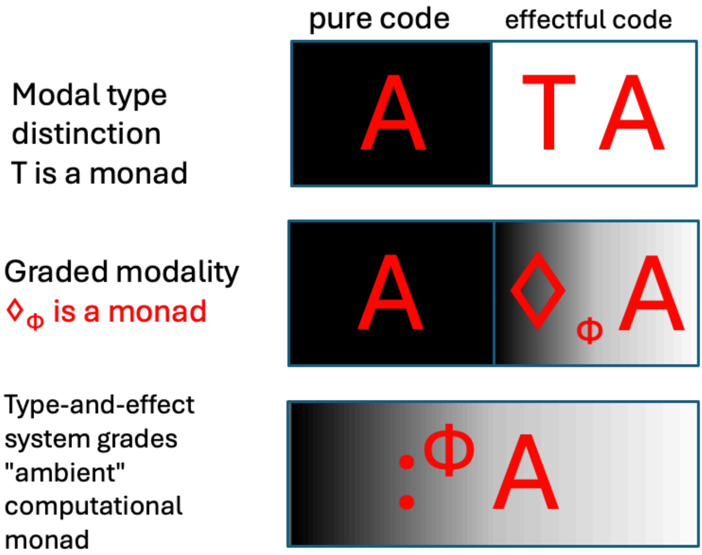
CBPV + effects
CBPV + coeffects

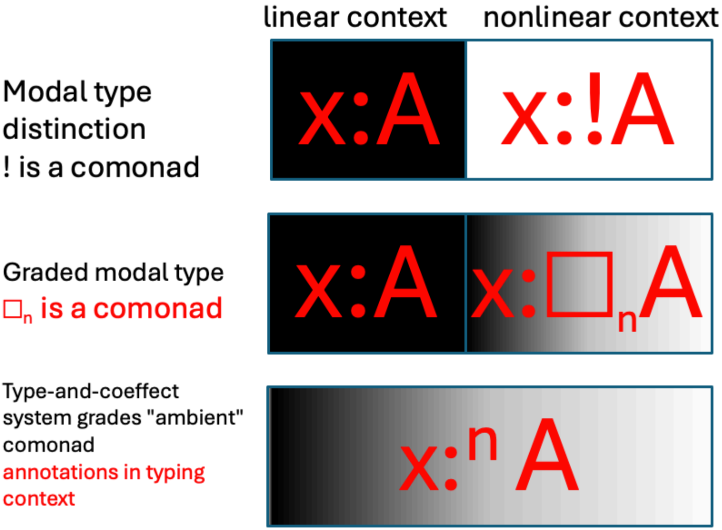
Stephanie Weirich

joint work with Cassia Torczon, Emmanuel Suárez Acevedo, Shubh Agrawal and Joey Velez-Ginorio

March 25, 2024



pure code
effectful code
Modal type distinction
T is a monad
A
T A
Graded modality
$\Diamond_\phi$ is a monad
A
$\Diamond_\phi$ A
Type-and-effect system grades "ambient" computational monad
$:^{\phi}$ A


linear context
nonlinear context
Modal type distinction
! is a comonad
x:A
x:!A
Graded modal type
$\Box_n$ is a comonad
x:A
x:$\Box_n$A
Type-and-coeffect system grades "ambient" comonad
annotations in typing context
x:^{n} A

What is this talk about?

1. Extending CBPV’s type system with effect tracking
2. Extending CBPV’s type system with coeffect tracking
3. Extending CBPV’s type system with effect *and* coeffect tracking (1 slide)

Why CBPV?

What is this talk about?

1. Extending CBPV’s type system with effect tracking
2. Extending CBPV’s type system with coeffect tracking
3. Extending CBPV’s type system with effect *and* coeffect tracking (1 slide)

Why CBPV?

Effects and Coeffects can be tracked in types using graded monads and comonads.
But this requires us to isolate effects and coeffects in dedicated structures.

What is this talk about?

1. Extending CBPV's type system with effect tracking
2. Extending CBPV's type system with coeffect tracking
3. Extending CBPV's type system with effect *and* coeffect tracking (1 slide)

Why CBPV?

Effects and Coeffects can be tracked in types using graded monads and comonads. But this requires us to isolate effects and coeffects in dedicated structures.

But, CBPV already makes the ambient monad and comonad explicit. We just need to grade it!

What is this talk about?

1. Extending CBPV's type system with effect tracking
2. Extending CBPV's type system with coeffect tracking
3. Extending CBPV's type system with effect *and* coeffect tracking (1 slide)

Why CBPV?

Effects and Coeffects can be tracked in types using graded monads and comonads. But this requires us to isolate effects and coeffects in dedicated structures.

But, CBPV already makes the ambient monad and comonad explicit. We just need to grade it!

And, CBPV is a polarized type system: we can observe the duality between effects and coeffects, and understand their interactions with evaluation order.

We can track effects with types

$$\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau$$

An effect annotation ϕ tells us what happens when e is evaluated.

For example,

- To track running time, ϕ is natural number that counts executions of an effectful “tick” term.

We can track effects with types

$$\boxed{\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau}$$

An effect annotation ϕ tells us what happens when e is evaluated.

For example,

- To track running time, ϕ is natural number that counts executions of an effectful "tick" term.
- With algebraic effects, ϕ is the set of operations triggered during computation.

We can track effects with types

$$\boxed{\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau}$$

An effect annotation ϕ tells us what happens when e is evaluated.

For example,

- To track running time, ϕ is natural number that counts executions of an effectful “tick” term.
- With algebraic effects, ϕ is the set of operations triggered during computation.
- To precisely trace logging or other outputs, ϕ is a list of strings.

