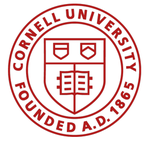
Effects and Coeffects in Call-By-Push-Value

Cassia Torczon, Emmanuel Suárez Acevedo, Shubh Agrawal,
Joey Velez-Ginorio, Stephanie Weirich

OOPSLA 2024



Cornell University

Northeastern University

Dead code elimination

```
1| let x = "hello" in
2| print "world"
```

Dead code elimination

```
1| let x = "hello" in
2| print "world"
```

can the first line be removed?

Dead code elimination

```
1| let x = print "hello" in
2| print "world"
```

Dead code elimination

```
1| let x = e1 in
2| e2
```

Dead code elimination

```
1| let x = e1 in
2| e2
```

Dead code elimination

```
1| let x = e1 in
2| e2
```

is the code bound to x pure?

Dead code elimination

```
1| let x = e1 in
2| e2
```

is the code bound to x pure?

is x unused in the code that follows?

Effects

$$x : \text{string}, y : \text{nat} \vdash \mathbf{print}\ x : \ \text{unit}$$

Effects

$$x : \text{string}, y : \text{nat} \vdash \mathbf{print}\ x : \overset{\text{Output}}{\phantom{\text{unit}}}\ \text{unit}$$

Effects

$$\Gamma \vdash e :^{\phi} \tau$$

Effects

IO effects

$$\Gamma \vdash e :^{\phi} \tau$$

Effects

IO effects

Running time

$$\Gamma \vdash e :^{\phi} \tau$$

Effects

IO effects

Running time

$$\Gamma \vdash e :^{\phi} \tau$$

Exceptions

… and Coeffects

$$x : \text{string}, y : \text{nat} \vdash \mathbf{print}\ x : \text{unit}$$

… and Coeffects

$$x :^{1} \text{string}, y :^{0} \text{nat} \vdash \mathbf{print}\ x : \text{unit}$$

… and Coeffects

$$x_1 :^{q_1} \tau_1, x_2 :^{q_2} \tau_2 \vdash e : \tau$$

… and Coeffects

$$\gamma \cdot \Gamma \vdash e : \tau$$

… and Coeffects

Relevance

$$\gamma \cdot \Gamma \vdash e : \tau$$

… and Coeffects

Relevance

Resources

$$\gamma \cdot \Gamma \vdash e : \tau$$

… and Coeffects

Relevance

Resources

$$\gamma \cdot \Gamma \vdash e : \tau$$

Differential privacy

… and Coeffects

Relevance

Resources

$$\gamma \cdot \Gamma \vdash e : \tau$$

Differential privacy

Information flow

Effects and Coeffects

```
1| let x = e1 in
2| e2
```

Effect: is the code bound to x pure?

Coeffect: is x unused in the code that follows?

Statically tracking effects

```
1| let x = print "hello" in
2| x; x
```

Statically tracking effects

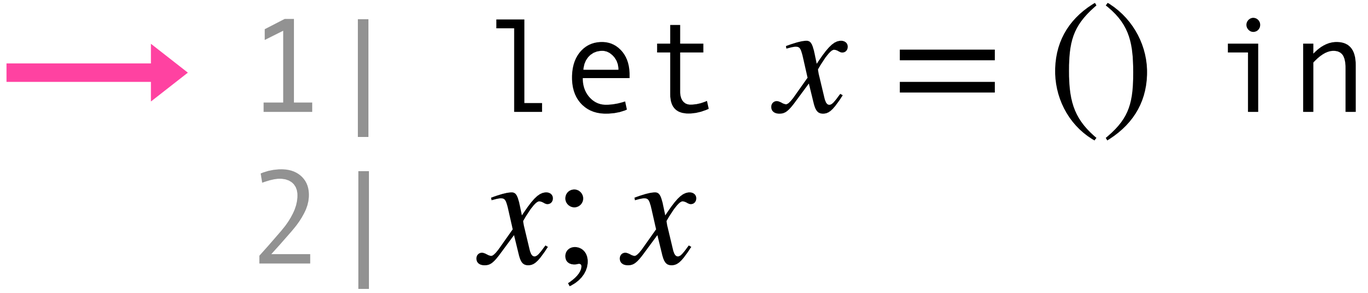
```
1| let x = print "hello" in
2| x; x
```

Expected effect?

