Reasoning about effectful programs and evaluation order

Dylan McDermott

University of Cambridge

Joint work with Alan Mycroft

Goal

General framework for proving statements of the form

If <restriction on side-effects> then <evaluation order 1> is equivalent to <evaluation order 2>

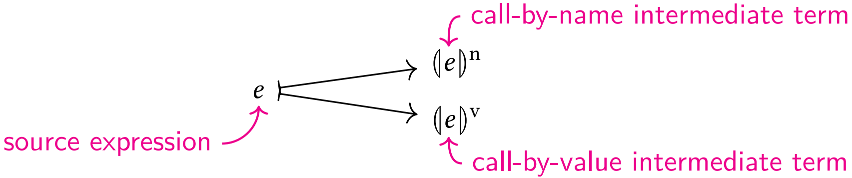
Examples:

- If there are no effects, then call-by-value is equivalent to call-by-name
- If the only effect is nontermination, then call-by-name is equivalent to call-by-need
- If the only effect is nondeterminism, then call-by-value is equivalent to call-by-need

Method

Use an intermediate language that supports various evaluation orders:

1. Translate from source language to intermediate language



2. Prove contextual equivalence

$$(\!|e|\!)^{\mathrm{n}} \;\cong_{\mathrm{ctx}}\; (\!|e|\!)^{\mathrm{v}}$$

Method

Use an intermediate language that supports various evaluation orders:

1. Translate from source language to intermediate language

call-by-name intermediate term

$$e \mapsto \begin{cases} (\!|e|\!)^{\mathrm{n}} \\ (\!|e|\!)^{\mathrm{v}} \end{cases}$$

source expression

call-by-value intermediate term

2. Prove contextual equivalence

$$\phi((\!|e|\!)^{\mathrm{n}}) \;\cong_{\mathrm{ctx}}\; (\!|e|\!)^{\mathrm{v}}$$

Subtlety: two translations have different types

$$(\!|e|\!)^{\mathrm{n}} \longmapsto \phi((\!|e|\!)^{\mathrm{n}})$$

another intermediate term

Outline

How do we prove evaluation order equivalences (assuming global restriction on side-effects)?

- When are call-by-value and call-by-name equivalent?

How do we do call-by-need?

- New intermediate language: extension of Levy's call-by-push-value to capture call-by-need
- Example: name and need are equivalent if only effect is nontermination

How do we do local (per expression) restrictions?

Call-by-push-value [Levy '99]

Split syntax into values and computations

- Values don't have side-effects, computations might

Call-by-push-value [Levy '99]

Split syntax into values and computations

- Values don't have side-effects, computations might

Not:

- Values don't reduce, computations might (complex values)
- Value types are call-by-value, computations types are call-by-name

Call-by-push-value [Levy '99]

- Can put two computations together: if M_1, M_2 are computations then
$$M_1 \ \mathbf{to} \ x.\, M_2$$
is a computation
- Can thunk computations: if M is a computation then
$$\mathbf{thunk}\, M$$
is a value

$\Rightarrow$ can do call-by-value and call-by-name (but not call-by-need)

Call-by-push-value syntax

Value types:

$$A, B ::= \dots$$
$$\mid \mathbf{U}\underline{C}$$

Computation types:

$$\underline{C}, \underline{D} ::= \dots$$
$$\mid A \to \underline{C}$$
$$\mid \mathbf{F}A$$

Value terms:

$$V, W ::= c \mid \dots \quad \text{constants, products, etc.}$$
$$\mid \mathbf{thunk}\, M \quad \text{thunks}$$
$$\mid x$$

Computation terms:

$$M, N ::= \dots \quad \text{products, etc.}$$
$$\mid \lambda x.\, M \mid V \text{ ` } M \quad \text{functions}$$
$$\mid \mathbf{return}\, V \mid M_1 \;\mathbf{to}\; x.\, M_2 \quad \text{returners}$$
$$\mid \mathbf{force}\, V$$

