Full abstraction for multi-language systems
ML plus linear types

Gabriel Scherer, Amal Ahmed, Max New

Northeastern University, Boston

January 15, 2017

Multi-language systems

Languages of today tend to evolve into behemoths by piling features up: C++, Scala, GHC Haskell, OCaml...

Multi-language systems: several languages working together to cover the feature space. (simpler?)

Multi-language system **design** may include designing new languages for interoperation.

Full abstraction to understand graceful language interoperability.

Full abstraction for multi-language systems

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

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

Full abstraction preserves (equational) reasoning.

Full abstraction for multi-language systems

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

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

Full abstraction preserves (equational) reasoning.

$$\begin{tikzcd}
S_1 \arrow[dr, "\text{full abs.}"'] & & S_2 \arrow[dl, "\text{full abs.}"] \\
& T &
\end{tikzcd}$$

Mixed S_1, S_2 programs preserve (equational) reasoning of their fragments.

Full abstraction for multi-language systems

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

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

Full abstraction preserves (equational) reasoning.

$$\begin{tikzcd}
S_1 \arrow[rr, squiggly, no head, "\text{interop.}"] \arrow[dr, "\text{full abs.}"'] & & S_2 \arrow[dl, "\text{full abs.}"] \\
& T &
\end{tikzcd}$$

Mixed S_1, S_2 programs preserve (equational) reasoning of their fragments.
Graceful multi-language semantics.
(or vice versa)

Full abstraction for multi-language systems

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

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

Full abstraction preserves (equational) reasoning.

$$\begin{tikzcd}
S_1 \arrow[rr, squiggly, dash, "\text{interop.}"] \arrow[dr, "\text{full abs.}"'] & & S_2 \arrow[dl, "\text{full abs.}"] \\
& T &
\end{tikzcd}$$

Mixed S_1, S_2 programs preserve (equational) reasoning of their fragments.
Graceful multi-language semantics.
(or vice versa)

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

U: a core ML

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

$\mathbf{L}$: linear types

Resource tracking, unique ownership.

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

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

We own $\mathbf{e}$ at type σ (duplicable or not), $\mathbf{e}$ owns the resources in Γ.

Multi-language applications

Protocol with resource handling requirements.

"This file descriptor must be closed"

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

(details about the boundaries come later)

Typestate.

(details about the boundaries come later)

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

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

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

Linear types: linear locations

Box $1\ \sigma$: full cell

Box $0\ \sigma$: empty cell

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

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 tail = function
  | Nil → tail
  | Cons l x xs → loop (Conx l x tail) xs

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

$$\vdash_{\mathsf{ul}} \sigma \simeq \sigma$$

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!

Interaction: lump

Types σ | σ

$$\begin{aligned} &\sigma \\ &\sigma \quad +::= \cdots \mid [\sigma] \end{aligned}$$

Values v | $\mathbf{v}$

$$\begin{aligned} &\mathsf{v} \\ &\mathbf{v} \quad +::= \cdots \mid [\mathsf{v}] \end{aligned}$$

Expressions e | $\mathbf{e}$

$$\begin{aligned} &\mathsf{e} \quad +::= \cdots \mid \mathcal{UL}(\mathbf{e}) \\ &\mathbf{e} \quad +::= \cdots \mid \mathcal{LU}(\mathsf{e}) \end{aligned}$$

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

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

Interaction: compatibility

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

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

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

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

Interaction primitives and derived constructs:

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

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

$$\mathcal{UL}^\sigma(\mathbf{e}) \stackrel{\text{def}}{=} \mathcal{UL}(\text{lump}^\sigma\ \mathbf{e})$$