Statically tracking effects

Expected effect?

Statically tracking effects



Expected effect?

```
hello
```

Statically tracking effects

$$x \mapsto ()$$

```
1| let x = print "hello" in
2|  x; x
```

Expected effect?

```
hello
```

Statically tracking effects

$$x \mapsto ()$$

```
1| let x = print "hello" in
2|  ();()
```

Expected effect?

```
hello
```

Statically tracking effects

```
1| let x = print "hello" in
2| x; x
```

Expected effect?

Statically tracking effects

Expected effect?

Statically tracking effects

```
x ↦ print "hello"
1| let x = print "hello" in
2| x; x
```

Expected effect?

Statically tracking effects

```
x ↦ print "hello"

1| let x = print "hello" in
2| print "hello";print "hello"
```

Expected effect?

Statically tracking effects

```
x ↦ print "hello"

1| let x = print "hello" in
2| ();()
```

Expected effect?

```
hello
hello
```

Statically tracking effects

Intermediate representation

Statically tracking effects

Intermediate representation

Explicit evaluation order

… in Call-By-Push-Value (CBPV)

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

… in Call-By-Push-Value (CBPV)

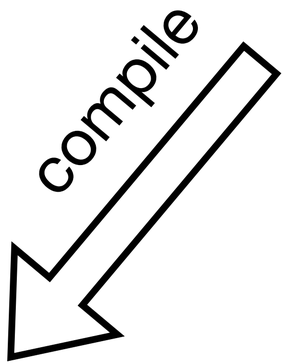
A subsuming paradigm

```
1| let x = print "hello" in
2| x; x
```

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

```
1| let x = print "hello" in
2| x; x
```

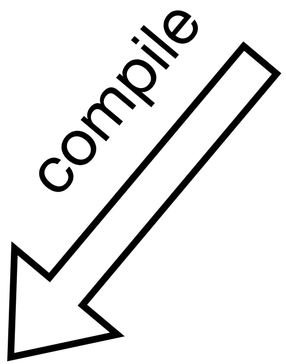


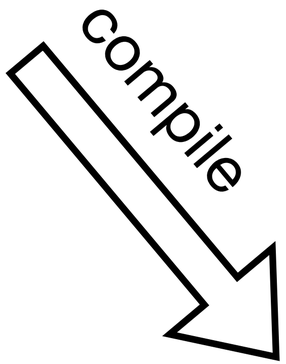

```
hello
```

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

```
1| let x = print "hello" in
2| x; x
```






```
hello
```

```
hello
hello
```

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

Expression

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

Value	Computation

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

Value	Computation
“is”	

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

Value

“is”

A string is a value

Computation

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

Value

“is”

A string is a value

Computation

“does”

… in Call-By-Push-Value (CBPV)

A subsuming paradigm

Value

“is”

A string is a value

Computation

“does”

Printing a string is a computation

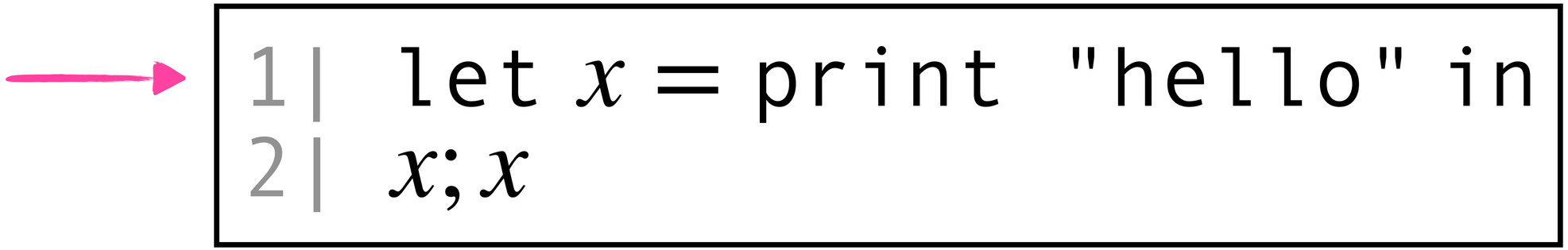
```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

hello


1| let x = print "hello" in
2| x; x
Compilation 1
hello

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
```

"sequence"

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
```

$\vdash$ `print "hello"`: $\mathbf{F}$ unit

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
```