Call-by-push-value syntax

Value types:

$$A, B ::= \ldots \mid \mathbf{U}\underline{C}$$

Value terms:

$V, W ::=$	$c \mid \ldots$	constants, products, etc.
	$\mid \mathbf{thunk}\, M$	thunks
	$\mid x$	

Computation types:

$$\underline{C}, \underline{D} ::= \ldots \mid A \to \underline{C} \mid \mathbf{F}A$$

Computation terms:

$M, N ::=$	$\ldots$	products, etc.
	$\mid \lambda x.\, M \mid V \text{ ` } M$	functions
	$\mid \mathbf{return}\, V \mid M_1 \ \mathbf{to}\ x.\, M_2$	returners
	$\mid \mathbf{force}\, V$	

$$\frac{\Gamma \,\underline{\vdash}\, M : \underline{C}}{\Gamma \vdash \mathbf{thunk}\, M : \mathbf{U}\underline{C}} \qquad \frac{\Gamma \vdash V : \mathbf{U}\underline{C}}{\Gamma \,\underline{\vdash}\, \mathbf{force}\, V : \underline{C}}$$

Call-by-push-value syntax

Value types:

$A, B ::= \ldots$

$\mid \mathbf{U}\underline{C}$

Value terms:

$V, W ::= c \mid \ldots$	constants, products, etc.
$\mid \mathbf{thunk}\, M$	thunks
$\mid x$	

Computation types:

$\underline{C}, \underline{D} ::= \ldots$

$\mid A \to \underline{C}$

$\mid \mathbf{F}A$

Computation terms:

$M, N ::= \ldots$	products, etc.
$\mid \lambda x.\, M \mid V \text{ ` } M$	functions
$\mid \mathbf{return}\, V \mid M_1 \ \mathbf{to}\ x.\, M_2$	returners
$\mid \mathbf{force}\, V$	

Typing contexts: $\Gamma ::= \diamond \mid x : A$

Call-by-push-value syntax

Value types:

$A, B ::= \ldots$

$\mid \mathbf{U}\underline{C}$

Value terms:

$V, W ::= c \mid \ldots$ constants, products, etc.

$\mid \mathbf{thunk}\, M$ thunks

$\mid x$

Computation types:

$\underline{C}, \underline{D} ::= \ldots$

$\mid A \to \underline{C}$

$\mid \mathbf{F}A$

Computation terms:

$M, N ::= \ldots$ products, etc.

$\mid \lambda x.\, M \mid V\text{ ` }M$ functions

$\mid \mathbf{return}\, V \mid M_1 \;\mathbf{to}\; x.\, M_2$ returners

$\mid \mathbf{force}\, V$

$$\frac{\Gamma \vdash V : A}{\Gamma \,\underline{\vdash}\, \mathbf{return}\, V : \mathbf{F}A} \qquad \frac{\Gamma \,\underline{\vdash}\, M_1 : \mathbf{F}A \qquad \Gamma, x : A \,\underline{\vdash}\, M_2 : \underline{C}}{\Gamma \,\underline{\vdash}\, M_1 \;\mathbf{to}\; x.\, M_2 : \underline{C}}$$

Call-by-push-value equational theory

We also have an equational theory

$$V \equiv V' \qquad M \equiv M'$$

Use this to define contextual equivalence

$$M \cong_{\mathrm{ctx}} M'$$

iff

$$\mathcal{C}[M] \equiv \mathcal{C}[M']$$

for all closed $\mathcal{C}$ of type $\mathbf{F}G$, where G doesn't contain thunks

