Classical-by-Need

(A Classy Call-by-Need)

Pierre-Marie Pédrot & Alexis Saurin

7th April 2016

ESOP 2016

The Two Faces of Computation on Demand

$$\Delta = \lambda x.(x)x, \quad \Omega = (\Delta)\Delta, \quad I = \lambda y.y$$

Unnecessary computations in call-by-value:

$$M = (\lambda x.I)\Omega \to_{CBN} I$$

$$M = (\lambda x.I)\Omega \to_{CBV} M \to_{CBV} M \to_{CBV} \dots$$

Duplication of computations in call-by-name:

$$N = (\Delta)(I)I \to_{CBN} (I)I(I)I \to_{CBN} (I)(I)I \to_{CBN} (I)I \to_{CBN} I$$

$$N = (\Delta)(I)I \to_{CBV} (\Delta)I \to_{CBN} (I)I \to_{CBN} I$$

The Two Faces of Computation on Demand

$$\Delta = \lambda x.(x)x, \quad \Omega = (\Delta)\Delta, \quad I = \lambda y.y$$

Unnecessary computations in call-by-value:

$$M = (\lambda x.I)\Omega \rightarrow_{CBN} I$$

$$M = (\lambda x.I)\Omega \rightarrow_{CBV} M \rightarrow_{CBV} M \rightarrow_{CBV} \ldots$$

Duplication of computations in call-by-name:

$$N = (\Delta)(I)I \rightarrow_{CBN} (I)I(I)I \rightarrow_{CBN} (I)(I)I \rightarrow_{CBN} (I)I \rightarrow_{CBN} I$$

$$N = (\Delta)(I)I \rightarrow_{CBV} (\Delta)I \rightarrow_{CBN} (I)I \rightarrow_{CBN} I$$

Ideally, one would like to have one's cake and eat it too: to postpone evaluating an expression (...) until it is clear that its value is really needed, but also to avoid repeated evaluation.

(John Reynolds)

Call-by-need λ-calculus

Ariola-Felleisen, JFP 97

Syntax

$$\begin{array}{llcl}
\text{terms} & t & ::= & x \parallel \lambda x.t \parallel (t)t \\
\text{values} & V & ::= & \lambda x.t \\
\text{answers} & A & ::= & V \parallel (\lambda x.A)\ t \\
\text{evaluation contexts} & E & ::= & \Box \parallel Et \parallel (\lambda x.E)\ t \\
 & & & \parallel (\lambda x.E[x])\ E
\end{array}$$

Reductions

$$\begin{array}{llcl}
(\mathit{deref}) & (\lambda x.E[x])\ V & \to & (\lambda x.E[V])\ V \\
(\mathit{lift}) & ((\lambda x.A)\ t)u & \to & (\lambda x.Au)\ t \\
(\mathit{assoc}) & (\lambda x.E[x])\ (\lambda y.A)\ t & \to & (\lambda y.(\lambda x.E[x])\ A)\ t
\end{array}$$

Other calculi:

Maraist et al, JFP 98: same standard reduction

Ariola, Herbelin & S., TLCA 11: in $\overline{\lambda}\mu\tilde{\mu}$

Chang & Felleisen, ESOP 12: single axiom call-by-need

Accattoli et al., ICFP 14: explicit substitution call-by-need

Classical By-need

- Call-by-need is somehow an effect
- Not distinguishable from by-name in a pure setting...

Classical By-need

- Call-by-need is somehow an effect
- Not distinguishable from by-name in a pure setting...

- But difference observable in presence of other effects!
- Several possible interactions
- In particular with first-class continuations

Classical By-need Calculi?

- Previous work: Ariola, Herbelin and S. formulated call-by-need strategies in $\overline{\lambda}\mu\tilde{\mu}$.
- In such a setting: control built-in and by-need wrought out

Classical By-need Calculi?