"Returner"

$\vdash$ `print "hello"`: $\mathbf{F}$ unit

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
```

x : unit value

$\vdash$ `print "hello"`: $\mathbf{F}$ unit

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
```

$$\frac{\vdash \texttt{print "hello"} : \mathbf{F}\ \text{unit} \qquad x : \text{unit} \vdash N : B}{\vdash x \leftarrow \texttt{print "hello"}; N : B}$$

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
```

$$\frac{\vdash \texttt{print "hello"} : \mathbf{F}\ \mathsf{unit} \qquad x : \mathsf{unit} \vdash N : B}{\vdash x \leftarrow \texttt{print "hello"} ; N : B}$$

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
2| _ ← x
3| x
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
2| _ ← x
3| x
```

"sequence"

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
2| _ ← x
3| x
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
2| _ ← x
3| x
```

x : unit value

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

hello

```
1| x ← print "hello"
2| _ ← return x
3| return x
```

$x : \text{unit}$ value

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
2| _ ← return x
3| return x
```

$$\frac{\vdash x : \mathsf{unit}}{\vdash \mathtt{return}\ x : \mathbf{F}\,\mathsf{unit}}$$

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
2| _ ← return x
3| return x
```

Expected effect?

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
2| _ ← return x
3| return x
```

Expected effect?

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← return ()
2| _ ← return x
3| return x
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

$x \mapsto ()$

```
1| x ← print "hello"
2| _ ← return x
3| return x
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

$x \mapsto ()$

```
1| x ← print "hello"
2| _ ← return ()
3| return x
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

$x \mapsto ()$

```
1| x ← print "hello"
2| _ ← return x
3| return x
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

$x \mapsto ()$

```
1| x ← print "hello"
2| _ ← return x
3| return ()
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 1

```
hello
```

```
1| x ← print "hello"
2| _ ← return x
3| return x
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← print "hello"
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← {print "hello"}
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← {print "hello"}
```

"suspend"

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

"suspend"

```
1| x ← {print "hello"}
```

$\vdash$ `print "hello"`: $\mathbf{F}$ unit

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← {print "hello"}
```

$$\frac{\vdash \texttt{print "hello"} : \mathbf{F}\ \text{unit}}{\vdash \{\texttt{print "hello"}\} : \mathbf{U}\ (\mathbf{F}\ \text{unit})}$$

"Suspender"

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← {print "hello"}
```

$$\vdash \{\texttt{print "hello"}\} : \mathbf{U}\ (\mathbf{F}\ \text{unit})$$

value

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x
3| x
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x
3| x
```

$$x : \mathbf{U}\ (\mathbf{F}\ \text{unit})$$

value

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x
3| x
```

$$x : \mathbf{U}\ (\mathbf{F}\ \text{unit})$$

suspended computation

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x!
3| x!
```

$$x : \mathbf{U}\ (\mathbf{F}\ \text{unit})$$

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x ! "force"
3| x !
```

$$x : \mathbf{U}\ (\mathbf{F}\ \text{unit})$$

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x !
3| x !
```

$$\frac{\vdash x : \mathbf{U}\ (\mathbf{F}\ \mathsf{unit})}{\vdash x\,! : \mathbf{F}\ \mathsf{unit}}$$

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x !
3| x !
```

Expected effect?

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x!
3| x!
```

Expected effect?

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

$x \mapsto$ {print "hello"}

```
1| x ← return {print "hello"}
2|   _ ← x !
3| _x !
```

Expected effect?

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

$x \mapsto$ {print "hello"}

```
1| x ← return {print "hello"}
2| _ ← {print "hello"}!
3| x!
```

Expected effect?

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

$x \mapsto$ {print "hello"}

```
1| x ← return {print "hello"}
2| _ ← print "hello"
3| x !
```

Expected effect?

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

$x \mapsto$ {print "hello"}

```
1| x ← return {print "hello"}
2| _ ← return ()
3| x!
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

$x \mapsto$ {print "hello"}

```
1| x ← return {print "hello"}
2| _ ← x!
3| x!
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

$x \mapsto$ {print "hello"}

```
1| x ← return {print "hello"}
2| _ ← x!
3| {print "hello"} !
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

$x \mapsto$ {print "hello"}

```
1| x ← return {print "hello"}
2| _ ← x!
3| print "hello"
```

Expected effect?

```
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

$x \mapsto$ {print "hello"}

```
1| x ← return {print "hello"}
2| _ ← x!
3| return ()
```

Expected effect?