Call-by-value and call-by-name

$$(\!|e|\!)^{\mathrm{v}} \cong_{\mathrm{ctx}} (\!|e|\!)^{\mathrm{n}}$$

Call-by-value and call-by-name

Source language types:

$$\tau ::= \mathbf{unit} \mid \mathbf{bool} \mid \tau \to \tau'$$

Translations from value and name into CBPV:

$$\begin{array}{rcl} \tau & \mapsto & \text{value type } (\!|\tau|\!)^{\mathrm{v}} \\ \mathbf{unit} & \mapsto & \mathbf{unit} \\ \mathbf{bool} & \mapsto & \mathbf{bool} \\ (\tau \to \tau') & \mapsto & \mathbf{U}((\!|\tau|\!)^{\mathrm{v}} \to \mathbf{F}(\!|\tau'|\!)^{\mathrm{v}}) \end{array} \qquad \begin{array}{rcl} \tau & \mapsto & \text{computation type } (\!|\tau|\!)^{\mathrm{n}} \\ \mathbf{unit} & \mapsto & \mathbf{F}\,\mathbf{unit} \\ \mathbf{bool} & \mapsto & \mathbf{F}\,\mathbf{bool} \\ (\tau \to \tau') & \mapsto & ((\mathbf{U}(\!|\tau|\!)^{\mathrm{n}}) \to (\!|\tau'|\!)^{\mathrm{n}}) \end{array}$$

$$\Gamma, x : \tau \;\mapsto\; (\!|\Gamma|\!)^{\mathrm{v}}, x : (\!|\tau|\!)^{\mathrm{v}} \qquad \Gamma, x : \tau \;\mapsto\; (\!|\Gamma|\!)^{\mathrm{n}}, x : \mathbf{U}(\!|\tau|\!)^{\mathrm{n}}$$

$$\Gamma \vdash e : \tau \;\mapsto\; (\!|\Gamma|\!)^{\mathrm{v}} \,\underline{\vdash}\, (\!|e|\!)^{\mathrm{v}} : \mathbf{F}(\!|\tau|\!)^{\mathrm{v}} \qquad \Gamma \vdash e : \tau \;\mapsto\; (\!|\Gamma|\!)^{\mathrm{n}} \,\underline{\vdash}\, (\!|e|\!)^{\mathrm{n}} : (\!|\tau|\!)^{\mathrm{n}}$$

Call-by-value and call-by-name

$$
\begin{array}{ccc}
(\!|\Gamma|\!)^{\mathrm{v}} & \xrightarrow{(\!|e|\!)^{\mathrm{v}}} & \mathbf{F}(\!|\tau|\!)^{\mathrm{v}} \\
\big\downarrow & \cong_{\mathrm{ctx}} & \big\uparrow \\
(\!|\Gamma|\!)^{\mathrm{n}} & \xrightarrow[(\!|e|\!)^{\mathrm{n}}]{} & (\!|\tau|\!)^{\mathrm{n}}
\end{array}
$$

Call-by-value and call-by-name

Isomorphism between call-by-value and call-by-name computations?

$$\Gamma \underline{\vdash} M : \mathbf{F}(\!|\tau|\!)^{\mathrm{v}} \quad \mapsto \quad \Gamma \underline{\vdash} \Phi_\tau M : \ (\!|\tau|\!)^{\mathrm{n}}$$

$$\Gamma \underline{\vdash} N : \ (\!|\tau|\!)^{\mathrm{n}} \quad \mapsto \quad \Gamma \underline{\vdash} \Psi_\tau N : \mathbf{F}(\!|\tau|\!)^{\mathrm{v}}$$

Call-by-value and call-by-name

Isomorphism between call-by-value and call-by-name computations?

$$\Gamma \vdash M : \mathbf{F}(\!|\tau|\!)^{\mathrm{v}} \quad \mapsto \quad \Gamma \vdash \Phi_\tau M : (\!|\tau|\!)^{\mathrm{n}}$$

$$\Gamma \vdash N : (\!|\tau|\!)^{\mathrm{n}} \quad \mapsto \quad \Gamma \vdash \Psi_\tau N : \mathbf{F}(\!|\tau|\!)^{\mathrm{v}}$$

