mathsf{S}\mathsf{T}\ (\mathsf{let}\ x{:}\tau_1^+ = \mathsf{TS}^{\tau_1}\ x$
$\mathsf{in}\ f\ [\tau_2^+]\ x\ \mathsf{id})\,]$

"JUST RETURN RESULT"

RECAP: AB11

USE MULTI-LANGUAGE TO RELATE $e \dashv \llbracket e \rrbracket^S_T$

- MF07 TECHNIQUE HELPS WHEN $T > S$ IN POWER

- CASE STUDY: WIDER GAP BETWEEN LANGUAGES
  - > DIFF. CONTROL FLOW: DIRECT - 1ST CLASS CONTINUATIONS
  - > NONTRIVIAL TYPE TRANSLATION

- PAVED WAY FOR OPEN-WORLD COMPILER CORRECTNESS PROOFS

> THM: $e : \tau \rightsquigarrow \mathbf{e} : \tau^+ \Rightarrow e \approx_{ST} ({}^{\tau}S T\, \mathbf{e}) : \tau$

> COR: $e : \tau \rightsquigarrow \mathbf{e} : \tau^+ \Rightarrow \mathbf{e} \approx_{ST} (T S^{\tau}\, e) : \tau^+$

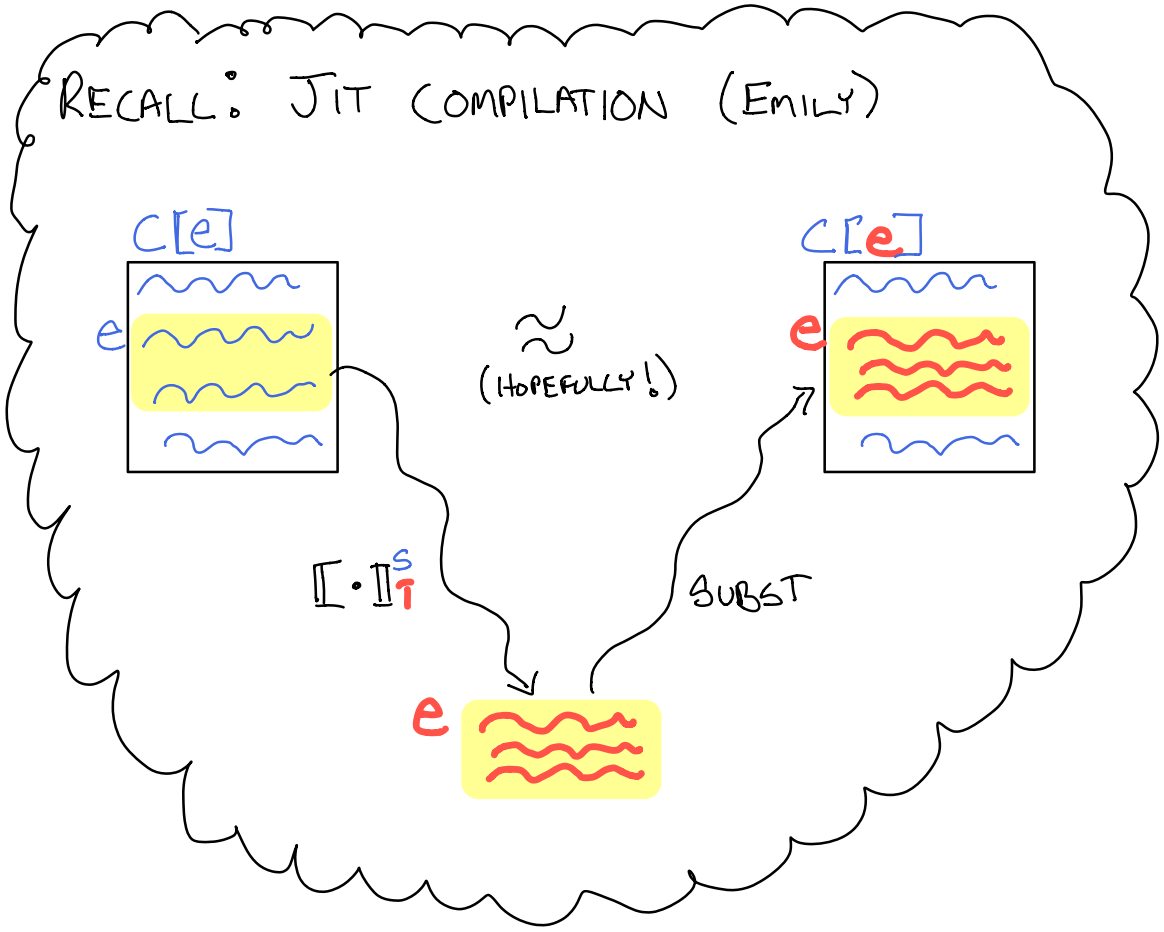
PF: BY BOUNDARY CANCELLATION. □

> COR: $e_1 \approx_{ST} e_2 : \tau \Rightarrow \llbracket e_1 \rrbracket \approx_{ST} \llbracket e_2 \rrbracket : \tau^+$

PF: EASY ☺ □

"FunTAL: Reasonably mixing
A
Functional language
with
[Typed] Assembly"
—Patterson et al. '17

PPDA 17: MOTIVATION



USE MULTI-LANG SEMANTICS TO REASON!

MAYBE,

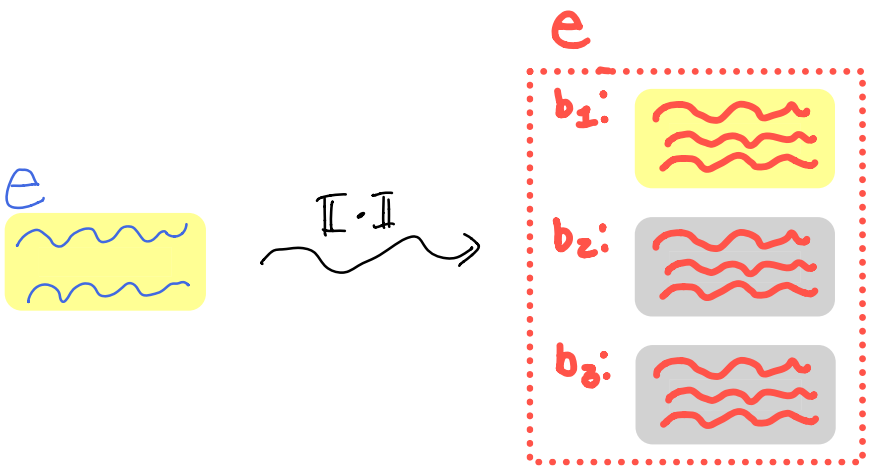
$$e \rightsquigarrow \mathbf{e} \Rightarrow e \approx_{ST} (ST\,\mathbf{e})$$

PPDA 17: What if T = Assembly?

Following Recipe:

$$\mathcal{E}[F\ T\ e] \longrightarrow^* \mathcal{E}[F\ T\ v] \rightarrow \mathcal{E}[v]$$

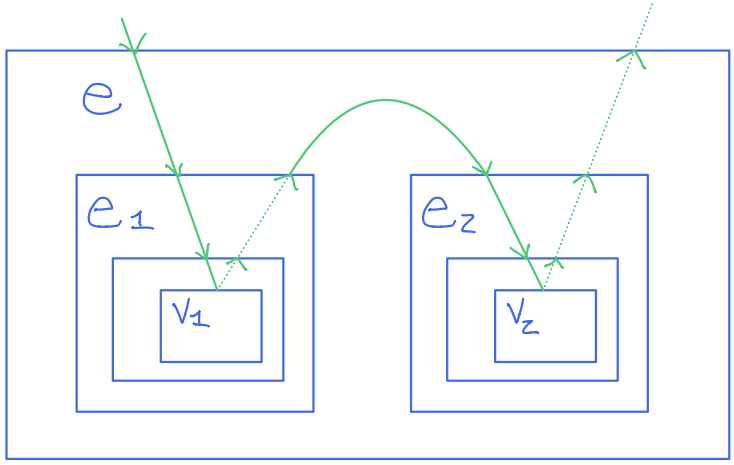
What even are these?