```
hello
hello
```

```
1| let x = print "hello" in
2| x; x
```

Compilation 2

```
hello
hello
```

```
1| x ← return {print "hello"}
2| _ ← x!
3| x!
```

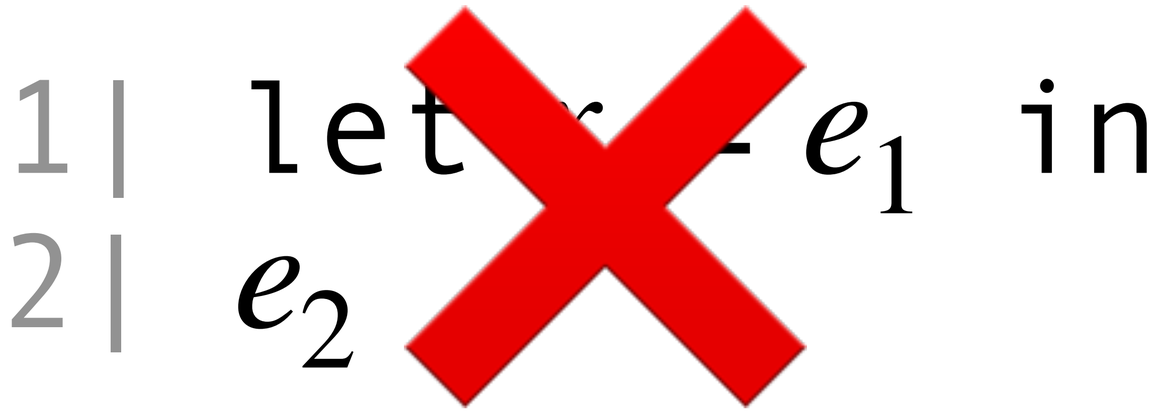
Expected effect?

```
hello
hello
```

Dead code elimination

```
1| let x = e1 in
2| e2
```

Dead code elimination



… in Call-By-Push-Value

$$
\begin{array}{l|l}
1 & x \leftarrow M \\
2 & N
\end{array}
$$

… in Call-By-Push-Value