Value to Name to Value:

$$\Psi_\tau(\Phi_\tau(\mathbf{return}\, V)) \equiv \mathbf{return}\, V$$

The other way depends on the effects

Logical relations for CBPV

$$\begin{array}{rcl} \text{value types } A & \mapsto & \text{relations } \mathcal{R}[\![A]\!] \text{ on closed terms } V : A \\ \text{computation types } \underline{C} & \mapsto & \text{relations } \mathcal{R}[\![\underline{C}]\!] \text{ on closed terms } M : \underline{C} \end{array}$$

We'll want

$$(M, M') \in \mathcal{R}[\![\mathbf{F}G]\!] \;\Rightarrow\; M \equiv M'$$

for ground types G (to prove contextual equivalence)

Logical relations for CBPV

Assume:

- Defined in usual way on type formers excluding **F**

$$\mathcal{R}[\![\mathbf{U}\underline{C}]\!] = \{(\mathbf{thunk}\, M, \mathbf{thunk}\, M') \mid (M, M') \in \mathcal{R}[\![\underline{C}]\!]\}$$
$$\mathcal{R}[\![A \to \underline{C}]\!] = \{(M, M') \mid \forall (V, V') \in \mathcal{R}[\![A]\!].\, (V`M, V'`M') \in \mathcal{R}[\![\underline{C}]\!]\}$$

- Closed under **return**:

$$(V, V') \in \mathcal{R}[\![A]\!] \quad \Rightarrow \quad (\mathbf{return}\, V, \mathbf{return}\, V') \in \mathcal{R}[\![\mathbf{F}A]\!]$$

- Closed under **to**: if $x : A \vdash N, N' : \underline{C}$ and

$$(M, M') \in \mathcal{R}[\![\mathbf{F}A]\!] \qquad \forall (V, V') \in \mathcal{R}[\![A]\!].\, (N[x \mapsto V], N'[x \mapsto V'])$$

  then

$$(M \;\mathbf{to}\; x.\, N, M' \;\mathbf{to}\; x.\, N') \in \mathcal{R}[\![\underline{C}]\!]$$

- Constants related to themselves: if $c : A$ then $(c, c) \in \mathcal{R}[\![A]\!]$
- Transitivity

Logical relations for CBPV

Lemma (Fundamental)

If $x_1 : A_1, \ldots, x_n : A_n \vdash M : \underline{C}$ *and* $(V_i, V_i') \in \mathcal{R}[\![A_i]\!]$ *for each* i *then*

$$(M[x_1 \mapsto V_1, \ldots, x_n \mapsto V_n], M[x_1 \mapsto V_1', \ldots, x_n \mapsto V_n']) \in \mathcal{R}[\![\underline{C}]\!]$$

From Name to Value and back

Definition (Thunkable [Führmann '99])

A computation $\Gamma \vdash M : \mathbf{F}A$ is *thunkable* if

$$M \ \mathbf{to}\ x.\,\mathbf{return}\,(\mathbf{thunk}\,(\mathbf{return}\,x)) \qquad \text{and} \qquad \mathbf{return}\,(\mathbf{thunk}\,M)$$

are related by $\mathcal{R}[\![\mathbf{F}(\mathbf{U}(\mathbf{F}A))]\!]$.

This implies:

$$M \ \mathbf{to}\ x.\,\mathbf{thunk}\,(\mathbf{return}\,x)\,\text{`}\,N \qquad \text{related to} \qquad \mathbf{thunk}\,M\,\text{`}\,N$$

Lemma

If everything is thunkable and $M : (\!|\tau|\!)^{\mathrm{n}}$ *then*

$$(\Phi_\tau(\Psi_\tau M)) \quad \mathcal{R}\left[\!\left[(\!|\tau|\!)^{\mathrm{n}}\right]\!\right] \quad M$$