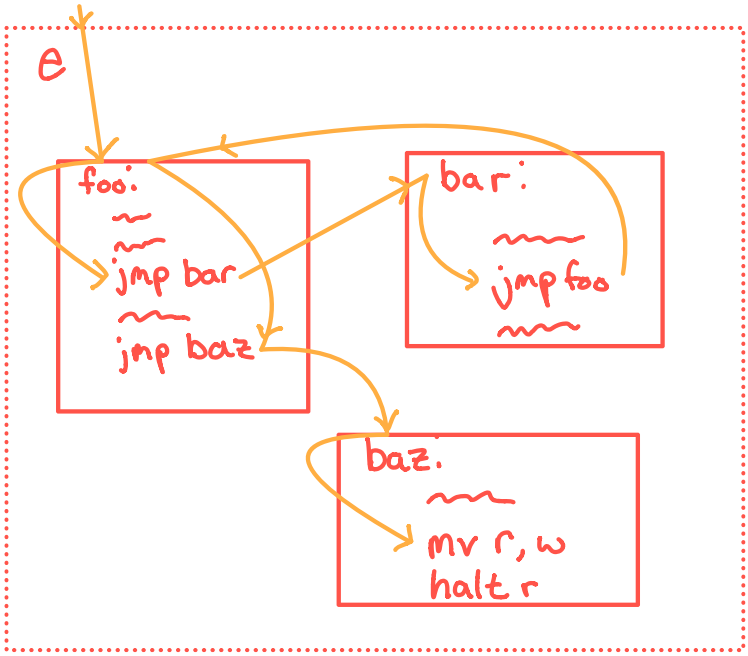


Component $e ::= (I, H)$

- I: Initial Instruction Sequence
- H: Heap fragment with other basic blocks

PPDA17: Control Flow

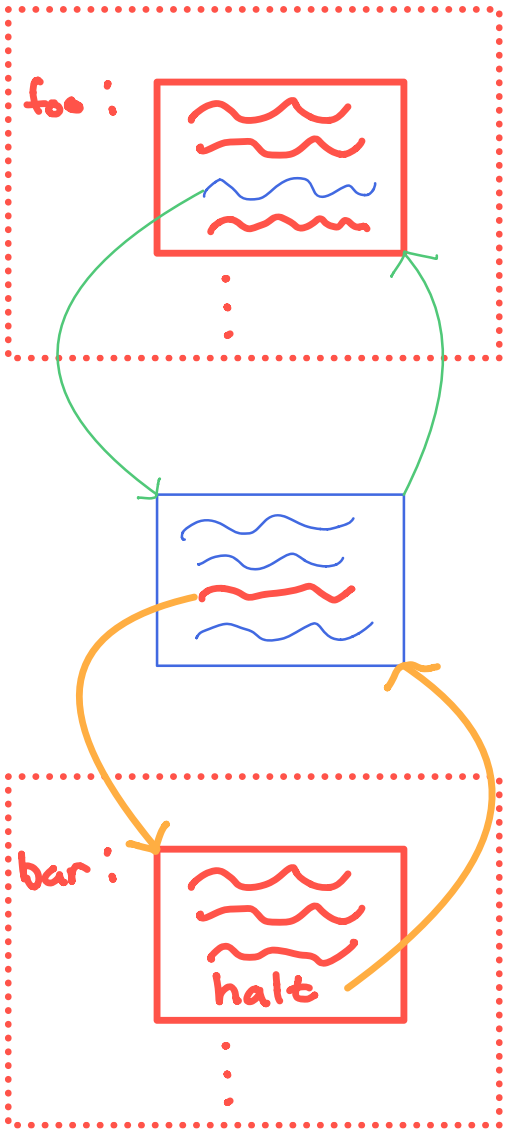




- Serious mismatch!
- Don't want to change control flow of either
- Use rich type system to guide boundaries

