Coeffects

Unified static analysis of context-dependence

Tomas Petricek, University of Cambridge
tomas.petricek@cl.cam.ac.uk | @tomaspetricek

Joint work with Dominic Orchard and Alan Mycroft
Properties of computations

**Effects on** the environment

\[ \Gamma \vdash e : M^\sigma \tau \]

**Effect systems and monads**

**Requirements from** the environment

\[ \mathcal{C}^\sigma \Gamma \vdash e : \tau \]

**Coeffect systems and comonads**
Effect systems

Imperative language with global locations

\[
\begin{align*}
\text{let mutateGlobal} &= \text{fun} \ (x) \rightarrow \ l := x \\
\Gamma \vdash e : M^\sigma \tau \\
\Gamma \vdash l := e : M^\sigma U\{\text{write}(l)\} \ \text{unit} \\
\Gamma, x: \tau_1 \vdash e : M^\sigma \tau_2 \\
\Gamma \vdash \lambda x. e : M^\emptyset \ (\tau_1 \rightarrow M^\sigma \tau_2)
\end{align*}
\]
Is lambda abstraction “pure”?

“In the rule for abstraction, the effect is empty because evaluation immediately returns the function, with no side effects. The effect on the function arrow is the same as the effect for the function body, because applying the function will have the same side effects as evaluating the body.”

[The marriage of effects and monads (2003)]

Context-dependent properties are different!
Think resources in distributed languages
Coeffects in action
**Implicit parameter coeffects**

Dynamically scoped parameters `?param`

```ocaml
let ?culture = "en-US"

let print = fun (num) →
    printNumber num ?culture ?format
```

Abstraction splits requirements

\[
C^{r \cup s}(\Gamma, x: \tau_1) \vdash e: \tau_2
\]

\[
C^r \Gamma \vdash \lambda x. e: C^s \tau_1 \rightarrow \tau_2
\]

← call site

July 11

ICALP 2013, Riga
Implicit parameter effects

Dynamically scoped parameters ?param

let ?culture = "en-US"

let print = fun (num) →
    printNumber num ?culture ?format

Different typing for different uses

\( C^{\{?\text{culture}\}} \Gamma \vdash \text{print} : C^{\{?\text{format}\}} \text{int} \rightarrow \text{unit} \)

\( C^{\emptyset} \Gamma \vdash \text{print} : C^{\{?\text{format, ?culture}\}} \text{int} \rightarrow \text{unit} \)
Efficient data-flow language

Access past value using `prev`

How large cache is needed?

\[
Cr\Gamma \vdash e : \tau \\
Cr+1\Gamma \vdash prev\ e : \tau
\]

let skipOne = if (prev tick) = 1 then (fun (x) → x) 
else (fun (x) → prev x) 
skipOne (prev counter)
Liveness coeffect

Is the variable context live or dead?

$$\frac{x: \tau \in \Gamma}{C^1 \Gamma \vdash x: \tau}$$

$$\frac{n \in \{0,1,2,\ldots\}}{C^0 \Gamma \vdash n: \iota}$$

Application propagates contexts

foo 42  (fun x → 42) y  (fun x → y) 42

$$\frac{C^r \Gamma \vdash e_1: C^t \tau_1 \rightarrow \tau_2 \quad C^s \Gamma \vdash e_2: \tau_1}{C^{r \lor (s \land t)} \Gamma \vdash e_1 \; e_2: \tau_2}$$
Semantics of liveness

Context with liveness annotation

$$[C^r \Gamma \vdash e : \tau] = C^r(\tau_1 \times \cdots \times \tau_n) \to \tau$$

Indexed **Maybe** type

- Definitely dead: **Nothing**
  - $C^0 \tau = 1$
- Maybe live: **Just** of context
  - $C^1 \tau = \tau$

Impossible using a monad!

$$[C^r \Gamma \vdash e : \tau] \neq (\tau_1 \times \cdots \times \tau_n) \to M^\sigma \tau$$
Unified coeffect system
Flat coeffect system

Coeffect algebra \((S, \oplus, \vee, \wedge, \varepsilon)\)

Monoid \((S, \oplus, \varepsilon)\), semi-lattice \((S, \vee)\) and binary \(\wedge\)