The equivalence

Want to show that

$$\begin{array}{ccc} (\!|\Gamma|\!)^{\mathrm{v}} & \xrightarrow{(\!|e|\!)^{\mathrm{v}}} & \mathbf{F}(\!|\tau|\!)^{\mathrm{v}} \\ \downarrow & \cong_{\mathrm{ctx}} & \uparrow \\ (\!|\Gamma|\!)^{\mathrm{n}} & \xrightarrow[(\!|e|\!)^{\mathrm{n}}]{} & (\!|\tau|\!)^{\mathrm{n}} \end{array}$$

Meaning:

$$(\!|e|\!)^{\mathrm{v}} \cong_{\mathrm{ctx}} \Psi_B\left((\!|e|\!)^{\mathrm{n}} \begin{bmatrix} x_1 \mapsto \mathbf{thunk}\,(\Phi_{A_1}(\mathbf{return}\,x_1)) \\ ,\ldots, \\ x_n \mapsto \mathbf{thunk}\,(\Phi_{A_n}(\mathbf{return}\,x_n)) \end{bmatrix} \right)$$

In particular, for closed e of ground type ($\mathbf{unit}$ or $\mathbf{bool}$):

$$(\!|e|\!)^{\mathrm{v}} \equiv (\!|e|\!)^{\mathrm{n}}$$

The equivalence

Lemma

Suppose everything is thunkable. If $x_1 : A_1, \ldots, x_n : A_n \vdash e : A$ *and* V_i *related to* V_i' *for each* i *then*

$$(\!|e|\!)^{\mathrm{v}}[x_1 \mapsto V_1, \ldots, x_n \mapsto V_n]$$

is related to

$$\Psi_B\left((\!|e|\!)^{\mathrm{n}}\begin{bmatrix} x_1 \mapsto \mathbf{thunk}\,(\Phi_{A_1}(\mathbf{return}\,V_1')) \\ ,\ldots, \\ x_n \mapsto \mathbf{thunk}\,(\Phi_{A_n}(\mathbf{return}\,V_n')) \end{bmatrix}\right)$$

A trivial example

For no side-effects:

$$\mathcal{R}[\![\mathrm{F}A]\!] \;=\; \{(\mathbf{return}\, V, \mathbf{return}\, V') \mid (V, V') \in \mathcal{R}[\![A]\!]\}$$

A non-example

Read-only state

$$\mathbf{get} : \mathbf{F}\,\mathbf{bool}$$

$$\mathbf{get}\ \mathbf{to}\ x.\,\mathbf{return}\,() \equiv \mathbf{return}\,()$$
$$\mathbf{get}\ \mathbf{to}\ x.\,\mathbf{get}\ \mathbf{to}\ y.\,\mathbf{return}\,(x, y) \equiv \mathbf{get}\ \mathbf{to}\ z.\,\mathbf{return}\,(z, z)$$

Logical relation:

$$\mathcal{R}[\![\mathbf{F}A]\!] = \left\{ \begin{matrix} (\mathbf{get}\ \mathbf{to}\ x.\,\mathbf{if}\ x\ \mathbf{then}\ \mathbf{return}\ V_1\ \mathbf{else}\ \mathbf{return}\ V_2 \\ ,\mathbf{get}\ \mathbf{to}\ x.\,\mathbf{if}\ x\ \mathbf{then}\ \mathbf{return}\ V_1'\ \mathbf{else}\ \mathbf{return}\ V_2') \end{matrix} \;\middle|\; (V_1, V_1'), (V_2, V_2') \in \mathcal{R}[\![A]\!] \right\}$$

Not all computations are thunkable!