PPOA17: Stack Protection

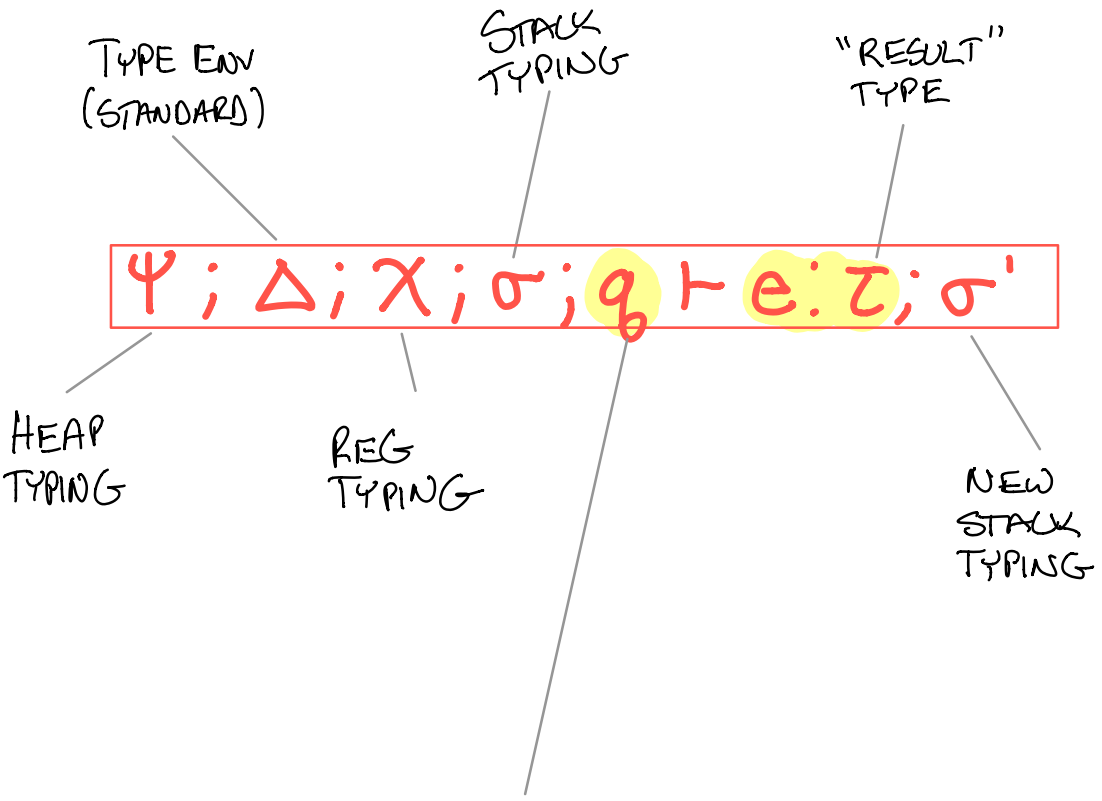
Recall: Caller-saved vs. Callee-saved



What if bar clobbers the stack?

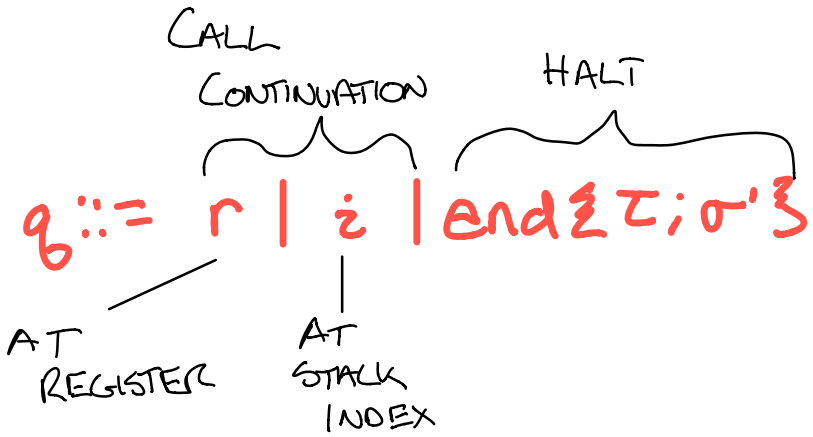
PPDA 17: A TASTE OF TAL

BUILDING ON TAL WORK (MORRISETT ET AL. '98, '02)



RETURN MARKER

WHAT DO WE DO WHEN e IS DONE?



