Full abstraction for multi-language systems
ML plus linear types

Gabriel Scherer, Max New, Nick Rioux, Amal Ahmed

INRIA, France
Northeastern University, Boston, USA

April 15, 2018

A question worth asking

What does it **mean** for two languages to "interact well together"?

A question worth asking

What does it **mean** for two languages to "interact well together"?

- no segfaults?

A question worth asking

What does it **mean** for two languages to “interact well together”?

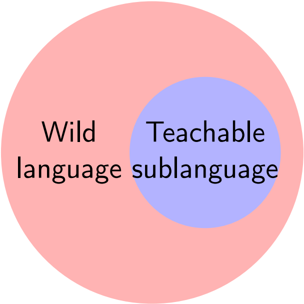
- no segfaults?
- the type systems are not broken?
  (correspondence between types on both sides, or runtime checks)

A question worth asking

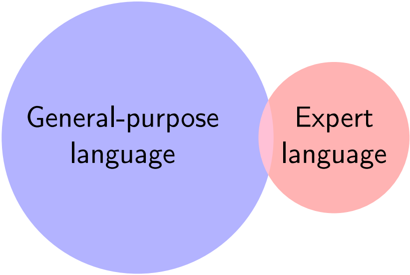
What does it **mean** for two languages to "interact well together"?

- no segfaults?
- the type systems are not broken?
  (correspondence between types on both sides, or runtime checks)
- more?

Multi-language stories



Abstraction leaks?



Graceful interoperation?

Full abstraction

$[\![\_]\!] : S \longrightarrow T$ fully abstract:

$$a \approx^{ctx} b \quad \iff \quad [\![a]\!] \approx^{ctx} [\![b]\!]$$

Full abstraction preserves (equational) reasoning.

Full abstraction

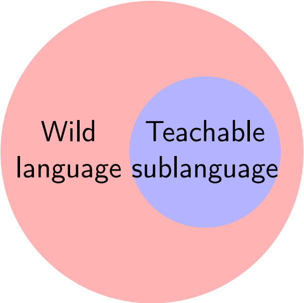
$[\![\_]\!] : S \longrightarrow T$ fully abstract:

$$a \approx^{ctx} b \quad \iff \quad [\![a]\!] \approx^{ctx} [\![b]\!]$$

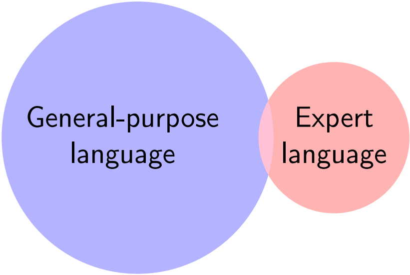
Full abstraction preserves (equational) reasoning.

Can be used to think about compilation, but not only...

Full abstraction for multi-language systems

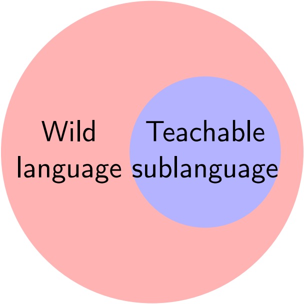


No abstraction leaks: $T \xrightarrow{f.a.} W$

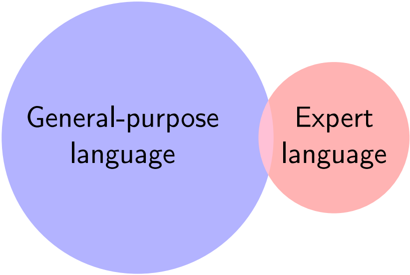


Graceful interoperation: $G \xrightarrow{f.a.} (G + E)$

Full abstraction for multi-language systems



No abstraction leaks: $T \xrightarrow{f.a.} W$



Graceful interoperation: $G \xrightarrow{f.a.} (G + E)$

In this talk: a first experiment on **ML** plus **linear types**.

U: a core ML

$$\Gamma \vdash_{\mathsf{u}} \mathsf{e} : \sigma$$

L: linear types

Resource tracking, unique ownership.

$$\sigma \qquad !\sigma \qquad \Gamma \qquad !\Gamma$$

$$\Gamma \vdash_{l} e : \sigma$$

We own e at type σ (duplicable or not), e owns the resources in Γ.