```
1 | x ← M
2 | N
```

$$\Gamma \vdash M : \mathbf{F}\, A$$

$$\Gamma, x : A \vdash N : B$$

Effects … in Call-By-Push-Value

$$
\begin{array}{ll}
1 & x \leftarrow M \\
2 & N
\end{array}
$$

$$\Gamma \vdash M :^{\varepsilon} \mathbf{F}\, A$$

$$\Gamma, x : A \vdash N :^{\phi} B$$

Effect: is the code bound to x (i.e. M) pure?

Effects … in Call-By-Push-Value

$$\begin{array}{ll} 1 \mid & x \leftarrow_{\varepsilon} M \\ 2 \mid & N \end{array}$$

$$\Gamma \vdash M :^{\varepsilon} \mathbf{F}\, A$$

$$\Gamma, x : A \vdash N :^{\phi} B$$

Effect: is the code bound to x (i.e. M) pure?

Effects and Coeffects in Call-By-Push-Value

$$\begin{array}{l} 1 \mid \quad x \leftarrow_{\varepsilon} M \\ 2 \mid \quad N \end{array}$$

$$\gamma_1 \cdot \Gamma \vdash M :^{\varepsilon} \mathbf{F}\, A$$

$$\gamma_2 \cdot \Gamma, x :^{0} A \vdash N :^{\phi} B$$

Effect: is the code bound to x (i.e. M) pure?

Coeffect: is x unused in N?

Effects and Coeffects in Call-By-Push-Value

$$\begin{array}{l} 1 \mid \quad x \leftarrow^{0}_{\varepsilon} M \\ 2 \mid \quad N \end{array}$$

$$\gamma_1 \cdot \Gamma \vdash M :^{\varepsilon} \mathbf{F}\, A$$

$$\gamma_2 \cdot \Gamma, x :^{0} A \vdash N :^{\phi} B$$

Effect: is the code bound to x (i.e. M) pure?

Coeffect: is x unused in N?

Effects and Coeffects in Call-By-Push-Value

$$\begin{array}{l} 1 \mid \quad x \leftarrow^{0}_{\varepsilon} M \\ 2 \mid \quad N \end{array}$$

$$\gamma_1 \cdot \Gamma \vdash M :^{\varepsilon} \mathbf{F}\, A \qquad \gamma_2 \cdot \Gamma, x :^{0} A \vdash N :^{\phi} B$$

Effects and Coeffects in Call-By-Push-Value

$$\begin{array}{l} 1 \mid \quad x \leftarrow^{0}_{\varepsilon} M \\ 2 \mid \quad N \end{array}$$

$$\gamma_1 \cdot \Gamma \vdash M :^{\varepsilon} \mathbf{F}\, A \qquad \gamma_2 \cdot \Gamma, x :^{0} A \vdash N :^{\phi} B$$

Effects and Coeffects in Call-By-Push-Value

$$\begin{array}{l} 1 \mid \quad x \leftarrow^{0}_{\varepsilon} M \\ 2 \mid \quad N \end{array}$$

$$\frac{\gamma_1 \cdot \Gamma \vdash M :^{\varepsilon} \mathbf{F}\, A \qquad \gamma_2 \cdot \Gamma, x :^{0} A \vdash N :^{\phi} B}{\gamma_2 \cdot \Gamma \vdash x \leftarrow^{0}_{\varepsilon} M \,;\, N :^{\phi} B}$$

Effects and Coeffects in Call-By-Push-Value

$$\begin{array}{l} 1 \mid \quad x \leftarrow^{0}_{\varepsilon} M \\ 2 \mid \quad N \end{array}$$

$$\frac{\gamma_1 \cdot \Gamma \vdash M :^{\varepsilon} \mathbf{F}\, A \qquad \gamma_2 \cdot \Gamma, x :^{0} A \vdash N :^{\phi} B}{\gamma_2 \cdot \Gamma \vdash x \leftarrow^{0}_{\varepsilon} M \,;\, N :^{\phi} B}$$

