Grading call-by-push-value, explicitly and implicitly

Dylan McDermott

University of Oxford

dylan@dylanm.org

Two developments in computational effects

- ***Grading*** (Katsumata 2014, and others)
  Static analysis of computational effects
- ***Call-by-push-value*** (Levy 1999)
  A calculus for studying computational effects

Two developments in computational effects

- ***Grading*** (Katsumata 2014, and others)
  Static analysis of computational effects
- ***Call-by-push-value*** (Levy 1999)
  A calculus for studying computational effects

Grading call-by-push-value, explicitly and implicitly

Grading

A paradigm for static analysis of effectful programs

1. Choose a collection of *grades* e
2. Instantiate language-specific inference rules, to associate a grade to each effectful syntactic element
3. Prove properties of "computations of grade e"

Grading example: type-and-effect analysis

"has grade $e \subseteq \{\text{get}, \text{put}, \text{raise}, \ldots\}$"	means	"does not use any operation that is not in e"

$$\begin{array}{l} x \leftarrow \texttt{get}()\,; \\ \texttt{if}\ x\ \texttt{then}\ \texttt{raise}()\ \texttt{else}\ \texttt{return}\ x \end{array}$$

has grade $\{\text{get}, \text{raise}\}$

$$\begin{array}{l} x \leftarrow \texttt{get}()\,; \\ \texttt{put}(\texttt{true})\,; \\ \texttt{if}\ x\ \texttt{then}\ \texttt{raise}()\ \texttt{else}\ \texttt{return}\ x \end{array}$$

does not have grade $\{\text{get}, \text{raise}\}$

Grading example: type-and-effect analysis

"has grade $e \subseteq \{\text{get}, \text{put}, \text{raise}, \ldots\}$"	means	"does not use any operation that is not in e"

```
x ← get() ;
if x then raise() else return x
```

has grade $\{\text{get}, \text{raise}\}$

Some of the inference rules:

$$\frac{}{\texttt{return}(v) \text{ has grade } \{\}} \qquad \frac{t \text{ has grade } d \qquad u \text{ has grade } e}{(x \leftarrow t\,;\ u) \text{ has grade } (d \cup e)} \qquad \frac{t \text{ has grade } d \qquad d \subseteq e}{t \text{ has grade } e}$$

Grading example: session types

$$\text{grades } e ::=_{\nu} \text{end} \mid \bigoplus_{i \in I} \mathsf{p}!\ell_i.\, e_i \mid \&_{i \in I} \mathsf{p}?\ell_i.\, e_i$$

$$\mathtt{recv}_{\mathsf{p}}\, \{(\mathtt{price}, x).\ \mathtt{send}_{\mathsf{p}}\, (\mathtt{yes});\ \mathtt{return}\, x\} \quad \text{has grade} \quad \mathsf{p}?\mathtt{price}\langle\mathtt{int}\rangle. \begin{pmatrix} \mathsf{p}!\mathtt{yes}.\ \text{end} \\ \oplus \mathsf{p}!\mathtt{no}.\ \text{end} \end{pmatrix}$$

Grading example: session types

$$\text{grades } e ::=_\nu \text{end} \mid \bigoplus_{i \in I} \mathsf{p}!\ell_i.\, e_i \mid \&_{i \in I} \mathsf{p}?\ell_i.\, e_i$$

$$\mathtt{recv}_{\mathsf{p}} \{(\mathtt{price}, x).\ \mathtt{send}_{\mathsf{p}}\,(\mathtt{yes});\ \mathtt{return}\, x\} \quad \text{has grade} \quad \mathsf{p}?\mathtt{price}\langle\mathtt{int}\rangle.\begin{pmatrix}\mathsf{p}!\mathtt{yes}.\,\text{end} \\ \oplus\mathsf{p}!\mathtt{no}.\,\text{end}\end{pmatrix}$$