- Previous work: Ariola, Herbelin and S. formulated call-by-need strategies in $\overline{\lambda}\mu\tilde{\mu}$.
- In such a setting: control built-in and by-need wrought out

- We provide a more canonical presentation of call-by-need
- Inspired by this one weird trick from Linear Logic
- Naturally provides a classical by-need calculus (actually several)

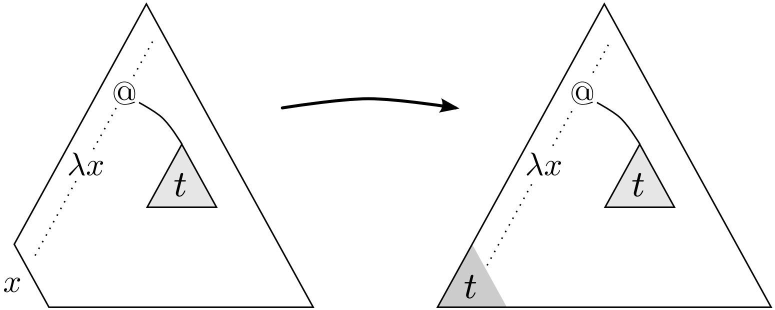
Organization of the Talk

- Linear Head Reduction
- Classical Linear Head Reduction
- From LHR to Call-by-need
- Classical By-need

Linear Head Reduction

Linear head reduction, informally

(Danos & Regnier, ≈ 1990)



Comparison between LHR and call-by-need

Striking similarities

- Both can be viewed as optimization of standard evaluation strategies;
- Both rely on a linear, rather than destructive, substitution;
- A variable is substituted only if it is necessary for pursuing the computation;
- Both share with call-by-name the same notion of convergence and the induced observational equivalences;
- Not easily presented as reduction relation.

Krivine Abstract Machine

Closures	c	$::=$	(t,σ)
Environments	σ	$::=$	$\emptyset \mid \sigma + (x := c)$
Stacks	π	$::=$	$\varepsilon \mid c \cdot \pi$
Processes	p	$::=$	$\langle c \mid \pi \rangle$

Push	$\langle ((t)\,u,\sigma) \mid \pi \rangle$	$\rightarrow$	$\langle (t,\sigma) \mid (u,\sigma) \cdot \pi \rangle$
Pop	$\langle (\lambda x.t,\sigma) \mid c \cdot \pi \rangle$	$\rightarrow$	$\langle (t,\sigma + (x := c)) \mid \pi \rangle$
Grab	$\langle (x,\sigma + (x := c)) \mid \pi \rangle$	$\rightarrow$	$\langle c \mid \pi \rangle$
Garbage	$\langle (x,\sigma + (y := c)) \mid \pi \rangle$	$\rightarrow$	$\langle (x,\sigma) \mid \pi \rangle$

Krivine Abstract Machine

$$\begin{array}{llcl} \text{Closures} & c & ::= & (t, \sigma) \\ \text{Environments} & \sigma & ::= & \emptyset \mid \sigma + (x := c) \\ \text{Stacks} & \pi & ::= & \varepsilon \mid c \cdot \pi \\ \text{Processes} & p & ::= & \langle c \mid \pi \rangle \end{array}$$

$$\begin{array}{llcl} \textsc{Push} & \langle ((t)\,u, \sigma) \mid \pi \rangle & \rightarrow & \langle (t, \sigma) \mid (u, \sigma) \cdot \pi \rangle \\ \textsc{Pop} & \langle (\lambda x.t, \sigma) \mid c \cdot \pi \rangle & \rightarrow & \langle (t, \sigma + (x := c)) \mid \pi \rangle \\ \textsc{Grab} & \langle (x, \sigma + (x := c)) \mid \pi \rangle & \rightarrow & \langle c \mid \pi \rangle \\ \textsc{Garbage} & \langle (x, \sigma + (y := c)) \mid \pi \rangle & \rightarrow & \langle (x, \sigma) \mid \pi \rangle \end{array}$$

$$\boxed{\text{Is this really (weak) head reduction?}}$$

Krivine Abstract Machine

$$\begin{array}{llll} \text{Closures} & c & ::= & (t,\sigma) \\ \text{Environments} & \sigma & ::= & \emptyset \mid \sigma + (x := c) \\ \text{Stacks} & \pi & ::= & \varepsilon \mid c \cdot \pi \\ \text{Processes} & p & ::= & \langle c \mid \pi \rangle \end{array}$$

$$\begin{array}{llcl} \textsc{Push} & \langle ((t)\,u,\sigma) \mid \pi \rangle & \to & \langle (t,\sigma) \mid (u,\sigma)\cdot\pi \rangle \\ \textsc{Pop} & \langle (\lambda x.t,\sigma) \mid c\cdot\pi \rangle & \to & \langle (t,\sigma + (x := c)) \mid \pi \rangle \\ \textsc{Grab} & \langle (x,\sigma + (x := c)) \mid \pi \rangle & \to & \langle c \mid \pi \rangle \\ \textsc{Garbage} & \langle (x,\sigma + (y := c)) \mid \pi \rangle & \to & \langle (x,\sigma) \mid \pi \rangle \end{array}$$

Is this really (weak) head reduction?