Effects and Coeffects in Call-By-Push-Value

$$
\begin{array}{l|l}
1 & x \leftarrow^{0}_{\varepsilon} M \\
2 & N
\end{array}
$$

Effects and Coeffects in Call-By-Push-Value

$$
\begin{array}{l|l}
1 & x \leftarrow^{0}_{\varepsilon} M \\
2 & N
\end{array}
$$

Effects and Coeffects in Call-By-Push-Value

$$\underset{N}{x \overset{0}{\underset{\varepsilon}{\leftarrow}} M} \equiv N$$

Effects and Coeffects in Call-By-Push-Value

$$\begin{matrix} x \\ N \end{matrix} \overset{0}{\underset{\varepsilon}{\leftarrow}} M \equiv N$$

Effects and Coeffects in Call-By-Push-Value

$$x \overset{0}{\underset{\varepsilon}{\leftarrow}} M \\ N \equiv N$$

mechanized in 

Effects and Coeffects in Call-By-Push-Value

Semantics

$$\rho \vdash M \Downarrow T$$

Effects and Coeffects in Call-By-Push-Value

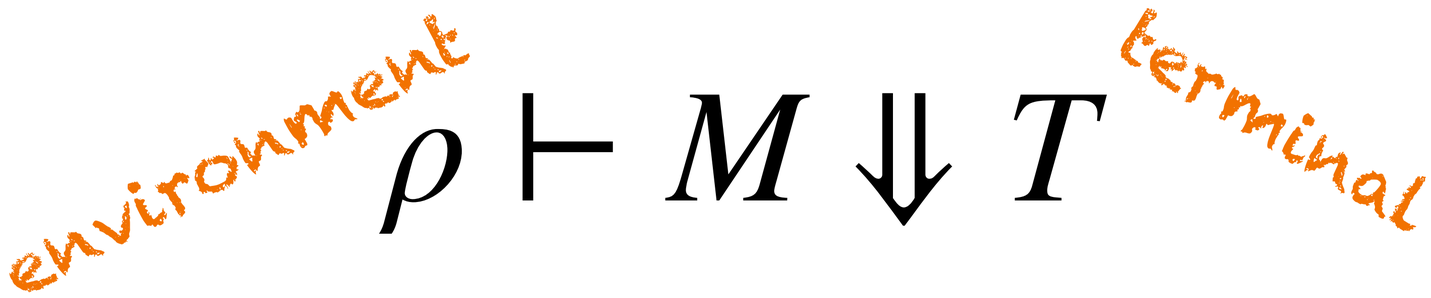
Semantics

environment

$$\rho \vdash M \Downarrow T$$

Effects and Coeffects in Call-By-Push-Value

Semantics



Effects and Coeffects in Call-By-Push-Value

Effect-tracking semantics

$$\rho \vdash M \Downarrow T \# \phi$$

Effects and Coeffects in Call-By-Push-Value

Effect-tracking semantics

$$\rho \vdash M \Downarrow T \# \phi$$

resulting effect

Effects and Coeffects in Call-By-Push-Value

Effect-tracking and coeffect-tracking semantics

$$\gamma \cdot \rho \vdash M \Downarrow T \# \phi$$

resulting effect

Effects and Coeffects in Call-By-Push-Value

Effect-tracking and coeffect-tracking semantics

consumed coeffects

$$\gamma \cdot \rho \vdash M \Downarrow T \# \phi$$

resulting effect

Effects and Coeffects in Call-By-Push-Value

Effect-tracking and coeffect-tracking semantics

$$\gamma \cdot \rho \vdash M \Downarrow T \# \phi$$

Effects and Coeffects in Call-By-Push-Value

$$\Gamma \vdash M : B$$

$$\gamma \cdot \rho \vdash M \Downarrow T \# \phi$$

Effects and Coeffects in Call-By-Push-Value

$$\Gamma \vdash M :^{\phi'} B$$

$$\gamma \cdot \rho \vdash M \Downarrow T \# \phi$$

Effects and Coeffects in Call-By-Push-Value

Type-and-effect soundness

$$\frac{\Gamma \vdash M :^{\phi'} B \qquad \phi \leq \phi'}{\gamma \cdot \rho \vdash M \Downarrow T \# \phi}$$

Effects and Coeffects in Call-By-Push-Value

Type-and-effect soundness

$$\frac{\gamma' \cdot \Gamma \vdash M :^{\phi'} B \qquad \phi \leq \phi'}{\gamma \cdot \rho \vdash M \Downarrow T \# \phi}$$

Effects and Coeffects in Call-By-Push-Value

Type-and-effect-and-coeffect soundness

$$\frac{\gamma \leq \gamma' \qquad \gamma' \cdot \Gamma \vdash M :^{\phi'} B \qquad \phi \leq \phi'}{\gamma \cdot \rho \vdash M \Downarrow T \# \phi}$$