Some of the inference rules:

$$\frac{}{\mathtt{return}(v) \text{ has grade end}} \qquad \frac{t \text{ has grade } d \qquad u \text{ has grade } e}{(x \leftarrow t\,;\ u) \text{ has grade } (d \cdot e)} \qquad \frac{t \text{ has grade } d \qquad d \leqslant e}{t \text{ has grade } e}$$

$$\begin{aligned} \text{end} \cdot e &= e \\ (\bigoplus_{i \in I} \mathsf{p}!\ell_i.\, d_i) \cdot e &= \bigoplus_{i \in I} \mathsf{p}!\ell_i.\, (d_i \cdot e) \\ (\&_{i \in I} \mathsf{p}?\ell_i.\, d_i) \cdot e &= \&_{i \in I} \mathsf{p}?\ell_i.\, (d_i \cdot e) \end{aligned}$$

session subtyping

Grading

Grades e are elements of an ordered monoid

$$(\mathbb{E}, \leqslant, \mathbf{1}, \cdot)$$

$$\frac{}{\texttt{return}(v) \text{ has grade } \mathbf{1}} \qquad \frac{t \text{ has grade } d \qquad u \text{ has grade } e}{(\, x \leftarrow t \,;\; u) \text{ has grade } (d \cdot e)} \qquad \frac{t \text{ has grade } d \qquad d \leqslant e}{t \text{ has grade } e}$$

Call-by-push-value (without grades)

Split syntax into *values* $V, W : A, B$ and *computations* $M, N : \underline{C}, \underline{D}$

$\underline{C}, \underline{D} ::= \mathbf{F}A$		*returners*: running $M : \mathbf{F}A$ may have effects, and any result has type A
	$\mid A \to \underline{C}$	*functions*: application of $M : A \to \underline{C}$ to $V : A$ has type $\underline{C}$
	$\mid \prod_{i \in I} \underline{C}_i$	*tuples*: the ith projection of $M : \prod_{i \in I} \underline{C}_i$ has type $\underline{C}_i$

Call-by-push-value (without grades)

Split syntax into *values* $V, W : A, B$ and *computations* $M, N : \underline{C}, \underline{D}$

$\underline{C}, \underline{D} ::=$	$\mathbf{F}A$	*returners*: running $M : \mathbf{F}A$ may have effects, and any result has type A
	$\mid A \to \underline{C}$	*functions*: application of $M : A \to \underline{C}$ to $V : A$ has type $\underline{C}$
	$\mid \prod_{i \in I} \underline{C}_i$	*tuples*: the ith projection of $M : \prod_{i \in I} \underline{C}_i$ has type $\underline{C}_i$

Computations include:

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

$$\frac{\mathrm{op} : A \rightsquigarrow B \qquad \Gamma \vdash V : A \qquad \Gamma, y : B \vdash M : \underline{C}}{\Gamma \vdash \mathbf{do}\ y \leftarrow \mathrm{op}\, V\ \mathbf{then}\ M : \underline{C}}$$

e.g.

$$\mathrm{get} : \mathbf{1} \rightsquigarrow \mathbf{bool}$$
$$\mathrm{raise} : \mathbf{1} \rightsquigarrow \mathbf{empty}$$
$$\mathrm{send}_{\mathsf{p},\ell\langle A \rangle} : A \rightsquigarrow \mathbf{1}$$

Call-by-push-value (without grades)

Split syntax into *values* $V, W : A, B$ and *computations* $M, N : \underline{C}, \underline{D}$

$\underline{C}, \underline{D} ::= \mathbf{F}A$		*returners*: running $M : \mathbf{F}A$ may have effects, and any result has type A
	$\mid A \to \underline{C}$	*functions*: application of $M : A \to \underline{C}$ to $V : A$ has type $\underline{C}$
	$\mid \prod_{i \in I} \underline{C}_i$	*tuples*: the ith projection of $M : \prod_{i \in I} \underline{C}_i$ has type $\underline{C}_i$

Computations include:

$$\frac{\Gamma, x : A \vdash M : \underline{C}}{\Gamma \vdash \lambda x : A.\, M : A \to \underline{C}}$$

This work: grading call-by-push-value

Key insights:

1. Grades are for tracking observable effects
2. We observe effects at returner type

$\underline{C}, \underline{D} ::= \mathbf{F}A$	*returners*: ***running*** $M : \mathbf{F}A$ ***may have effects***, and any result has type A
$\mid A \to \underline{C}$	*functions*: application of $M : A \to \underline{C}$ to $V : A$ has type $\underline{C}$
$\mid \prod_{i \in I} \underline{C}_i$	*tuples*: the ith projection of $M : \prod_{i \in I} \underline{C}_i$ has type $\underline{C}_i$