Simulating is not the same as implementing.

σ-equivalence

(Danos & Regnier, ≈ 1990)

$$\boxed{\begin{array}{rcl} (\lambda x_1.t)u_1 u_2 & =_\sigma & (\lambda x_1.(t)u_2)u_1 \\ (\lambda x_1.\lambda x_2.t)u & =_\sigma & \lambda x_2.(\lambda x_1.t)u \end{array}}$$

- $\leadsto$ Originated in the theory of linear logic proof nets: Inspired by the translation of λ-terms in proof-nets and the induced identification.
- $\leadsto$ A relation capturing the KAM behaviour.
- $\leadsto$ Skips redexes ignored by the KAM.
- $\leadsto$ Up to σ-equivalence, LHR is the usual head reduction, made linear.

LHR as a calculus

Insensitivity to σ-equivalence can be achieved by a context grammar:

Definition (λ_{lh})

closure contexts	$\mathscr{C}$	$::=$	$[\cdot] \parallel (\mathscr{C}[\lambda x.\mathscr{C}])\,t$
left evaluation contexts	E	$::=$	$[\cdot] \parallel (E)\,t \parallel \lambda x.E$

$$(\beta_{lh}) \quad (\mathscr{C}[\lambda x.E[x]])\,t \quad \rightarrow \quad (\mathscr{C}[\lambda x.E[t]])\,t$$

$+$ congruence w.r.t E

Theorem

- *β_{lh} is stable by σ-equivalence.*
- *λ_{lh} coincides with Danos-Regnier LHR.*

Closure Contexts and the KAM

PUSH and POP transitions implement the computation of closure contexts

Proposition

Let $\mathscr{C}$ be a closure context. There exists $[\mathscr{C}]_\sigma$ such that:

$$\langle(\mathscr{C}[t],\sigma) \mid \pi\rangle \longrightarrow^*_{\text{PUSH},\text{POP}} \langle(t,\sigma+[\mathscr{C}]_\sigma) \mid \pi\rangle$$

Conversely, for all t_0 and σ_0 such that

$$\langle(t,\sigma) \mid \pi\rangle \longrightarrow^*_{\text{PUSH},\text{POP}} \langle(t_0,\sigma_0) \mid \pi\rangle$$

there exists $\mathscr{C}_0$ such that $t = \mathscr{C}_0[t_0]$.

$[\mathscr{C}]_\sigma$ defined by induction over $\mathscr{C}$ as follows:

$[[\cdot]]_\sigma \equiv \emptyset \qquad [\mathscr{C}_1[\lambda x.\mathscr{C}_2]\ t]_\sigma \equiv [\mathscr{C}_1]_\sigma + (x := (t,\sigma)) + [\mathscr{C}_2]_{\sigma+[\mathscr{C}_1]_\sigma+(x:=(t,\sigma))}$

Classical LHR

$\lambda\mu$-calculus variant of the LHR

Left stack contexts K:

$$K ::= [\cdot] \mid [\alpha]L[\mu\beta.K]$$

Classical extension of left contexts and closure contexts:

$$\begin{array}{lcl} \overline{\mathscr{C}} & ::= & [\cdot] \mid \overline{\mathscr{C}}_1[\lambda x.\overline{\mathscr{C}}_2]\ t \mid \overline{\mathscr{C}}_1[\mu\alpha.K[[\alpha]\overline{\mathscr{C}}_2]] \\ \overline{L} & ::= & [\cdot] \mid \lambda x.\overline{L} \mid \overline{L}\ t \mid \mu\beta.[\alpha]\overline{L} \end{array}$$

Classical LHR:

The classical LHR is defined by the following reduction:

$$\overline{\mathscr{C}}[\lambda x.\overline{L}[x]]\ t \quad \rightarrow_{clh} \quad \overline{\mathscr{C}}[\lambda x.\overline{L}[t]]\ t$$

$+$ congruence w.r.t. $\overline{L}$.