Multi-language applications

Protocol with resource handling requirements.

"This file descriptor must be closed"

$$\begin{array}{lcl} \text{open} & : & !(![\text{Path}] \multimap \text{Handle}) \\ \text{read\_line} & : & !(\text{Handle} \multimap (\text{Handle} \oplus (![\text{String}] \otimes \text{Handle}))) \\ \text{close} & : & !(\text{Handle} \multimap 1) \end{array}$$

Typestate.

Multi-language applications

Protocol with resource handling requirements.

"This file descriptor must be closed"

$$\begin{array}{lcl} \text{open} & : & !(![\text{Path}] \multimap \text{Handle}) \\ \text{read\_line} & : & !(\text{Handle} \multimap (\text{Handle} \oplus (![\text{String}] \otimes \text{Handle}))) \\ \text{close} & : & !(\text{Handle} \multimap 1) \end{array}$$

Typestate.

```
pattern EOF handle = inl handle
pattern Next line handle = inr (line, handle)
```

$$\begin{array}{lcl} \mathsf{open} & : & !(![\mathsf{Path}] \multimap \mathsf{Handle}) \\ \mathsf{read\_line} & : & !(\mathsf{Handle} \multimap (\mathsf{Handle} \oplus (![\mathsf{String}] \otimes \mathsf{Handle}))) \\ \mathsf{close} & : & !(\mathsf{Handle} \multimap 1) \end{array}$$

```
let concat_lines path : String = UL(
  loop (open LU(path)) LU(Nil)
  where rec loop handle (acc : ![List String]) =
    match read_line handle with
    | Next line handle ->
      loop handle LU(Cons UL(line) UL(acc))
    | EOF handle ->
      close handle; LU(rev_concat "\n" UL(acc)))
```

$$\frac{!\Gamma \vdash_{\mathsf{lu}} \mathsf{e} : \sigma}{!\Gamma \vdash_{\mathsf{ul}} \mathcal{LU}(\mathsf{e}) : ![\sigma]} \qquad \frac{!\Gamma \vdash_{\mathsf{ul}} \mathsf{e} : ![\sigma]}{!\Gamma \vdash_{\mathsf{lu}} \mathcal{UL}(\mathsf{e}) : \sigma}$$

(opaque boundaries)

Linear types: linear locations

Box 1 σ: full cell

Box 0 σ: empty cell

$$1 \underset{\text{free}}{\overset{\text{new}}{\rightleftarrows}} \text{Box } 0\ \sigma \qquad\qquad \text{Box } 1\ \sigma \underset{\text{box}}{\overset{\text{unbox}}{\rightleftarrows}} \text{Box } 0\ \sigma \otimes \sigma$$

Applications: in-place reuse of memory cells.

List reversal

```
type LList a = μt. 1 ⊕ Box 1 (a ⊗ t)
pattern Nil = inl ()
pattern Cons l x xs = inr (box (l, (x, xs)))

val reverse : LList a ⊸ LList a
let reverse list = loop Nil list
  where rec loop acc = function
  | Nil → acc
  | Cons l x xs → loop (Cons l x acc) xs

type List a = μt. 1 + (a × t)
let reverse list =
  UL{!LList _}(share (reverse (copy (LU{!LList _}(list)))))
```

List reversal

```
type LList a = μt. 1 ⊕ Box 1 (a ⊗ t)
pattern Nil = inl ()
pattern Cons l x xs = inr (box (l, (x, xs)))

val reverse : LList a ⊸ LList a
let reverse list = loop Nil list
  where rec loop acc = function
  | Nil → acc
  | Cons l x xs → loop (Cons l x acc) xs

type List a = μt. 1 + (a × t)
let reverse list =
  UL{!LList _}(share (reverse (copy (LU{!LList _}(list)))))
```