Call-by-push-value with effects

$\underline{C}, \underline{D} ::= \mathbf{F}_e\, A$		*returners*: running $M : \mathbf{F}_e A$ may have effects ***of grade e***, and any result has type A
	$\mid A \to \underline{C}$	*functions*: application of $M : A \to \underline{C}$ to $V : A$ has type $\underline{C}$
	$\mid \prod_{i \in I} \underline{C}_i$	*tuples*: the ith projection of $M : \prod_{i \in I} \underline{C}_i$ has type $\underline{C}_i$

Call-by-push-value with effects

$\underline{C}, \underline{D} ::= \mathbf{F}_e\, A$		*returners*: running $M : \mathbf{F}_e A$ may have effects ***of grade e***, and any result has type A
	$\mid A \to \underline{C}$	*functions*: application of $M : A \to \underline{C}$ to $V : A$ has type $\underline{C}$
	$\mid \prod_{i \in I} \underline{C}_i$	*tuples*: the ith projection of $M : \prod_{i \in I} \underline{C}_i$ has type $\underline{C}_i$

Subtyping $A <: B$ and $\underline{C} <: \underline{D}$:

$$\frac{d \leqslant e \qquad A <: B}{\mathbf{F}_d A <: \mathbf{F}_e B}$$

+ congruence rules

Action $\langle\!\langle d \rangle\!\rangle \underline{C}$ of $\mathbb{E}$ on computation types:

$$\langle\!\langle d \rangle\!\rangle (\mathbf{F}_e A) = \mathbf{F}_{d \cdot e} A$$

$$\langle\!\langle d \rangle\!\rangle \left(\prod_{i \in I} \underline{C}_i\right) = \prod_{i \in I} \langle\!\langle d \rangle\!\rangle \underline{C}_i$$

$$\langle\!\langle d \rangle\!\rangle (A \to \underline{C}) = A \to \langle\!\langle d \rangle\!\rangle \underline{C}$$

Call-by-push-value with effects

$\underline{C}, \underline{D} ::= \mathbf{F}_e\, A$ — *returners*: running $M : \mathbf{F}_e A$ may have effects ***of grade e***, and any result has type A

$\mid A \to \underline{C}$ — *functions*: application of $M : A \to \underline{C}$ to $V : A$ has type $\underline{C}$

$\mid \prod_{i \in I} \underline{C}_i$ — *tuples*: the ith projection of $M : \prod_{i \in I} \underline{C}_i$ has type $\underline{C}_i$

Computations include:

$$\frac{\Gamma \vdash^{\mathsf{g}} V : A}{\Gamma \vdash^{\mathsf{g}} \mathbf{return}\, V : \mathbf{F}_{\mathbf{1}} A} \qquad \frac{\Gamma \vdash^{\mathsf{g}} M : \mathbf{F}_d A \qquad \Gamma, x : A \vdash^{\mathsf{g}} N : \underline{C}}{\Gamma \vdash^{\mathsf{g}} M\, \mathbf{to}\, x.\, N : \langle\!\langle d \rangle\!\rangle \underline{C}} \qquad \frac{\Gamma \vdash^{\mathsf{g}} M : \underline{C} \qquad \underline{C} <: \underline{D}}{\Gamma \vdash^{\mathsf{g}} \mathbf{coerce}_{\underline{D}}\, M : \underline{D}}$$

$$\frac{\mathrm{op} : A \rightsquigarrow_d B \qquad \Gamma \vdash^{\mathsf{g}} V : A \qquad \Gamma, y : B \vdash^{\mathsf{g}} M : \underline{C}}{\Gamma \vdash^{\mathsf{g}} \mathbf{do}\, y \leftarrow \mathrm{op}\, V\, \mathbf{then}\, M : \langle\!\langle d \rangle\!\rangle \underline{C}}$$

e.g.