We can track effects with types

$$\boxed{\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau}$$

lam-eff-unit

$$\frac{}{\Gamma \vdash_{\mathit{eff}} () :^{\varepsilon} \mathbf{unit}}$$

lam-eff-var

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash_{\mathit{eff}} x :^{\varepsilon} \tau}$$

lam-eff-abs

$$\frac{\Gamma, x : \tau_1 \vdash_{\mathit{eff}} e :^{\phi} \tau_2}{\Gamma \vdash_{\mathit{eff}} \lambda x.e :^{\varepsilon} \tau_1 \xrightarrow{\phi} \tau_2}$$

lam-eff-app

$$\frac{\Gamma \vdash_{\mathit{eff}} e_1 :^{\phi_1} \tau_1 \xrightarrow{\phi_3} \tau_2 \quad \Gamma \vdash_{\mathit{eff}} e_2 :^{\phi_2} \tau_1}{\Gamma \vdash_{\mathit{eff}} e_1\, e_2 :^{\phi_1 \cdot \phi_2 \cdot \phi_3} \tau_2}$$

lam-eff-sub

$$\frac{\Gamma \vdash_{\mathit{eff}} e :^{\phi_1} \tau \quad \phi_1 \leq_{\mathit{eff}} \phi_2}{\Gamma \vdash_{\mathit{eff}} e :^{\phi_2} \tau}$$

lam-eff-tick

$$\frac{}{\Gamma \vdash_{\mathit{eff}} \mathbf{tick} :^{\mathit{Tick}} \mathbf{unit}}$$

To track effects throughout the computation, need a *pre-ordered monoid*.

[Lucassen and Gifford 1988, Katsumata 2014]

We can track effects with types

$$\boxed{\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau}$$

lam-eff-unit

$$\frac{}{\Gamma \vdash_{\mathit{eff}} () :^{\varepsilon} \mathbf{unit}}$$

lam-eff-var

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash_{\mathit{eff}} x :^{\varepsilon} \tau}$$

lam-eff-abs

$$\frac{\Gamma, x : \tau_1 \vdash_{\mathit{eff}} e :^{\phi} \tau_2}{\Gamma \vdash_{\mathit{eff}} \lambda x.e :^{\varepsilon} \tau_1 \xrightarrow{\phi} \tau_2}$$

lam-eff-app

$$\frac{\Gamma \vdash_{\mathit{eff}} e_1 :^{\phi_1} \tau_1 \xrightarrow{\phi_3} \tau_2 \qquad \Gamma \vdash_{\mathit{eff}} e_2 :^{\phi_2} \tau_1}{\Gamma \vdash_{\mathit{eff}} e_1\, e_2 :^{\phi_1 \cdot \phi_2 \cdot \phi_3} \tau_2}$$

lam-eff-sub

$$\frac{\Gamma \vdash_{\mathit{eff}} e :^{\phi_1} \tau \qquad \phi_1 \leq_{\mathit{eff}} \phi_2}{\Gamma \vdash_{\mathit{eff}} e :^{\phi_2} \tau}$$

lam-eff-tick

$$\frac{}{\Gamma \vdash_{\mathit{eff}} \mathbf{tick} :^{\mathit{Tick}} \mathbf{unit}}$$

To track effects throughout the computation, need a *pre-ordered monoid*.

These rules are specific to a *call-by-value* semantics.

[Lucassen and Gifford 1988, Katsumata 2014]

We can track effects with types

$$\boxed{\Gamma \vdash_{\mathit{eff}} e :^{\phi} \tau}$$

lam-eff-unit

$$\frac{}{\Gamma \vdash_{\mathit{eff}} () :^{\varepsilon} \mathbf{unit}}$$

lam-eff-var

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash_{\mathit{eff}} x :^{\varepsilon} \tau}$$

lam-eff-abs

$$\frac{\Gamma, x : \tau_1 \vdash_{\mathit{eff}} e :^{\phi} \tau_2}{\Gamma \vdash_{\mathit{eff}} \lambda x.e :^{\varepsilon} \tau_1 \xrightarrow{\phi} \tau_2}$$

lam-eff-app

$$\frac{\Gamma \vdash_{\mathit{eff}} e_1 :^{\phi_1} \tau_1 \xrightarrow{\phi_3} \tau_2 \qquad \Gamma \vdash_{\mathit{eff}} e_2 :^{\phi_2} \tau_1}{\Gamma \vdash_{\mathit{eff}} e_1\, e_2 :^{\phi_1 \cdot \phi_2 \cdot \phi_3} \tau_2}$$

lam-eff-sub

$$\frac{\Gamma \vdash_{\mathit{eff}} e :^{\phi_1} \tau \qquad \phi_1 \leq_{\mathit{eff}} \phi_2}{\Gamma \vdash_{\mathit{eff}} e :^{\phi_2} \tau}$$

lam-eff-tick

$$\frac{}{\Gamma \vdash_{\mathit{eff}} \mathbf{tick} :^{\mathit{Tick}} \mathbf{unit}}$$

To track effects throughout the computation, need a *pre-ordered monoid*.

These rules are specific to a *call-by-value* semantics.

If we had a *call-by-name* semantics, we would need different rules. (And different types!)
[Lucassen and Gifford 1988, Katsumata 2014]

CBPV

CBPV is designed to model effects and subsume both CBV and CBN evaluation.

$$\boxed{\Gamma \vdash V : A} \qquad \boxed{\Gamma \vdash M : B}$$

CBPV is polarized: separate positive and negative types.

$$\begin{array}{llll}
\textit{(value type)} & A & ::= & \texttt{unit} \mid \mathbf{U}\,B \\
\textit{(value)} & V & ::= & x \mid () \mid \{M\} \\
\\
\textit{(computation type)} & B & ::= & A \to B \mid \mathbf{F}\,A \\
\textit{(computation)} & M & ::= & \lambda x.M \mid M\,V \mid V! \\
 & & \mid & \mathbf{return}\,V \mid x \leftarrow M\,\mathbf{in}\,N
\end{array}$$

The type constructors $\mathbf{U}$ and $\mathbf{F}$ form an *adjunction* between values and computations.