List a $\simeq$!LList ![a] $\qquad \vdash_{\mathsf{ul}} \sigma \simeq \sigma$

(transparent boundaries)

```
let partition p li = partition_aux p (Nil, Nil) li
partition_aux p (yes, no) = function
| Nil -> (yes, no)
| Cons l x xs ->
  let (yes, no) =
    if copy p x then (Cons l x yes, no) else (yes, Cons l x no)
  in partition_aux p (yes, no) xs

let lin_quicksort li = quicksort_aux li Nil
let quicksort_aux li acc = match li with
| Nil -> acc
| Cons l head li ->
  let p = share (fun x -> x < head) in
  let (below, above) = partition p li in
  quicksort_aux below (Cons l head (quicksort_aux above acc))
quicksort li =
  UL{!LList _}(share (lin_quicksort (copy LU{!LList _}(li))))
```

Full abstraction

Theorem

The embedding of U into UL is fully abstract.

Proof: by pure interpretation of the linear language into ML.
(Cogent)

Questions ?

Thanks!

$$\sigma \quad ::= \quad \alpha \mid \sigma_1 \times \sigma_2 \mid 1 \mid \sigma_1 \to \sigma_2 \mid$$
$$\sigma_1 + \sigma_2 \mid \mu\alpha.\,\sigma \mid \forall\alpha.\,\sigma$$

$$\sigma \quad ::= \quad \sigma_1 \otimes \sigma_2 \mid 1 \mid \sigma_1 \multimap \sigma_2 \mid$$
$$\sigma_1 \oplus \sigma_2 \mid \mu\alpha.\,\sigma \mid \alpha \mid$$
$$!\sigma \mid$$
$$\mathsf{Box\ b}\ \sigma$$

Linear typing rules

$$\frac{}{!\Gamma, x{:}\sigma \vdash_l x : \sigma} \qquad \frac{}{!\Gamma \vdash_l \langle\rangle : 1} \qquad \frac{\Gamma \vdash_l e : 1 \qquad \Gamma' \vdash_l e' : \sigma}{\Gamma \curlyvee \Gamma' \vdash_l e; e' : \sigma}$$

$$\frac{\Gamma_1 \vdash_l e_1 : \sigma_1 \qquad \Gamma_2 \vdash_l e_2 : \sigma_2}{\Gamma_1 \curlyvee \Gamma_2 \vdash_l \langle e_1, e_2\rangle : \sigma_1 \otimes \sigma_2} \qquad \frac{\Gamma \vdash_l e : \sigma_1 \otimes \sigma_2 \quad \Gamma', x_1{:}\sigma_1, x_2{:}\sigma_2 \vdash_l e' : \sigma}{\Gamma \curlyvee \Gamma' \vdash_l \mathsf{let}\ \langle x_1, x_2\rangle = e\ \mathsf{in}\ e' : \sigma}$$

$$\frac{\Gamma, x{:}\sigma \vdash_l e : \sigma'}{\Gamma \vdash_l \lambda(x{:}\sigma).\, e : \sigma \multimap \sigma'} \qquad \frac{\Gamma \vdash_l e : \sigma' \multimap \sigma \qquad \Gamma' \vdash_l e' : \sigma'}{\Gamma \curlyvee \Gamma' \vdash_l e\ e' : \sigma}$$

$$\frac{\Gamma \vdash_l e : \sigma_i}{\Gamma \vdash_l \mathsf{inj_i}\ e : \sigma_1 \oplus \sigma_2} \qquad \frac{\Gamma \vdash_l e : \sigma_1 \oplus \sigma_2 \qquad (\Gamma', x_i : \sigma_i \vdash_l e_i : \sigma)_{i\in\{1,2\}}}{\Gamma \curlyvee \Gamma' \vdash_l \mathsf{case}\ e\ \mathsf{of}\ x_1.\, e_1 \mid x_2.\, e_2 : \sigma}$$

$$\frac{!\Gamma \vdash_l e : \sigma}{!\Gamma \vdash_l \mathsf{share}\ e : !\sigma} \qquad \frac{\Gamma \vdash_l e : !\sigma}{\Gamma \vdash_l \mathsf{copy}^{\sigma}\ e : \sigma} \qquad \mu\alpha.\,\sigma \ \overset{\mathsf{unfold}}{\underset{\mathsf{fold}_{\mu\alpha.\sigma}}{\rightleftarrows}}\ \sigma[\mu\alpha.\,\sigma/\alpha]$$