- All thunkable computations have the form

$$\mathbf{return}\,V$$

Goal

General framework for proving statements of the form

> *If <restriction on side-effects> then <evaluation order 1> is equivalent to <evaluation order 2>*

Examples:

- ~~If there are no effects, then call-by-value is equivalent to call-by-name~~
- If the only effect is nontermination, then call-by-name is equivalent to call-by-need
- If the only effect is nondeterminism, then call-by-value is equivalent to call-by-need

Extended call-by-push-value (ECBPV)

New computation forms:

$$
\begin{array}{rcll}
M, N & ::= & \ldots & \\
& & \mid \underline{x} & \text{computation variables} \\
& & \mid M_1 \ \mathbf{need} \ \underline{x}.\, M_2 & \text{call-by-need sequencing}
\end{array}
$$

Typing:

$$\Gamma ::= \ldots \mid \underline{x} : \mathbf{F}A$$

$$
\frac{(\underline{x} : \mathbf{F}A) \in \Gamma}{\Gamma \vdash \underline{x} : \mathbf{F}A}
\qquad
\frac{\Gamma \vdash M_1 : \mathbf{F}A \qquad \Gamma, \underline{x} : \mathbf{F}A \vdash M_2 : \underline{C}}{\Gamma \vdash M_1 \ \mathbf{need} \ \underline{x}.\, M_2 : \underline{C}}
$$

Extended call-by-push-value

Important equation:

$$M_1 \ \mathbf{need}\ \underline{x}.\, \underline{x}\ \mathbf{to}\ y.\, M_2 \ \equiv\ M_1\ \mathbf{to}\ y.\, M_2[\underline{x} \mapsto \mathbf{return}\, y]$$

Associativity:

$$\begin{aligned}
(M_1 \ \mathbf{to}\ x.\, M_2)\ \mathbf{to}\ y.\, M_3 &\equiv M_1\ \mathbf{to}\ x.\, (M_2\ \mathbf{to}\ y.\, M_3) \\
(M_1 \ \mathbf{need}\ x.\, M_2)\ \mathbf{need}\ y.\, M_3 &\equiv M_1\ \mathbf{need}\ x.\, (M_2\ \mathbf{need}\ y.\, M_3) \\
(M_1 \ \mathbf{need}\ x.\, M_2)\ \mathbf{to}\ y.\, M_3 &\equiv M_1\ \mathbf{need}\ x.\, (M_2\ \mathbf{to}\ y.\, M_3) \\
(M_1 \ \mathbf{to}\ x.\, M_2)\ \mathbf{need}\ y.\, M_3 &\not\equiv M_1\ \mathbf{to}\ x.\, (M_2\ \mathbf{need}\ y.\, M_3)
\end{aligned}$$

Extended call-by-push-value

Given

$$\Gamma \underline{\vdash} M_1 : \mathbf{F}A \qquad\qquad \Gamma, \underline{x} : \mathbf{F}A \underline{\vdash} M_2 : \underline{C}$$

have various evaluation orders:

- Call-by-value: $M_1 \, \mathbf{value} \, \underline{x}.\, M_2 \;\equiv\; M_1 \;\mathbf{to}\; y.\, M_2[\underline{x} \mapsto \mathbf{return}\, y]$
- Call-by-name: $M_1 \, \mathbf{name} \, \underline{x}.\, M_2 \;\equiv\; M_2[\underline{x} \mapsto M_1]$
- Call-by-need: $M_1 \;\mathbf{need}\; \underline{x}.\, M_2$ (builtin)

Call-by-need translation

$$
\begin{array}{rcl}
\tau & \mapsto & \text{value type } (\!|\tau|\!)^{\text{need}} \\[1em]
\mathbf{unit} & \mapsto & \mathbf{unit} \\
\mathbf{bool} & \mapsto & \mathbf{bool} \\
(\tau \to \tau') & \mapsto & \mathbf{U}\Big(\mathbf{U}(\mathbf{F}(\!|\tau|\!)^{\text{need}}) \;\to\; \mathbf{F}(\!|\tau'|\!)^{\text{need}}\Big)
\end{array}
$$

$$
\Gamma, x : \tau \quad \mapsto \quad (\!|\Gamma|\!)^{\text{need}},\; \underline{x} : \mathbf{F}(\!|\tau|\!)^{\text{need}}
$$