- $\mathbf{U}\,\mathbf{F}\,A$ is a monad
- $\mathbf{F}\,\mathbf{U}\,B$ is a comonad

CBPV + effect tracking

Let's extend the CBPV type system to track effects.

$$\boxed{\Gamma \vdash_{\mathit{eff}} V : A} \qquad \boxed{\Gamma \vdash_{\mathit{eff}} M \;:^{\phi}\; B}$$

We'll record latent effects in the thunk type as $\mathbf{U}_\phi\, B$.

(value type)	A	$::=$	$\mathtt{unit} \mid \mathbf{U}_\phi\, B$
(value)	V	$::=$	$x \mid \{M\}$
(computation type)	B	$::=$	$A \to B \mid \mathbf{F}\, A$
(computation)	M	$::=$	$\lambda x.M \mid M\, V \mid V!$
		$\mid$	$\mathbf{return}\, V \mid x \leftarrow M\, \mathbf{in}\, N$

CBPV + effect tracking

Let's extend the CBPV type system to track effects.

$$\boxed{\Gamma \vdash_{\mathit{eff}} V : A} \qquad \boxed{\Gamma \vdash_{\mathit{eff}} M \;:^{\phi}\; B}$$

We'll record latent effects in the thunk type as $\mathbf{U}_{\phi}\, B$.

$$
\begin{array}{llll}
\textit{(value type)} & A & ::= & \texttt{unit} \mid \mathbf{U}_{\phi}\, B \\
\textit{(value)} & V & ::= & x \mid \{M\} \\
 & & & \\
\textit{(computation type)} & B & ::= & A \to B \mid \mathbf{F}\, A \\
\textit{(computation)} & M & ::= & \lambda x.M \mid M\, V \mid V! \\
 & & \mid & \mathbf{return}\, V \mid x \leftarrow M\, \mathbf{in}\, N \mid \mathbf{tick}
\end{array}
$$

and add example effect: **tick**.

$$\frac{\text{eff-tick}}{\Gamma \vdash_{\mathit{eff}} \mathbf{tick} \;:^{\mathit{Tick}}\; \mathbf{F}\,\mathbf{unit}}$$

CBPV with effect tracking

$\boxed{\Gamma \vdash_{\mathit{eff}} V : A}$ *(value effect typing)*

eff-var
$$\frac{x : A \in \Gamma}{\Gamma \vdash_{\mathit{eff}} x : A}$$

eff-thunk
$$\frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi} B}{\Gamma \vdash_{\mathit{eff}} \{M\} : \mathbf{U}_{\phi}\, B}$$

eff-unit
$$\frac{}{\Gamma \vdash_{\mathit{eff}} () : \mathbf{unit}}$$

$\boxed{\Gamma \vdash_{\mathit{eff}} M :^{\phi} B}$ *(computation effect typing)*

eff-abs
$$\frac{\Gamma, x : A \vdash_{\mathit{eff}} M :^{\phi} B}{\Gamma \vdash_{\mathit{eff}} \lambda x.M :^{\phi} A \to B}$$

eff-app
$$\frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi} A \to B \qquad \Gamma \vdash_{\mathit{eff}} V : A}{\Gamma \vdash_{\mathit{eff}} M\, V :^{\phi} B}$$

eff-force
$$\frac{\Gamma \vdash_{\mathit{eff}} V : \mathbf{U}_{\phi}\, B}{\Gamma \vdash_{\mathit{eff}} V! :^{\phi} B}$$

eff-ret
$$\frac{\Gamma \vdash_{\mathit{eff}} V : A}{\Gamma \vdash_{\mathit{eff}} \mathbf{return}\, V :^{\varepsilon} \mathbf{F}\, A}$$

eff-letin
$$\frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi_1} \mathbf{F}\, A \qquad \Gamma, x : A \vdash_{\mathit{eff}} N :^{\phi_2} B}{\Gamma \vdash_{\mathit{eff}} x \leftarrow M\, \mathbf{in}\, N :^{\phi_1 \cdot \phi_2} B}$$

eff-sub
$$\frac{\Gamma \vdash_{\mathit{eff}} M :^{\phi_1} B \qquad \phi_1 \leq_{\mathit{eff}} \phi_2}{\Gamma \vdash_{\mathit{eff}} M :^{\phi_2} B}$$

[Kammar and Plotkin 2012, Kammar, Lindley, Oury 2013]

Effect soundness

Key result of type system is *effect soundness*: the type system bounds effects that could occur at runtime.

Big-step operational semantics: $\boxed{\rho \vdash_{\mathit{eff}} M \Downarrow T \,\#\, \phi}$ counts ticks while evaluating computation M to terminal T.

Theorem

If $\varnothing \vdash_{\mathit{eff}} M \;:^{\phi}\; \mathbf{F}\, A$ *and* $\emptyset \vdash_{\mathit{eff}} M \Downarrow \mathbf{return}\, W \,\#\, \phi'$ *then* $\phi' \leq_{\mathit{eff}} \phi$.

Proof.

Uses logical relations. □

What about coeffects?

Coeffects track how input values contribute to the output result.

- Bounded linear types
- Whether functions use their arguments
- Differential privacy (how sensitive are function outputs to their inputs)
- Whether functions are monotonic
- Information-flow
- …

(Technically, these are examples of *structured* coeffects.)

What about coeffects?

Coeffects track how input values contribute to the output result.