λ_{clh} is classical LHR

Definition (μ-KAM)

$$\sigma \quad ::= \quad \cdots \mid \sigma + (\alpha := \pi) \qquad\qquad \pi \quad ::= \quad \cdots \mid (\alpha, \sigma)$$

$$\begin{array}{lcl} \langle(\mu\alpha.c, \sigma) \mid \pi\rangle & \rightarrow_{Save} & \langle(c, \sigma + (\alpha := \pi)) \mid \varepsilon\rangle \\ \langle([\alpha]t, \sigma) \mid \varepsilon\rangle & \rightarrow_{Restore} & \langle(t, \sigma) \mid \sigma(\alpha)\rangle \end{array}$$

As expected, λ_{clh} simulates intensionally the μKAM:

Theorem

Let $c_1 \rightarrow_{clh} c_2$ where $c_1 := [\alpha]\overline{L}_1[\overline{\mathscr{C}}[\lambda x.\overline{L}_2[x]]\ t]$, then the substitution sequence of process c_1 is either empty or of the form $t :: \ell$ where ℓ is the substitution sequence of process c_2.

Towards Call-by-need

From LHR to Call-by-need

In three easy steps!

1. Weak LHR
2. Value passing
3. Closure sharing

(Step 1) Weak LHR

We need to track λ-abstractions that pertain to a closure context.

Definition (Marked λ-calculus)

$$t, u ::= x \mid (t)u \mid \lambda x.t \mid \ell x.t$$

We only consider well-balanced terms.

Definition (Marked closure contexts)

$$\mathscr{C} ::= [\cdot] \mid (\mathscr{C}_1[\ell x.\mathscr{C}_2])\,t$$

$\rightsquigarrow$ Such contexts are a more structured version of explicit substitutions

$$(\mathscr{C}_1[\ell x.\mathscr{C}_2])\,t \cong \texttt{let } x := t \texttt{ in } \mathscr{C}_1[\mathscr{C}_2]$$

(Step 1) Weak LHR

Definition (Weak LHR)

Weak left contexts $\quad E^w ::= [\cdot] \mid (E^w)\, t \mid \ell x. E^w$

$$
\begin{array}{lrcl}
(\beta_{wlh}) & \mathscr{C}[\lambda x.t]\, u & \to & \mathscr{C}[\ell x.t]\, u \\
 & \mathscr{C}[\ell x.E^w[x]]\, t & \to & \mathscr{C}[\ell x.E^w[t]]\, t
\end{array}
$$

$+$ congruence w.r.t. E^w

This reduction is still stable by σ.

(Step 2) Call-by-"value" LHR

We restrict substitution to values-up-to closures:

$$W ::= \mathscr{C}[\lambda x.t]$$

and adapt the contexts accordingly:

Value left contexts $\quad E^v ::= [\cdot] \mid (E^v)\,t \mid \ell x.E^v \mid (\mathscr{C}[\ell x.E^v_1[x]])\,E^v_2$

The call-by-value weak LHR is then obtained straightforwardly:

Definition (By-value LHR)

$$\begin{array}{llcl} (\beta_{wlv}) & \mathscr{C}[\lambda x.t]\,u & \to & \mathscr{C}[\ell x.t]\,u \\ & \mathscr{C}[\ell x.E^v[x]]\,W & \to & \mathscr{C}[\ell x.E^v[W]]\,W \end{array}$$

$+$ congruence w.r.t. E^v

Still stable by σ.

By-value ?

- λ_{wlv} already implements a call-by-need strategy
- Not a reduction scheme from the literature, though.

There is a duplication of computation:

$$(\mathscr{C}'[\ell x.E^v[x]])\mathscr{C}[V] \quad \rightarrow \quad (\mathscr{C}'[\ell x.E^v[\mathscr{C}[V]]])\mathscr{C}[V]$$

$\mathscr{C}$ is copied, which will end up in recomputing its bound terms if ever they are going to be used throughout the reduction.

(Step 3) Closure sharing

Solve this similarly to the Assoc rule in Ariola-Felleisen's calculus:

Definition (By-value LHR with sharing)

$$
\begin{array}{lrcl}
(\beta_{wls}) & \mathscr{C}[\lambda x.t]\,u & \to & \mathscr{C}[\ell x.t]\,u \\
 & \mathscr{C}'[\ell x.E^v[x]]\,\mathscr{C}[V] & \to & \mathscr{C}[\mathscr{C}'[\ell x.E^v[V]]\,V]
\end{array}
$$

$+$ congruence w.r.t. E^v

Theorem

λ_{wls} is essentially Chang-Felleisen's calculus.