Call-by-need translation

$$
\begin{array}{rcl}
\tau & \mapsto & \text{value type } (\!|\tau|\!)^{\mathrm{need}} \\[1em]
\mathbf{unit} & \mapsto & \mathbf{unit} \\
\mathbf{bool} & \mapsto & \mathbf{bool} \\
(\tau \to \tau') & \mapsto & \mathbf{U}\Big(\mathbf{U}(\mathbf{F}(\!|\tau|\!)^{\mathrm{need}}) \;\to\; \mathbf{F}(\!|\tau'|\!)^{\mathrm{need}}\Big)
\end{array}
$$

$$
\Gamma, x : \tau \quad \mapsto \quad (\!|\Gamma|\!)^{\mathrm{need}},\; \underline{x} : \mathbf{F}(\!|\tau|\!)^{\mathrm{need}}
$$

This could also be call-by-name!

Call-by-need translation

$$\Gamma \vdash e : \tau \longmapsto (\!|\Gamma|\!)^{\text{need}} \underline{\vdash} (\!|e|\!)^{\text{need}} : \mathbf{F}(\!|\tau|\!)^{\text{need}}$$

$$e\,e' \qquad (\!|e|\!)^{\text{need}}\ \mathbf{to}\ f.\,(\mathbf{thunk}\ (\!|e'|\!)^{\text{need}})\,`\,(\mathbf{force}\ f)$$

$$\lambda \underline{x}.\, e \qquad \mathbf{return}\ (\mathbf{thunk}\ (\lambda x'.\ (\mathbf{force}\ x')\ \mathbf{need}\ \underline{x}.\ (\!|e|\!)^{\text{need}}))$$

Two nice properties:

- Applying lambdas

$$(\!|(\lambda x.\, e)\ e'|\!)^{\text{need}} \equiv (\!|e'|\!)^{\text{need}}\ \mathbf{need}\ \underline{x}.\ (\!|e|\!)^{\text{need}}$$

- Translation is sound (wrt small-step operational semantics)

$$e \overset{\text{need}}{\rightsquigarrow} e' \qquad \Rightarrow \qquad (\!|e|\!)^{\text{need}} \equiv (\!|e'|\!)^{\text{need}}$$

[Ariola & Felleisen '97]

Proving an equivalence

If the only effect is nontermination, call-by-name is equivalent to call-by-need

Method:

1. Instantiate ECBPV: add constants that induce diverging computations $\Omega_{\underline{C}}$
2. Prove internal equivalence:

$$M_1 \ \mathbf{name} \ \underline{x}.\, M_2 \quad \cong_{\text{ctx}} \quad M_1 \ \mathbf{need} \ \underline{x}.\, M_2$$

3. Corollary:

$$(\!|e|\!)^{\text{moggi}} \ \cong_{\text{ctx}} \ (\!|e|\!)^{\text{need}}$$

Internal equivalence: proof idea

$$M_1 \,\mathbf{name}\, \underline{x}.\, M_2 \quad \cong_{\text{ctx}} \quad M_1 \;\mathbf{need}\; \underline{x}.\, M_2$$

Proof: use logical relations

- Reasoning about **to**:

$$\Omega_{\mathbf{F}A} \;\mathbf{to}\; x.\, M_2 \equiv \Omega_{\underline{C}} \qquad \mathbf{return}\, V \;\mathbf{to}\; x.\, M_2 \equiv M_2[x \mapsto V]$$

diverging computation

pure computation

- Don't have similar equations for **need**:

$$\Omega_{\mathbf{F}A} \;\mathbf{need}\; \underline{x}.\, M_2 \not\equiv \Omega_{\underline{C}}$$

- Relate open terms: Kripke logical relations of varying arity [Jung and Tiuryn '93]

$$\mathcal{R}[\![A]\!]\,\Gamma \subseteq \text{Term}_A^{\Gamma} \times \text{Term}_A^{\Gamma}$$