- Bounded linear types
- Whether functions use their arguments
- Differential privacy (how sensitive are function outputs to their inputs)
- Whether functions are monotonic
- Information-flow
- ...

(Technically, these are examples of *structured* coeffects.)

We mark variables in the context with coeffects q (short for quantity).

Coeffect examples

- For *bounded linear types*, we can use natural numbers.

$$x :^{1} \mathbf{int}, y :^{3} \mathbf{int}, z :^{0} \mathbf{int} \vdash_{coeff} x + (y + y) : \mathbf{int}$$

Coeffect examples

- For *bounded linear types*, we can use natural numbers.

$$x :^{1} \mathbf{int}\,, y :^{3} \mathbf{int}\,, z :^{0} \mathbf{int} \vdash_{coeff} x + (y + y) : \mathbf{int}$$

- For *relevance analysis*, 0 marks arguments that are not used and ω marks arguments that *may* be used.

$$x :^{\omega} \mathbf{int}\,, y :^{\omega} \mathbf{int}\,, z :^{0} \mathbf{int} \vdash_{coeff} x + (y + y) : \mathbf{int}$$

Coeffect examples

- For *bounded linear types*, we can use natural numbers.

$$x :^{1} \mathbf{int}\,, y :^{3} \mathbf{int}\,, z :^{0} \mathbf{int} \vdash_{\mathit{coeff}} x + (y + y) : \mathbf{int}$$

- For *relevance analysis*, 0 marks arguments that are not used and ω marks arguments that *may* be used.

$$x :^{\omega} \mathbf{int}\,, y :^{\omega} \mathbf{int}\,, z :^{0} \mathbf{int} \vdash_{\mathit{coeff}} x + (y + y) : \mathbf{int}$$

- For *data flow caching*, we want to provide access to prior values during streaming computation.

$$x :^{1} \mathbf{int}\,, y :^{0} \mathbf{int} \vdash_{\mathit{coeff}} (\mathbf{prev}\, x) + x + y : \mathbf{int}$$

We can use natural numbers that track how many previous values are required.

We can track coeffects with types

Context comes with a list of coeffects for every variable.

$$\gamma ::= \varnothing \mid \gamma , q$$

We use notation to extend both at once:

$$\gamma \cdot \Gamma , x :^q \tau = (\gamma , q) \cdot (\Gamma , x : \tau)$$

$\boxed{\gamma \cdot \Gamma \vdash_{coeff} e : \tau}$

lam-coeff-var

$$\frac{}{\overline{0} \cdot \Gamma_1 , x :^1 \tau , \overline{0} \cdot \Gamma_2 \vdash_{coeff} x : \tau}$$

lam-coeff-abs

$$\frac{\gamma \cdot \Gamma , (x :^q \tau_1) \vdash_{coeff} e : \tau_2}{\gamma \cdot \Gamma \vdash_{coeff} \lambda^q x.e : \tau_1^q \to \tau_2}$$

lam-coeff-app

$$\frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e_1 : \tau_1^q \to \tau_2 \qquad \gamma_2 \cdot \Gamma \vdash_{coeff} e_2 : \tau_1 \qquad \gamma \equiv \gamma_1 + q \cdot \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} e_1\, e_2 : \tau_2}$$

lam-coeff-sub

$$\frac{\gamma_1 \cdot \Gamma \vdash_{coeff} e : \tau \qquad \gamma_2 \leq_{co} \gamma_1}{\gamma_2 \cdot \Gamma \vdash_{coeff} e : \tau}$$

This is for a call-by-name language [Abel and Bernardy 2020, Choudhury et al. 2021].

We can track coeffects with types

Context comes with a list of coeffects for every variable.

$$\gamma ::= \varnothing \mid \gamma\,,q$$

We use notation to extend both at once:

$$\gamma \cdot \Gamma\,, x :^{q} \tau = (\gamma\,, q) \cdot (\Gamma\,, x : \tau)$$

$\boxed{\gamma \cdot \Gamma \vdash_{\mathit{coeff}} e : \tau}$

lam-coeff-var

$$\frac{}{\overline{0} \cdot \Gamma_1\,, x :^{1} \tau\,, \overline{0} \cdot \Gamma_2 \vdash_{\mathit{coeff}} x : \tau}$$

lam-coeff-abs

$$\frac{\gamma \cdot \Gamma\,, (x :^{q} \tau_1) \vdash_{\mathit{coeff}} e : \tau_2}{\gamma \cdot \Gamma \vdash_{\mathit{coeff}} \lambda^{q} x.e : \tau_1^{q} \to \tau_2}$$

lam-coeff-appv

$$\frac{\gamma_1 \cdot \Gamma \vdash_{\mathit{coeff}} e_1 : \tau_1^{q} \to \tau_2 \qquad \gamma_2 \cdot \Gamma \vdash_{\mathit{coeff}} e_2 : \tau_1 \qquad \gamma \equiv \gamma_1 + (q \wedge 1 \cdot \gamma_2)}{\gamma \cdot \Gamma \vdash_{\mathit{coeff}} e_1\, e_2 : \tau_2}$$

lam-coeff-sub

$$\frac{\gamma_1 \cdot \Gamma \vdash_{\mathit{coeff}} e : \tau \qquad \gamma_2 \leq_{co} \gamma_1}{\gamma_2 \cdot \Gamma \vdash_{\mathit{coeff}} e : \tau}$$

Call-by-value language forces usage in application rule [Gavazzo 2018].