For example, for data-flow \((\mathbb{N}, +, \text{max}, \text{min}, 0)\)

\[
\begin{array}{l}
\frac{x : \tau \in \Gamma}{C^\varepsilon \Gamma \vdash x : \tau} \\
\frac{C^r \Gamma \vdash e_1 : C^t \tau_1 \rightarrow \tau_2 \quad C^s \Gamma \vdash e_2 : \tau_1}{C^{r \vee (s \oplus t)} \Gamma \vdash e_1 e_2 : \tau_2} \\
\frac{C^{r \wedge s} (\Gamma, x : \tau_1) \vdash e : \tau_2}{C^r \Gamma \vdash \lambda x . e : C^s \tau_1 \rightarrow \tau_2} \\
\frac{C^r \Gamma \vdash e : \tau}{C^s \Gamma \vdash e : \tau} \quad \text{(if } r \vee s = s)\\
\end{array}
\]

Categorical semantics uses indexed comonads
Conclusions
Conclusions

Unify contextual properties

Liveness, implicit parameters & type classes
Data-flow, cross-compilation, resource usage

Why coeffects are a good idea

Fancy types for plain lambda calculus
Can be embedded in Haskell and used!

[A Notation for Comonads (2012)]
[Efficient and Correct Stencil Computation (2011)]
BACKUP SLIDES
Indexed comonads
Coeffect systems

Unified calculus of context dependence

Examples: liveness, data-flow, implicit parameters
Parameterized by coeffect algebra

Semantics using indexed comonads

Comonad is an indexed comonad
Indexed comonad is not a comonad
With more structure for context passing
Flat coeffect system

**Coeffect algebra** \((S, \oplus, \lor, \land, \varepsilon)\)

Minimal set of required laws
Monoid \((S, \oplus, \varepsilon)\), semi-lattice \((S, \lor)\) and binary \(\land\)

\[
\begin{align*}
\frac{x : \tau \in \Gamma}{\mathcal{C}^\varepsilon \Gamma \vdash x : \tau} & \quad \frac{\mathcal{C}^r \Gamma \vdash e_1 : \mathcal{C}^t \tau_1 \rightarrow \tau_2}{\mathcal{C}^{r \lor (s \oplus t)} \Gamma \vdash e_1 \ e_2 : \tau_2} & \quad \frac{\mathcal{C}^s \Gamma \vdash e_2 : \tau_1}{\mathcal{C}^{r \land s} (\Gamma, x : \tau_1) \vdash e : \tau_2} & \quad \frac{\mathcal{C}^r \Gamma \vdash \lambda x . e : \mathcal{C}^s \tau_1 \rightarrow \tau_2}{\mathcal{C}^s \Gamma \vdash e : \tau (r \leq s)}
\end{align*}
\]
Indexed comonad semantics

Family \( C^r A \) of mappings (data types) with

\[
\begin{align*}
\epsilon & : C^\varepsilon A \to A \\
\tilde{\circ} & : (C^r A \to B) \to (C^s B \to C) \to (C^{r \oplus s} A \to C)
\end{align*}
\]

and for context propagation

\[
\begin{align*}
\text{merge}_{r,s} & : C^r A \times C^s B \to C^{(r \wedge s)} (A \times B) \\
\text{split}_{r,s} & : C^{(r \lor s)} (A \times B) \to C^r A \times C^s B
\end{align*}
\]
Syntactic equational theory

Substitution \((\lambda x. e_2)e_1 \rightarrow e_2[x \leftarrow e_1]\)

Function “values” still need context
Consider call-by-name evaluation

If \(C^r\Gamma \vdash e : \tau\) and \(e \rightarrow e'\) then \(C^r\Gamma \vdash e : \tau\) holds
  - If \(\varepsilon = T\) (for example, liveness)
  - If \(\lor = \land = \oplus\) (for example, implicit params)

Dataflow requires special semantics
Structural coeffects
Structural liveness coeffect

Track liveness of individual variables

Use structural type system

Product $\times$ that mirrors variable structure

$$C^r \times s (\Gamma, x: \tau_1) \vdash e: \tau_2$$

$$C^r \Gamma \vdash \lambda x. e: C^s \tau_1 \to \tau_2$$

Add contraction, exchange, weakening etc.

$$C^r \times s (x: \tau, y: \tau) \vdash e: \tau'$$

$$C^{r \vee s} (z: \tau) \vdash \{z/x\}{z/y}e: \tau'$$

$$C^r \times s (\Gamma_1, \Gamma_2) \vdash e: \tau$$

$$C^{s \times r} (\Gamma_2, \Gamma_1) \vdash e: \tau$$
Semantics of data-flow
Semantics of data-flow

Comonadic semantics
Non-empty list
Over the domain

\[ \text{NEList}\,\tau = \tau \times (1 + \text{NEList}\,\tau) \]

Indexed comonadic semantics
Indexed List type
Index specifies length

\[ C^0\tau = \tau \]
\[ C^1\tau = \tau \times (\tau) \]
\[ C^2\tau = \tau \times (\tau \times \tau) \]
\[ (\ldots) \]