Global restriction on side-effects

If whole language restricted to nontermination, then

$$M_1 \,\mathbf{name}\, \underline{x}.\, M_2 \quad \cong_{\mathrm{ctx}} \quad M_1 \,\mathbf{need}\, \underline{x}.\, M_2$$

Local restriction on side-effects

If ~~whole language~~ M_1 restricted to nontermination, then

$$M_1 \ \mathbf{name}\ \underline{x}.\, M_2 \quad \cong_{\mathrm{ctx}} \quad M_1 \ \mathbf{need}\ \underline{x}.\, M_2$$

Effect system for (E)CBPV

Goal: place upper bound on side-effects of computations

- Replace returner types $\mathbf{F}A$ with $\langle\varepsilon\rangle A$
- Track effects $\varepsilon \subseteq \Sigma$

$$\Sigma \coloneqq \{\mathsf{diverge}, \mathsf{get}, \mathsf{put}, \mathsf{raise}, \ldots\}$$

$$\Omega : \langle\{\mathsf{diverge}\}\rangle A \qquad \mathbf{get} : \langle\{\mathsf{get}\}\rangle\mathbf{bool} \qquad \cdots$$

- Internal equivalence (with effect system):
  If $M_1 : \langle\varepsilon\rangle A$ *for* $\varepsilon \subseteq \{\mathsf{diverge}\}$*, then*

$$M_1\;\mathbf{name}\;\underline{x}.\,M_2 \quad \cong_{\mathrm{ctx}} \quad M_1\;\mathbf{need}\;\underline{x}.\,M_2$$

Effect system for (E)CBPV

$$\frac{\Gamma \underline{\vdash} M : \underline{C} \qquad \underline{C} <: \underline{D}}{\Gamma \underline{\vdash} M : \underline{D}}$$

Subtyping $\underline{C} <: \underline{D}$

$\langle\varepsilon\rangle A <: \langle\varepsilon'\rangle B$ if $\varepsilon \subseteq \varepsilon'$ and $A <: B$

$$\frac{\Gamma \underline{\vdash} M_1 : \langle\varepsilon\rangle A \quad \Gamma, x : A \underline{\vdash} M_2 : \underline{C}}{\Gamma \underline{\vdash} M_1 \ \mathbf{to}\ x.\, M_2 : \langle\varepsilon\rangle\underline{C}}$$

Preordered monoid action: $\langle\varepsilon\rangle\underline{C}$

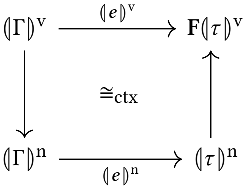
$$\langle\varepsilon\rangle(\langle\varepsilon'\rangle A) := \langle\varepsilon \cup \varepsilon'\rangle A$$

$$\langle\varepsilon\rangle(A \to \underline{C}) := A \to \langle\varepsilon\rangle\underline{C}$$

Overview

How to prove an equivalence between evaluation orders:

1. Translate from source language to intermediate language
2. Prove contextual equivalence



- Works for call-by-value, call-by-name
  - Call-by-need using extended call-by-push-value
- Also works for local restrictions on side-effects using an effect system

A slightly less trivial example

C-style undefined behaviour

$$\mathbf{undef}_{\underline{C}} \preccurlyeq M \qquad \mathbf{undef}_{\mathrm{F}A}\ \mathbf{to}\ x.\, M \equiv \mathbf{undef}_{\underline{C}}$$

Logical relation:

$$\begin{aligned} \mathcal{R}[\![\mathrm{F}A]\!] \ &:= \ \{(\mathbf{return}\,V, \mathbf{return}\,V') \mid (V, V') \in \mathcal{R}[\![A]\!]\} \\ &\quad \cup \{(\mathbf{undef}_{\mathrm{F}A},\, M)\} \end{aligned}$$

Can replace value with name (but not name with value)