CBPV with coeffects

$$\boxed{\gamma \cdot \Gamma \vdash_{coeff} V : A}$$

(Value typing)

coeff-var

$$\frac{}{\overline{0} \cdot \Gamma_1 , x :^1 A , \overline{0} \cdot \Gamma_2 \vdash_{coeff} x : A}$$

coeff-unit

$$\frac{}{\overline{0} \cdot \Gamma \vdash_{coeff} () : \mathbf{unit}}$$

coeff-thunk

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} M : B}{\gamma \cdot \Gamma \vdash_{coeff} \{M\} : \mathbf{U}\, B}$$

$$\boxed{\gamma \cdot \Gamma \vdash_{coeff} M : B}$$

(Computation typing)

coeff-abs

$$\frac{\gamma \cdot \Gamma , x :^q A \vdash_{coeff} M : B}{\gamma \cdot \Gamma \vdash_{coeff} \lambda x^q . M : A^q \to B}$$

coeff-app

$$\frac{\gamma_1 \cdot \Gamma \vdash_{coeff} M : A^q \to B \qquad \gamma_2 \cdot \Gamma \vdash_{coeff} V : A \qquad \gamma \equiv \gamma_1 + (q \cdot \gamma_2)}{\gamma \cdot \Gamma \vdash_{coeff} M\, V : B}$$

coeff-force

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} V : \mathbf{U}\, B}{\gamma \cdot \Gamma \vdash_{coeff} V! : B}$$

coeff-ret

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} V : A}{q \cdot \gamma \cdot \Gamma \vdash_{coeff} \mathbf{return}_q\ V : \mathbf{F}_q\ A}$$

coeff-letin-v

$$\frac{\gamma_1 \cdot \Gamma \vdash_{coeff} M : \mathbf{F}_{q_1}\ A \qquad \gamma_2 \cdot \Gamma , x :^{q_1 \cdot q_2'} A \vdash_{coeff} N : B \qquad \gamma \equiv (q_2' \cdot \gamma_1) + \gamma_2 \qquad q_2' = q_2 \wedge 1}{\gamma \cdot \Gamma \vdash_{coeff} x \leftarrow^{q_2} M\ \mathbf{in}\ N : B}$$

(+subrules)

Coeffect soundness

To show coeffect soundness, we define an environment-based operational semantics that counts uses of variables.

$\boxed{\gamma \cdot \rho \vdash_{\mathit{coeff}} V \Downarrow W}$ *(Value rules)*

eval-coeff-val-var

$$\frac{}{\overline{0}_1 \cdot \rho_1\,,\ x \mapsto^1 W,\ \overline{0}_2 \cdot \rho_2 \vdash_{\mathit{coeff}} x \Downarrow W}$$

eval-coeff-val-unit

$$\frac{}{\overline{0} \cdot \rho \vdash_{\mathit{coeff}} () \Downarrow ()}$$

eval-coeff-val-thunk

$$\frac{}{\gamma \cdot \rho \vdash_{\mathit{coeff}} \{M\} \Downarrow \mathbf{clo}(\gamma \cdot \rho, \{M\})}$$

eval-coeff-val-vsub

$$\frac{\gamma_1 \cdot \rho \vdash_{\mathit{coeff}} V \Downarrow W \qquad \gamma_2 \leq_{co} \gamma_1}{\gamma_2 \cdot \rho \vdash_{\mathit{coeff}} V \Downarrow W}$$

Lemma (Coeffect soundness)

1. *If* $\gamma \cdot \Gamma \vdash_{\mathit{coeff}} V : A$ *then* $\gamma \cdot \rho \vdash_{\mathit{coeff}} V \Downarrow W$.
2. *If* $\gamma \cdot \Gamma \vdash_{\mathit{coeff}} M : B$ *then* $\gamma \cdot \rho \vdash_{\mathit{coeff}} M \Downarrow T$.

A strange semantics?

Although sound, this semantics doesn't model *resource usage*.

$$\boxed{\gamma \cdot \rho \vdash_{\mathit{coeff}} M \Downarrow T}$$

(Computation rules)

eval-coeff-comp-abs

$$\frac{}{\gamma \cdot \rho \vdash_{\mathit{coeff}} \lambda x^{q}.M \Downarrow \mathbf{clo}(\gamma \cdot \rho, \lambda x^{q}.M)}$$

eval-coeff-comp-app-abs

$$\frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{\mathit{coeff}} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \lambda x^{q}.M') \\ \gamma_2 \cdot \rho \vdash_{\mathit{coeff}} V \Downarrow W \\ \gamma' \cdot \rho',\ x \mapsto^{q} W \vdash_{\mathit{coeff}} M' \Downarrow T \\ \gamma \equiv \gamma_1 + q \cdot \gamma_2\end{array}}{\gamma \cdot \rho \vdash_{\mathit{coeff}} M\, V \Downarrow T}$$

Application rule "invents" resources when q is zero!

A strange semantics?

Although sound, this semantics doesn't model *resource usage*.

$$\boxed{\gamma \cdot \rho \vdash_{coeff} M \Downarrow T}$$

(Computation rules)

eval-coeff-comp-abs

$$\frac{}{\gamma \cdot \rho \vdash_{coeff} \lambda x^q.M \Downarrow \mathbf{clo}(\gamma \cdot \rho, \lambda x^q.M)}$$

eval-coeff-comp-app-abs

$$\frac{\gamma_1 \cdot \rho \vdash_{coeff} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \lambda x^q.M') \quad \gamma_2 \cdot \rho \vdash_{coeff} V \Downarrow W \quad \gamma' \cdot \rho',\ x \mapsto^q W \vdash_{coeff} M' \Downarrow T \quad \gamma \equiv \gamma_1 + q \cdot \gamma_2}{\gamma \cdot \rho \vdash_{coeff} M\, V \Downarrow T}$$

Application rule "invents" resources when q is zero!

