Full abstraction for multi-language systems
ML plus linear types

Gabriel Scherer, Amal Ahmed, Max New

Northeastern University, Boston

January 15, 2017
Multilingual systems

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

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

Multilingual system design may include designing new languages for interoperation.

Full abstraction to understand graceful language interoperability.
Full abstraction for multi-language systems

\[ 
\boxed{\_} : S \rightarrow T \text{ fully abstract:} 
\]

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

Full abstraction preserves (equational) reasoning.
Full abstraction for multi-language systems

\[ \boxed{\cdot} : S \rightarrow T \text{ fully abstract:} \]

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

Full abstraction preserves (equational) reasoning.

Mixed \( S_1, S_2 \) programs preserve (equational) reasoning of their fragments.
Full abstraction for multi-language systems

\[ \boxed{} : S \to T \text{ fully abstract:} \]

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

Full abstraction preserves (equational) reasoning.

\[
\begin{array}{c}
S_1 \xrightarrow{\text{interop.}} S_2 \\
\text{full abs.} \quad \quad \quad \quad \text{full abs.}
\end{array}
\]

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

\[ \llbracket \_ \rrbracket : S \to T \text{ fully abstract:} \]

\[ a \approx_{ctx}^\llbracket \cdot \rrbracket b \implies \llbracket a \rrbracket \approx_{ctx}^\llbracket \cdot \rrbracket \llbracket b \rrbracket \]

Full abstraction preserves (equational) reasoning.

\[ \begin{array}{c}
S_1 \xrightarrow{\text{interop.}} S_2 \\
S_1 \xrightarrow{\text{full abs.}} T \\
S_2 \xrightarrow{\text{full abs.}} T
\end{array} \]

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

Γ ⊢_u e : σ
L: linear types

Resource tracking, unique ownership.

\[ \sigma \quad !\sigma \quad \Gamma \quad !\Gamma \]

\[ \Gamma \vdash_e e : \sigma \]

We own \( e \) at type \( \sigma \) (duplicable or not), \( e \) owns the resources in \( \Gamma \).
Multi-language applications

Protocol with resource handling requirements.

“This file descriptor must be closed”

\[
\begin{align*}
\text{open} & : \text{!}(\text{Path} \rightarrow \text{Handle}) \\
\text{line} & : \text{!}(\text{Handle} \rightarrow (\text{Handle} \oplus (\text{!}[\text{String}] \otimes \text{Handle}))) \\
\text{close} & : \text{!}(\text{Handle} \rightarrow 1)
\end{align*}
\]

(details about the boundaries come later)

Typestate.
open : !( ![Path] → Handle )
line : !( Handle → ( Handle ⊕ ( ![String] ⊗ Handle )))
close : !( Handle → 1 )

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))

!Γ ⊢ lu e : σ
!Γ ⊢ ul e : ![σ]
!Γ ⊢ ul LU(e) : ![σ]
!Γ ⊢ ul UL(e) : σ
Linear types: linear locations

**Box 1 $\sigma$:** full cell

**Box 0 $\sigma$:** empty cell

Applications: in-place reuse of memory cells.
List reversal

\[
\text{type LList} \ a = \mu t. \ 1 \oplus \text{Box} \ 1 \ (a \otimes t) \\
\text{pattern Nil} = \text{inl} () \\
\text{pattern Cons} \ l \ x \ xs = \text{inr} \ (\text{box} \ (l, (x, xs)))
\]

\[
\text{val} \ \text{reverse} : \ \text{LList} \ a \to \text{LList} \ a \\
\text{let} \ \text{reverse} \ \text{list} = \text{loop} \ \text{Nil} \ \text{list} \\
\quad \text{where rec loop tail} = \text{function} \\
\quad \ | \ \text{Nil} \to \text{tail} \\
\quad \ | \ \text{Cons} \ l \ x \ xs \to \text{loop} \ (\text{Conx} \ l \ x \ \text{tail}) \ xs
\]

\[
\text{type List} \ a = \mu t. \ 1 + (a \times t) \\
\text{let} \ \text{reverse} \ \text{list} = \text{UL}(\text{share} \ (\text{reverse} \ (\text{copy} \ (\text{LU}(\text{list})))))
\]

\[
\vdash_{ul} \sigma \simeq \sigma
\]
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 \( \sigma \mid \sigma \)

Values \( v \mid v \)

\[
\sigma \\
\sigma + ::= \cdots \mid [\sigma] \\
v \\
v + ::= \cdots \mid [v]
\]

Expressions \( e \mid e \)

\[
e + ::= \cdots \mid UL(e) \\
e + ::= \cdots \mid LU(e)
\]

Contexts \( \Gamma ::= \cdot \mid \Gamma, x: \sigma \mid \Gamma, \alpha \mid \Gamma, x: \sigma \)

\[
!\Gamma \vdash_{lu} e : \sigma \\
!\Gamma \vdash_{ul} LU(e) : ![\sigma] \\
!\Gamma \vdash_{ul} LU(e) : \sigma
\]
Interaction: compatibility

Compatibility relation \( \vdash \text{ul} \sigma \simeq \sigma \)

\[
\begin{align*}
\vdash \text{ul} 1 & \simeq !1 \\
\vdash \text{ul} \sigma_1 & \simeq !\sigma_1 & \vdash \text{ul} \sigma_2 & \simeq !\sigma_2 \\
\vdash \text{ul} \sigma_1 & \times \sigma_2 & \simeq ! (\sigma_1 \times \sigma_2) \\
\vdash \text{ul} \sigma_1 & \simeq !\sigma_1 & \vdash \text{ul} \sigma_2 & \simeq !\sigma_2 \\
\vdash \text{ul} \sigma_1 & \simeq !\sigma_2 \\
\vdash \text{ul} \sigma_1 & \simeq ! !\sigma \\
\vdash \text{ul} \sigma & \simeq ![\sigma] \\
\vdash \text{ul} \sigma & \simeq !([\sigma]) \\
\vdash \text{ul} \sigma & \rightarrow \sigma' \simeq ![!\sigma \rightarrow !\sigma'] \\
\vdash \text{ul} \sigma & \simeq ![[\sigma]] \\
\vdash \text{ul} \sigma & \simeq [[\sigma]] \\
\vdash \text{ul} \sigma & \simeq ![\text{Box} 1 \sigma]
\end{align*}
\]

Interaction primitives and derived constructs:

\[
\begin{align*}
\text{\(\sigma\)}\text{unlump} \\
\text{\([\sigma]\)} & \rightarrow \sigma & \text{when} & \vdash \text{ul} \sigma \simeq \sigma \\
\text{lump}\(\sigma\) \\
\text{\(\sigma\LU(e)\)} & \text{def} & \text{\(\sigma\)}\text{unlump} \text{LU}(e) \\
\text{\(UL\sigma(e)\)} & \text{def} & \text{UL(lump}\(\sigma\) e)
\end{align*}
\]