$\mathrm{get} : \mathbf{1} \rightsquigarrow_{\{\mathrm{get}\}} \mathbf{bool}$

$\mathrm{raise} : \mathbf{1} \rightsquigarrow_{\{\mathrm{raise}\}} \mathbf{empty}$

$\mathrm{send}_{\mathrm{p},\ell\langle A \rangle} : A \rightsquigarrow_{\mathrm{p}!\ell\langle A \rangle.\,\mathrm{end}} \mathbf{1}$

Call-by-push-value with effects

$\underline{C}, \underline{D} ::= \mathbf{F}_e\, A$		*returners*: running $M : \mathbf{F}_e A$ may have effects ***of grade e***, and any result has type A
	$\mid A \to \underline{C}$	*functions*: application of $M : A \to \underline{C}$ to $V : A$ has type $\underline{C}$
	$\mid \prod_{i \in I} \underline{C}_i$	*tuples*: the ith projection of $M : \prod_{i \in I} \underline{C}_i$ has type $\underline{C}_i$

Computations include:

$$\frac{\Gamma, x : A \vdash^{g} M : \underline{C}}{\Gamma \vdash^{g} \lambda x : A.\, M : A \to \underline{C}}$$

Example

```
x ← get() ;
if x then raise() else return x
```

has grade {get, raise}

CBPVE computation of type $\mathbf{F}_{\{\text{get},\text{raise}\}}\mathbf{bool}$:

$$\begin{array}{l} \mathbf{do}\ x \leftarrow \text{get}()\ \mathbf{then} \\ \mathbf{match}\ x\ \mathbf{with}\ \{\ \ \text{true}.\ \mathbf{do}\ z \leftarrow \text{raise}()\ \mathbf{then\ match}\ z\ \mathbf{with}\ \{\} \\ \qquad\qquad\quad ,\ \ \text{false}.\ \mathbf{coerce}_{\mathbf{F}_{\{\text{raise}\}}\mathbf{bool}}\ (\mathbf{return}\ x)\} \end{array}$$

Graded algebra models

We get a denotational semantics from any

- *strong graded monad* T on a bicartesian closed category, equipped with
- a morphism $\kappa_{\mathrm{op}} : [\![A]\!] \to T[\![B]\!]d$ for each $\mathrm{op} : A \leadsto_d B$

value type A	$\mapsto$	object $[\![A]\!]$
computation type $\underline{C}$	$\mapsto$	T-algebra $[\![\underline{C}]\!]$
typing context Γ	$\mapsto$	object $[\![\Gamma]\!]$
value subtyping $A <: B$	$\mapsto$	morphism $[\![A <: B]\!] : [\![A]\!] \to [\![B]\!]$
computation subtyping $\underline{C} <: \underline{D}$	$\mapsto$	algebra morphism $[\![\underline{C} <: \underline{D}]\!] : [\![\underline{C}]\!] \to [\![\underline{D}]\!]$
value $\Gamma \vdash^{\mathsf{g}} V : A$	$\mapsto$	morphism $[\![V]\!] : [\![\Gamma]\!] \to [\![A]\!]$
computation $\Gamma \vdash^{\mathsf{g}} M : \underline{C}$	$\mapsto$	morphism $[\![M]\!] : [\![\Gamma]\!] \to [\![\underline{C}]\!]_{\mathbf{1}}$

Call-by-push-value with effects

A calculus for studying ***graded*** computational effects

- Subsumes graded versions of (fine-grain) call-by-value, and of call-by-name
- Grades are ***explicit*** in the syntax

Grading as analysis

We have a judgment

$$\Gamma \vdash^{\mathtt{g}} M : \underline{C}$$

CBPVE typing context

CBPVE computation

CBPVE computation type

$$\frac{\Gamma \vdash^{\mathtt{g}} M : \underline{C} \qquad \underline{C} <: \underline{D}}{\Gamma \vdash^{\mathtt{g}} \mathbf{coerce}_{\underline{D}}\, M : \underline{D}}$$

$$\frac{\Gamma, x : A \vdash^{\mathtt{g}} M : \underline{C}}{\Gamma \vdash^{\mathtt{g}} \lambda x : A.\, M : A \to \underline{C}}$$

But we want

$$\Gamma \vdash^{\mathtt{i}} M : \underline{C}$$

CBPVE typing context

CBPV computation

CBPVE computation type

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

$$\frac{\Gamma, x : A' \vdash^{\mathtt{i}} M : \underline{C}}{\Gamma \vdash^{\mathtt{i}} \lambda x : A.\, M : A' \to \underline{C}}$$

(where A' is A annotated with grades)

Implicit grades

Define

$$\Gamma \vdash^{\mathtt{i}} M : \underline{C} \quad \text{if} \quad \exists M'.\ \lfloor M' \rfloor = M \ \wedge\ \Gamma \vdash^{\mathtt{g}} M' : \underline{C}$$

where

$$\lfloor - \rfloor : \text{CBPVE} \to \text{CBPV}$$

$$\lfloor \textbf{coerce}_{\underline{D}}\, M \rfloor = \lfloor M \rfloor$$
$$\lfloor \lambda x : A.\, M \rfloor = \lambda x : \lfloor A \rfloor.\ \lfloor M \rfloor$$

erases grades and **coerce**.