We can type this judgement, which says that *x* does not contribute to the final result.

$$x :^0 A \vdash_{coeff} (\lambda y^0.\mathbf{return}\,())\, x : \mathbf{F}\,\mathbf{unit}$$

Resource accounting semantics

Can discard unused values, without accounting for their resources

$\boxed{\gamma \cdot \rho \vdash_{coeff} M \Downarrow T}$

(Computation rules)

eval-lin-comp-app-abs

$$\frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{lin} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \lambda x^q.M') \\ \gamma_2 \cdot \rho \vdash_{lin} V \Downarrow W \\ (\gamma' \cdot \rho'), (x \mapsto^q W) \vdash_{lin} M' \Downarrow T \\ \gamma \equiv \gamma_1 + q \cdot \gamma_2 \\ q \neq 0\end{array}}{\gamma \cdot \rho \vdash_{lin} M\, V \Downarrow T}$$

eval-lin-comp-app-abs-zero

$$\frac{\begin{array}{c}\gamma \cdot \rho \vdash_{lin} M \Downarrow \mathbf{clo}(\gamma' \cdot \rho', \lambda x^0.M') \\ (\gamma' \cdot \rho'), (x \mapsto^0 \lightning) \vdash_{lin} M' \Downarrow T\end{array}}{\gamma \cdot \rho \vdash_{lin} M\, V \Downarrow T}$$

eval-lin-comp-return

$$\frac{\begin{array}{c}\gamma' \cdot \rho \vdash_{lin} V \Downarrow W \\ \gamma \equiv q \cdot \gamma' \qquad q \neq 0\end{array}}{\gamma \cdot \rho \vdash_{lin} \mathbf{return}_q\; V \Downarrow \mathbf{return}_q W}$$

eval-lin-comp-ret-zero

$$\frac{}{\overline{0} \cdot \rho \vdash_{lin} \mathbf{return}_0\; V \Downarrow \mathbf{return}_0\; \lightning}$$

Cannot discard effectful computations

$\boxed{\gamma \cdot \rho \vdash_{coeff} M \Downarrow T}$ *(Computation rules)*

eval-lin-comp-letin-ret

$$\frac{\begin{array}{c}\gamma_1 \cdot \rho \vdash_{lin} M \Downarrow \mathbf{return}_{q_1} W \\ \gamma_2 \cdot \rho,\ x \mapsto^{q_1 \cdot q_2'} W \vdash_{lin} N \Downarrow T \\ \gamma \equiv q_2' \cdot \gamma_1 + \gamma_2 \\ q_2' = q_2 \wedge 1\end{array}}{\gamma \cdot \rho \vdash_{lin} x \leftarrow^{q_2} M \,\mathbf{in}\, N \Downarrow T}$$

Combined effects and co-effects

Can discard computations that are **pure**.

Let's track effects and coeffects together.

$\boxed{\gamma \cdot \Gamma \vdash_{\mathit{full}} M :^{\phi} B}$ *(Typing rule)*

full-letin-zero

$$\frac{\gamma_1 \cdot \Gamma \vdash_{\mathit{full}} M_1 :^{\varepsilon} \mathbf{F}_{q_1}\, A \qquad \gamma_2 \cdot \Gamma, x :^{0} A \vdash_{\mathit{full}} M_2 :^{\phi} B}{\gamma_2 \cdot \Gamma \vdash_{\mathit{full}} x \leftarrow^{0} M_1 \ \mathbf{in}\ M_2 :^{\phi} B}$$

$\boxed{\gamma \cdot \rho \vdash_{\mathit{full}} M \Downarrow T \,\#\, \phi}$ *(Evaluation rule)*

eval-full-comp-letin-zero

$$\frac{\gamma \cdot \rho,\ x \mapsto^{q_1 \cdot q_2'} \lightning \vdash_{\mathit{full}} N \Downarrow T \,\#\, \phi}{\gamma \cdot \rho \vdash_{\mathit{full}} x \leftarrow^{0} M \ \mathbf{in}\ N \Downarrow T \,\#\, \phi}$$

Summary

- Augmented CBPV with effect and coeffect tracking.
- Effects describe computations, so annotate thunk type $\mathbf{U}_{\phi}\ B$.
  Coeffects describe values, so annotate returner type $\mathbf{F}_{q}\ A$
- $\mathbf{U}_{\phi}\ \mathbf{F} A$ is a graded monad in the value language.
  $\mathbf{F}_{q}\ \mathbf{U} B$ is a graded comonad in the computation language.
- Showed effect and coeffect soundness, even in the presence of a semantics that tracks resource usage.
- In the paper: Standard CBV and CBN translations are type, effect, coeffect preserving.
  Explains restrictions found in some CBV coeffect type systems. (CBN translation does not require the use of “letin”.)
- Proofs mechanized in Coq.

CBV Translation (Effects!)

We can translate type-and-effect CBV to effect-tracking CBPV. The standard translation just works.

$$
\begin{aligned}
[\![\mathbf{unit}]\!]_{\mathsf{v}} &= \mathbf{unit} \\
[\![\tau_1 \xrightarrow{\phi} \tau_2]\!]_{\mathsf{v}} &= \mathbf{U}_{\phi}\,([\![\tau_1]\!]_{\mathsf{v}} \to \mathbf{F}\,[\![\tau_2]\!]_{\mathsf{v}})
\end{aligned}
$$

$$
\begin{aligned}
[\![()]\!]_{\mathsf{v}} &= \mathbf{return}\,() \\
[\![x]\!]_{\mathsf{v}} &= \mathbf{return}\,x \\
[\![\lambda x.e]\!]_{\mathsf{v}} &= \mathbf{return}\,\{\lambda x.[\![e]\!]_{\mathsf{v}}\} \\
[\![e_1\,e_2]\!]_{\mathsf{v}} &= x \leftarrow [\![e_1]\!]_{\mathsf{v}}\,\mathbf{in}\,y \leftarrow [\![e_2]\!]_{\mathsf{v}}\,\mathbf{in}\,x!\,y \\
[\![\mathbf{tick}]\!]_{\mathsf{v}} &= \mathbf{tick}
\end{aligned}
$$

Theorem (Translation preserves types-and-effects)

If $\Gamma \vdash_{eff} e :^{\phi} \tau$ *then* $[\![\Gamma]\!]_{\mathsf{v}} \vdash_{eff} [\![e]\!]_{\mathsf{v}} :^{\phi} \mathbf{F}\,[\![\tau]\!]_{\mathsf{v}}$.