Classical By-need

(At last!)

Classical By Need

Following the same three steps...

$$
\begin{array}{lcl}
\mathscr{C} & ::= & [\cdot] \mid (\mathscr{C}_1[\ell x.\mathscr{C}_2])\,t \mid \mathscr{C}_1[\mu\alpha.K^v[[\alpha]\mathscr{C}_2]] \\
E^v & ::= & [\cdot] \mid (E^v)\,t \mid \ell x.E^v \mid (\mathscr{C}[\ell x.E_1^v[x]])\,E_2^v \mid \mu\alpha.K^v[[\alpha]E^v] \\
K^v & ::= & [\cdot] \mid [\alpha]E^v[\mu\beta.K^v]
\end{array}
$$

Definition (Classical-by-need)

$$
\begin{array}{llcl}
(\beta_{cls}) & \mathscr{C}[\lambda x.t]\,u & \to & \mathscr{C}[\ell x.t]\,u \\
 & \mathscr{C}'[\ell x.E^v[x]]\,\mathscr{C}[V] & \to & \mathscr{C}[\mathscr{C}'[\ell x.E^v[V]]\,V] \\
 & \mathscr{C}'[\ell x.E^v[x]]\,\mathscr{C}[\mu\alpha.c] & \to & \mathscr{C}[\mu\alpha.c\{\alpha := [\alpha](\mathscr{C}'[\ell x.E^v[x]])\,\_\}]
\end{array}
$$

$+$ congruence w.r.t. K^v

A bit too powerful

- A very smart stack substitution!
- Thanks to closure contexts, never need to substitute stacks eagerly
- ... **except** when a $\mu\alpha.c$ term needed

This does not look like anything known from the literature, so we can't relate it to a previous calculus...

A Dumber Classical By Need

$$
\begin{array}{lcl}
\mathscr{C} & ::= & [\cdot] \mid (\mathscr{C}_1[\ell x.\mathscr{C}_2])\,t \\
E^v & ::= & [\cdot] \mid (E^v)\,t \mid \ell x.E^v \mid (\mathscr{C}[\ell x.E^v_1[x]])\,E^v_2 \\
K^v & ::= & [\cdot] \mid [\alpha]E^v[\mu\beta.K^v]
\end{array}
$$

Definition (Classical-by-need with Intuitionistic Contexts)

$$
\begin{array}{lccc}
(\beta_{cls'}) & \mathscr{C}[\lambda x.t]\,u & \to & \mathscr{C}[\ell x.t]\,u \\
 & \mathscr{C}'[\ell x.E^v[x]]\,\mathscr{C}[V] & \to & \mathscr{C}[\mathscr{C}'[\ell x.E^v[V]]\,V] \\
 & \color{red}{[\alpha]E^v[\mu\beta.K^v[[\beta]t]]} & \color{red}{\to} & \color{red}{[\alpha]E^v[\mu\beta.K^v[[\alpha]E^v[t]]]}
\end{array}
$$

$+$ congruence w.r.t. K^v

Comparison with AHS classical call-by-need calculus

- Ariola, Herbelin and S. proposed a classical by-need λ-calculus derived from a call-by-need $\overline{\lambda}\mu\tilde{\mu}$-calculus.
- In that calculus, β is implemented by plain β_v-rule, a feature of sequent calculus.
- Correspondence with a modified version of this calculus, AHS', featuring a deref-rule à la Ariola-Felleisen:

Theorem

For any command c, there exists an infinite standard reduction in AHS'-calculus starting from c iff there exists an infinite reduction starting from c in the classical by-need calculus with Intuitionistic contexts.

Conclusion

- Reformulation of LHR;
- Extension to the $\lambda\mu$-calculus / classical logic;
- Connection between LHR and call-by-need by deriving call-by-need from LHR. Surprisingly, this connection seemed to have remained unexploited (and unnoticed?) until our work and Accattolli et al work.

Lazy	=	Demand-driven	+	Memoization	+	Sharing
		(weak LHR)		*(by value)*		*(closure shar.)*

- Closure contexts are not new but we made explicit their central role for both LHR and call-by-need, which are essentially calculi with reductions up-to closure contexts.
- We defined a classical by-need calculus, again from LHR.

Thanks