Models for implicit grades

If $\Gamma \vdash^{\mathtt{i}} M : \underline{C}$, then define

$$[\![M]\!] = [\![M']\!] \quad \text{where} \quad \lfloor M' \rfloor = M \;\wedge\; \Gamma \vdash^{\mathtt{g}} M' : \underline{C}$$

assuming *coherence*:

$$\lfloor M_1' \rfloor = \lfloor M_2' \rfloor \;\Rightarrow\; [\![M_1']\!] = [\![M_2']\!] \quad \text{for all } \Gamma \vdash^{\mathtt{g}} M_i' : \underline{C}$$

Models for implicit grades

If $\Gamma \vdash^{\mathtt{i}} M : \underline{C}$, then define

$$[\![M]\!] = [\![M']\!] \quad \text{where} \quad \lfloor M' \rfloor = M \;\wedge\; \Gamma \vdash^{\mathtt{g}} M' : \underline{C}$$

assuming *coherence*:

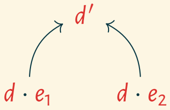
$$\lfloor M_1' \rfloor = \lfloor M_2' \rfloor \;\Rightarrow\; [\![M_1']\!] = [\![M_2']\!] \quad \text{for all } \Gamma \vdash^{\mathtt{g}} M_i' : \underline{C}$$

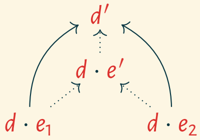
But coherence is *false* in general for graded algebra models.

Proving coherence

Assume the ordered monoid of grades has *left-cancellative upper bounds*:

$$d \cdot e_1 \leqslant d' \geqslant d \cdot e_2 \;\Rightarrow\; \exists e'.\, e_1 \leqslant e' \geqslant e_2 \wedge d \cdot e' \leqslant d'$$





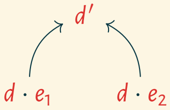
Examples:

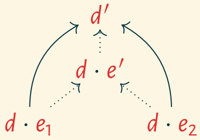
- Any join-semilattice such that multiplication left-distributes over joins $(d \cdot (e_1 \sqcup e_2) = (d \cdot e_1) \sqcup (d \cdot e_2))$: take $e' = e_1 \sqcup e_2$
  e.g. $(\mathcal{P}\{\text{get}, \text{put}, \text{raise}, \ldots\}, \subseteq, \{\}, \cup)$
- Not session types

Proving coherence

Assume the ordered monoid of grades has *left-cancellative upper bounds*:

$$d \cdot e_1 \leqslant d' \geqslant d \cdot e_2 \;\Rightarrow\; \exists e'.\, e_1 \leqslant e' \geqslant e_2 \wedge d \cdot e' \leqslant d'$$





Then coherence holds:

$$\lfloor M_1' \rfloor = \lfloor M_2' \rfloor \;\Rightarrow\; [\![M_1']\!] = [\![M_2']\!] \quad \text{for all } \Gamma \vdash^{g} M_i' : \underline{C}$$

Proof idea.
Use logical relations: relate $\Gamma \vdash^{g} N_1 : \underline{D}_1$ to $\Gamma \vdash^{g} N_2 : \underline{D}_2$, where $\lfloor \underline{D}_1 \rfloor = \lfloor \underline{D}_2 \rfloor$, by $\top\top$-lifting ∎

Three devlopments in computational effects

- ***Grading*** (Katsumata 2014, and others)
  Static analysis of computational effects
- ***Call-by-push-value*** (Levy 1999)
  A calculus for studying computational effects
- ***Call-by-push-value with effects*** (this paper)
  A calculus for studying graded computational effects
  (With implicit grades, assuming coherence)