CBN translation (Graded Monads!)

We can also use the CBN translation for a source language with graded monads.

However, while $\mathbf{U}\,\mathbf{F}\,A$ is a monad in CBPV, it is awkward to access.

Theorem (Translation preserves types)

If $\Gamma \vdash_{mon} e : \tau$ *then* $[\![\Gamma]\!]_{\mathsf{n}} \vdash_{eff} [\![e]\!]_{\mathsf{n}} \;:^{\varepsilon}\; [\![\tau]\!]_{\mathsf{n}}$.

CBN translation (Graded Monads!)

We can also use the CBN translation for a source language with graded monads.

However, while $\mathbf{U}\,\mathbf{F}\,A$ is a monad in CBPV, it is awkward to access.

$$[\![\mathbf{T}_{\phi}\ \tau]\!]_{\mathsf{n}} \quad = \mathbf{F}\,\mathbf{U}_{\phi}\ \mathbf{F}\,\mathbf{U}_{\varepsilon}\ [\![\tau]\!]_{\mathsf{n}}$$

Theorem (Translation preserves types)

If $\Gamma \vdash_{mon} e : \tau$ *then* $[\![\Gamma]\!]_{\mathsf{n}} \vdash_{eff} [\![e]\!]_{\mathsf{n}} \;:^{\varepsilon}\; [\![\tau]\!]_{\mathsf{n}}$.

CBN translation (Graded Monads!)

We can also use the CBN translation for a source language with graded monads.

However, while $\mathbf{U}\,\mathbf{F}\,A$ is a monad in CBPV, it is awkward to access.

$$\llbracket \mathbf{T}_{\phi}\ \tau \rrbracket_{\mathsf{n}} = \mathbf{F}\,\mathbf{U}_{\phi}\ \mathbf{F}\,\mathbf{U}_{\varepsilon}\ \llbracket \tau \rrbracket_{\mathsf{n}}$$

$$\llbracket \mathbf{return}\, e \rrbracket_{\mathsf{n}} = \mathbf{return}\,\{\mathbf{return}\,\{\llbracket e \rrbracket_{\mathsf{n}}\}\}$$

Theorem (Translation preserves types)

If $\Gamma \vdash_{mon} e : \tau$ *then* $\llbracket \Gamma \rrbracket_{\mathsf{n}} \vdash_{eff} \llbracket e \rrbracket_{\mathsf{n}} \ :^{\varepsilon}\ \llbracket \tau \rrbracket_{\mathsf{n}}$.

CBN translation (Graded Monads!)

We can also use the CBN translation for a source language with graded monads.

However, while $\mathbf{U}\,\mathbf{F}\,A$ is a monad in CBPV, it is awkward to access.

$$\begin{aligned}
[\![\mathbf{T}_{\phi}\ \tau]\!]_{\mathsf{n}} &= \mathbf{F}\,\mathbf{U}_{\phi}\ \mathbf{F}\,\mathbf{U}_{\varepsilon}\ [\![\tau]\!]_{\mathsf{n}} \\
\\
[\![\mathbf{return}\,e]\!]_{\mathsf{n}} &= \mathbf{return}\,\{\mathbf{return}\,\{[\![e]\!]_{\mathsf{n}}\}\} \\
[\![\mathbf{bind}\,x = e_1\,\mathbf{in}\,e_2]\!]_{\mathsf{n}} &= \mathbf{return}\,\{y \leftarrow [\![e_1]\!]_{\mathsf{n}}\,\mathbf{in}\,x \leftarrow y!\,\mathbf{in}\,z \leftarrow [\![e_2]\!]_{\mathsf{n}}\,\mathbf{in}\,z!\}
\end{aligned}$$

Theorem (Translation preserves types)

If $\Gamma \vdash_{mon} e : \tau$ *then* $[\![\Gamma]\!]_{\mathsf{n}} \vdash_{eff} [\![e]\!]_{\mathsf{n}} :^{\varepsilon} [\![\tau]\!]_{\mathsf{n}}$.

CBN translation (Graded Monads!)

We can also use the CBN translation for a source language with graded monads.

However, while $\mathbf{U}\,\mathbf{F}\,A$ is a monad in CBPV, it is awkward to access.

$$
\begin{aligned}
[\![\mathbf{T}_{\phi}\ \tau]\!]_{\mathsf{n}} &= \mathbf{F}\,\mathbf{U}_{\phi}\ \mathbf{F}\,\mathbf{U}_{\varepsilon}\ [\![\tau]\!]_{\mathsf{n}} \\
\\
[\![\mathbf{return}\,e]\!]_{\mathsf{n}} &= \mathbf{return}\,\{\mathbf{return}\,\{[\![e]\!]_{\mathsf{n}}\}\} \\
[\![\mathbf{bind}\,x = e_1\ \mathbf{in}\,e_2]\!]_{\mathsf{n}} &= \mathbf{return}\,\{y \leftarrow [\![e_1]\!]_{\mathsf{n}}\ \mathbf{in}\,x \leftarrow y!\ \mathbf{in}\,z \leftarrow [\![e_2]\!]_{\mathsf{n}}\ \mathbf{in}\,z!\} \\
[\![\mathbf{tick}]\!]_{\mathsf{n}} &= \mathbf{return}\,\{x \leftarrow \mathbf{tick}\ \mathbf{in}\,\mathbf{return}\,\{\mathbf{return}\,x\}\}
\end{aligned}
$$

Theorem (Translation preserves types)

If $\Gamma \vdash_{mon} e : \tau$ *then* $[\![\Gamma]\!]_{\mathsf{n}} \vdash_{eff} [\![e]\!]_{\mathsf{n}}\ :^{\varepsilon}\ [\![\tau]\!]_{\mathsf{n}}$.