Interaction: lump

Types $\sigma \mid \sigma$

$$\sigma$$
$$\sigma \quad + ::= \cdots \mid [\sigma]$$

Values $\mathsf{v} \mid \mathsf{v}$

$$\mathsf{v}$$
$$\mathsf{v} \quad + ::= \cdots \mid [\mathsf{v}]$$

Expressions $\mathsf{e} \mid \mathsf{e}$

$$\mathsf{e} \quad + ::= \cdots \mid \mathcal{UL}(\mathsf{e})$$
$$\mathsf{e} \quad + ::= \cdots \mid \mathcal{LU}(\mathsf{e})$$

Contexts $\Gamma ::= \cdot \mid \Gamma, \mathsf{x}{:}\sigma \mid \Gamma, \alpha \mid \Gamma, \mathsf{x}{:}\sigma$

$$\frac{!\Gamma \vdash_{\mathsf{lu}} \mathsf{e} : \sigma}{!\Gamma \vdash_{\mathsf{ul}} \mathcal{LU}(\mathsf{e}) : ![\sigma]} \qquad \frac{!\Gamma \vdash_{\mathsf{ul}} \mathsf{e} : ![\sigma]}{!\Gamma \vdash_{\mathsf{lu}} \mathcal{UL}(\mathsf{e}) : \sigma}$$

Interaction: compatibility

Compatibility relation $\boxed{\vdash_{\mathsf{ul}} \sigma \simeq \sigma}$

$$\frac{}{\vdash_{\mathsf{ul}} 1 \simeq !1} \qquad \frac{\vdash_{\mathsf{ul}} \sigma_1 \simeq !\sigma_1 \qquad \vdash_{\mathsf{ul}} \sigma_2 \simeq !\sigma_2}{\vdash_{\mathsf{ul}} \sigma_1 \times \sigma_2 \simeq !(\sigma_1 \otimes \sigma_2)}$$

$$\frac{\vdash_{\mathsf{ul}} \sigma_1 \simeq !\sigma_1 \qquad \vdash_{\mathsf{ul}} \sigma_2 \simeq !\sigma_2}{\vdash_{\mathsf{ul}} \sigma_1 + \sigma_2 \simeq !(\sigma_1 \oplus \sigma_2)} \qquad \frac{\vdash_{\mathsf{ul}} \sigma \simeq !\sigma \qquad \vdash_{\mathsf{ul}} \sigma' \simeq !\sigma'}{\vdash_{\mathsf{ul}} \sigma \to \sigma' \simeq !(!\sigma \multimap !\sigma')}$$

$$\frac{}{\vdash_{\mathsf{ul}} \sigma \simeq ![\sigma]} \qquad \frac{\vdash_{\mathsf{ul}} \sigma \simeq !\sigma}{\vdash_{\mathsf{ul}} \sigma \simeq !!\sigma} \qquad \frac{\vdash_{\mathsf{ul}} \sigma \simeq !\sigma}{\vdash_{\mathsf{ul}} \sigma \simeq !(\mathsf{Box}\ 1\ \sigma)}$$

Interaction primitives and derived constructs:

$$![\sigma] \ \underset{\mathsf{lump}^{\sigma}}{\overset{{}^{\sigma}\mathsf{unlump}}{\rightleftarrows}} \ \sigma \quad \text{when} \quad \vdash_{\mathsf{ul}} \sigma \simeq \sigma$$

$${}^{\sigma}\mathcal{LU}(\mathsf{e}) \overset{\text{def}}{=} {}^{\sigma}\mathsf{unlump}\ \mathcal{LU}(\mathsf{e})$$

$$\mathcal{UL}^{\sigma}(\mathsf{e}) \overset{\text{def}}{=} \mathcal{UL}(\mathsf{lump}^{\sigma}\ \mathsf{e})$$