Effects and Coeffects in Call-By-Push-Value

Type-and-effect-and-coeffect soundness

mechanized in 

$$\gamma \leq \gamma' \qquad \gamma' \cdot \Gamma \vdash M :^{\phi'} B \qquad \phi \leq \phi'$$

$$\gamma \cdot \rho \vdash M \Downarrow T \# \phi$$

Effects and Coeffects in Call-By-Push-Value

$$\frac{\Gamma \vdash M : B}{\Gamma \vdash \{M\} : \mathbf{U}\ B} \qquad \frac{\Gamma \vdash V : A}{\Gamma \vdash \texttt{return}\ V : \mathbf{F}\ A}$$

Effects and Coeffects in Call-By-Push-Value

$$\frac{\Gamma \vdash M \mathrel{\overset{\phi}{:}} B}{\Gamma \vdash \{M\} : \mathbf{U}_{\phi} B} \qquad \frac{\Gamma \vdash V : A}{\Gamma \vdash \texttt{return}\ V : \mathbf{F}\ A}$$

Effects and Coeffects in Call-By-Push-Value

$$\frac{\Gamma \vdash M \stackrel{\phi}{:} B}{\Gamma \vdash \{M\} : \mathbf{U}_{\phi} B}$$

$$\frac{\Gamma \vdash V : A}{\Gamma \vdash \texttt{return}\ V : \mathbf{F}\ A}$$

Effects and Coeffects in Call-By-Push-Value

$$\frac{\gamma \cdot \Gamma \vdash M \mathrel{\overset{\phi}{:}} B}{\gamma \cdot \Gamma \vdash \{M\} : \mathbf{U}_{\phi} B}$$

suspended effect

$$\frac{\gamma \cdot \Gamma \vdash V : A}{(q \cdot \gamma) \cdot \Gamma \vdash \mathtt{return}_{q}\, V : \mathbf{F}_{q} A}$$

Effects and Coeffects in Call-By-Push-Value

$$\frac{\gamma \cdot \Gamma \vdash M \mathrel{\dot{:}}^{\phi} B}{\gamma \cdot \Gamma \vdash \{M\} : \mathbf{U}_{\phi} B}$$

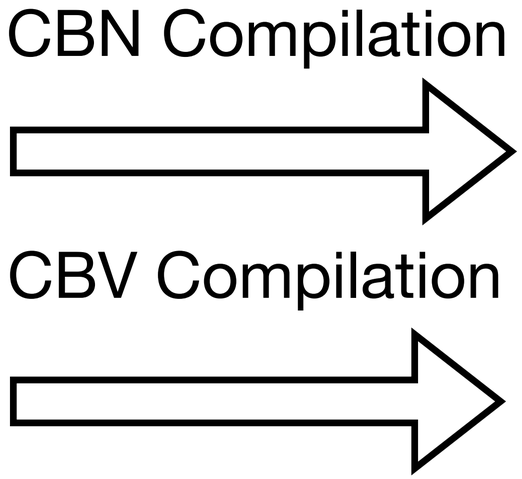
suspended effect

$$\frac{\gamma \cdot \Gamma \vdash V : A}{(q \cdot \gamma) \cdot \Gamma \vdash \mathtt{return}_{q}\, V : \mathbf{F}_{q} A}$$

multiplied coeffect

Effects and Coeffects in Call-By-Push-Value

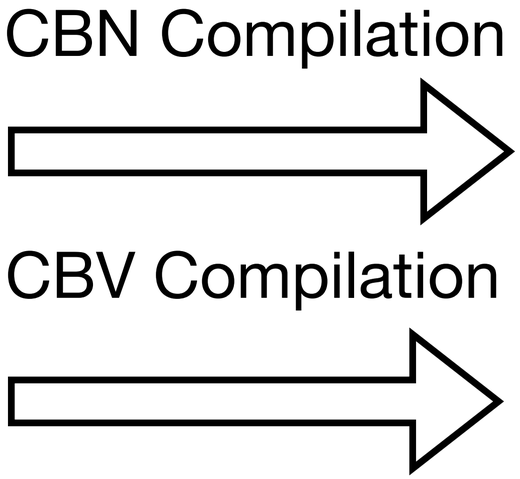
$$\Gamma \vdash e : \tau$$



Call-By-Push-Value

Effects and Coeffects in Call-By-Push-Value

$$\Gamma \vdash e : \overset{\phi}{\tau}$$

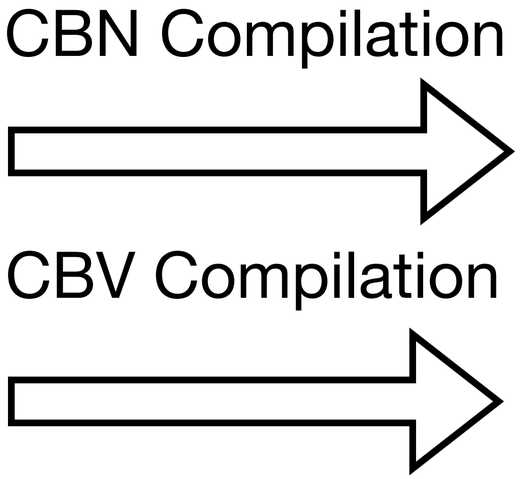


Call-By-Push-Value

preserve effects

Effects and Coeffects in Call-By-Push-Value

$$\gamma \cdot \Gamma \vdash e : \overset{\phi}{\tau}$$



Call-By-Push-Value

preserve coeffects

preserve effects

Effects and Coeffects in Call-By-Push-Value

$$\gamma \cdot \Gamma \vdash e : \overset{\phi}{\tau}$$

CBN Compilation

CBV Compilation

Call-By-Push-Value

preserve coeffects

preserve effects

mechanized in 

Effects and Coeffects in Call-By-Push-Value

Thank you!

$$\gamma \cdot \Gamma \vdash M :^{\phi} B$$

$$\gamma \cdot \rho \vdash M \Downarrow T \# \phi$$

$$x \leftarrow^{0}_{\varepsilon} M \; ; \; N \equiv N$$

mechanized in