CBN translation (coeffects!)

Standard translation of CBN to CBPV just works.

$$
\begin{aligned}
[\![\mathbf{unit}]\!]_{\mathsf{n}} &= \mathbf{F}_{1}\ \mathbf{unit} \\
[\![\tau_1^{q} \to \tau_2]\!]_{\mathsf{n}} &= (\mathbf{U}\,[\![\tau_1]\!]_{\mathsf{n}})^{q} \to [\![\tau_2]\!]_{\mathsf{n}} \\
\\
[\![\Gamma\,, x : \tau]\!]_{\mathsf{n}} &= [\![\Gamma]\!]_{\mathsf{n}}\,, x : \mathbf{U}\,[\![\tau]\!]_{\mathsf{n}} \\
\\
[\![()]\!]_{\mathsf{n}} &= \mathbf{return}_{1}\,() \\
[\![x]\!]_{\mathsf{n}} &= x! \\
[\![\lambda x.e]\!]_{\mathsf{n}} &= \lambda x.[\![e]\!]_{\mathsf{n}} \\
[\![e_1\, e_2]\!]_{\mathsf{n}} &= [\![e_1]\!]_{\mathsf{n}}\,\{[\![e_2]\!]_{\mathsf{n}}\}
\end{aligned}
$$

Theorem (Translation preserves types and coeffects)

If $\gamma \cdot \Gamma \vdash_{coeff} e : \tau$ *then* $\gamma \cdot [\![\Gamma]\!]_{\mathsf{n}} \vdash_{coeff} [\![e]\!]_{\mathsf{n}} : [\![\tau]\!]_{\mathsf{n}}$.

Interlude: Two kinds of products

CBPV has two forms of products: pairs of values and pairs of computations. The former are eliminated with pattern matching and the latter by projection.
Linear logic has two forms of conjunction: *additive* $\&$ (aka with) and *multiplicative* products $\otimes$ (aka tensor).
The former shares resources during construction, the latter does not.

coeff-pair

$$\frac{\gamma_1 \cdot \Gamma \vdash_{coeff} V_1 : A_1 \qquad \gamma_2 \cdot \Gamma \vdash_{coeff} V_2 : A_2}{\gamma_1 + \gamma_2 \cdot \Gamma \vdash_{coeff} (V_1, V_2) : A_1 \times A_2}$$

coeff-split

$$\frac{\gamma_1 \cdot \Gamma \vdash_{coeff} V : A_1 \times A_2 \qquad \gamma_2 \cdot \Gamma, x_1 :^q A_1, x_2 :^q A_2 \vdash_{coeff} N : B \qquad \gamma \equiv (q \cdot \gamma_1) + \gamma_2}{\gamma \cdot \Gamma \vdash_{coeff} \mathbf{case}_q\ V\ \mathbf{of}\ (x_1, x_2)\ \rightarrow\ N : B}$$

coeff-cpair

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} M_1 : B_1 \qquad \gamma \cdot \Gamma \vdash_{coeff} M_2 : B_2}{\gamma \cdot \Gamma \vdash_{coeff} \langle M_1, M_2 \rangle : B_1 \,\&\, B_2}$$

coeff-fst

$$\frac{\gamma \cdot \Gamma \vdash_{coeff} M : B_1 \,\&\, B_2}{\gamma \cdot \Gamma \vdash_{coeff} M.1 : B_1}$$

Interlude: Four kinds of products

But it doesn't have to be this way.
Can have "with" products in the value language, eliminated by projection.

coeff-vwith

$$\frac{\gamma \cdot \Gamma \vdash_{\mathit{coeff}} V_1 : A_1 \qquad \gamma \cdot \Gamma \vdash_{\mathit{coeff}} V_2 : A_2}{\gamma \cdot \Gamma \vdash_{\mathit{coeff}} \langle V_1, V_2 \rangle : A_1 \,\&\, A_2}$$

coeff-vfst

$$\frac{\gamma \cdot \Gamma \vdash_{\mathit{coeff}} V : A_1 \,\&\, A_2}{\gamma \cdot \Gamma \vdash_{\mathit{coeff}} V.1 : A_1}$$

Can have tensor products in the computation language, eliminated by pattern matching.

coeff-ctensor

$$\frac{\gamma_1 \cdot \Gamma \vdash_{\mathit{coeff}} M_1 : B_1 \qquad \gamma_2 \cdot \Gamma \vdash_{\mathit{coeff}} M_2 : B_2}{\gamma_1 + \gamma_2 \cdot \Gamma \vdash_{\mathit{coeff}} (M_1, M_2) : B_1 \times B_2}$$

coeff-csplit

$$\frac{\gamma_1 \cdot \Gamma \vdash_{\mathit{coeff}} M : B_1 \times B_2 \qquad \gamma_2 \cdot \Gamma, x_1 :^q \mathbf{U}\, B_1, x_2 :^q \mathbf{U}\, B_2 \vdash_{\mathit{coeff}} N : B \qquad \gamma \equiv q \cdot \gamma_1 + \gamma_2}{\gamma \cdot \Gamma \vdash_{\mathit{coeff}} \mathbf{case}_q\, M\, \mathbf{of}\, (x_1, x_2) \rightarrow N : B}$$