PPDA17. FT Boundary (Simplified)

$v ::= \text{halt}\ \tau\ \{r\}$

Type of "result" — Where it's located

Current state of memory

$\langle M \mid \mathcal{E}[\ ^{\tau}FT(\text{halt}\ \tau^{+}\{r\}, \_)]\rangle$

$\hookrightarrow \langle M' \mid \mathcal{E}[v]\rangle$

where $^{\tau}FT(M.R(r), M) = (v, M')$

Look up value in register r

Translation of $M.R(r)$

Memory can change during translation!

PPDA17. TF Boundary

$\langle M \mid \mathcal{E}[\text{import } r, {}^{\sigma}TF^{\tau}\ v\ ; I] \rangle$

- r: Register to import into
- v: Value to be imported

$\hookrightarrow \langle M' \mid \mathcal{E}[\text{mv } r, v\ ; I] \rangle$

- mv r, v: Load v into r

If $TF^{\tau}(v, M) \simeq (v', M')$

- v': Translation of v
- M': Again, memory can change during translation!

PPDA17: Recap

TAL components (I, H) + return markers &
⇒ compatability at boundaries

- Case study: wider gap between languages
  - > Control flow: direct – **unstructured**
  - > Mutability: immutable – **mutable**

- Didn't define $[\![\cdot]\!]^F_T$

- No correctness prop like

$$e \rightsquigarrow \mathbf{e} \not\Rightarrow e \approx_{FT} (FT\,\mathbf{e})$$

  - > May not always want this!
    Intuitively: T can't make use of extra power

Summary

Boundaries

$$\mathcal{E}[(\lceil A \rceil B\ e)] \rightarrow^* \mathcal{E}[(\lceil A \rceil B\ v)] \rightarrow \mathcal{E}[v]$$

	MF07	AB11	PPDA17
Lang	"Scheme" / "ML"	STLC / Sys F	STLC$^+$ / TAL
Types	Dynamic / Static	Static / Static	Static / Static
Control Flow	Direct / Direct	Direct / CPS	Direct / ??

- Design Choices
  - > What / How do we translate?
  - > What / When do we check?
  - > What equivalences do we break / preserve?

- Influenced by Contracts
- Legacy in
  - > Gradual Typing
  - > Secure Compilation

WHAT NOW?

- OTHER MF07 APPS
  - > DEPENDENT TYPES (OSERA ET AL. '12)
  - > LINEAR TYPES (SCHERER ET AL. '18)

- NEVER IMPLEMENTED AT SCALE

- MF07 REQUIRES BESPOKE SPEC FOR EVERY MIX
  - > WITH n LANGUAGES, n^2 SPECS
  - > DOESN'T SCALE TO FULL ECOSYSTEM

- LINKING TYPES? (DANIEL + AMAL)
  - > EACH LANG SPECIFIES ONLY ITS SIDE OF BOUNDARY
  - > EXPLICITLY STATE WHERE EQV. CAN BREAK

AB11°. Open World CC ⇒ Equ Pres

Thm°. $e \cdot \tau \rightsquigarrow \mathbf{e} : \tau^{+} \Rightarrow e \approx_{ST} ({}^{\tau}S\,T\,\mathbf{e}) : \tau$

Cor°. $e_1 \approx_{ST} e_2 : \tau \Rightarrow [\![e_1]\!] \approx_{ST} [\![e_2]\!] : \tau^{+}$

Pf°.

First,

$$\begin{array}{ccc} e_1 & \overset{\text{Asum.}}{\approx} & e_2 \\ \text{Thm.}\ \S\S & & \S\S\ \text{Thm.} \\ ({}^{\tau}S\,T\,[\![e_1]\!]) & \overset{\text{Trans.}}{\approx} & ({}^{\tau}S\,T\,[\![e_2]\!]) \end{array}$$

Then,

$$({}^{\tau}S\,T\,[\![e_1]\!]) \approx_{ST} ({}^{\tau}S\,T\,[\![e_2]\!])$$

Def Equ ↳ $(T\,S^{\tau}\,({}^{\tau}S\,T\,[\![e_1]\!])) \approx_{ST} (T\,S^{\tau}\,({}^{\tau}S\,T\,[\![e_1]\!]))$

Boundary Cancellation ↳ $[\![e_1]\!] \approx_{ST} [\![e_2]\!]$

□