Mirroring Call-by-Need, or Values Acting Silly

Beniamino Accattoli[1], Adrienne Lancelot[12]

[1]Inria & LIX, École Polytechnique
[2]IRIF, Université Paris Cité & CNRS

July 13th 2024 – FSCD 2024

Outline

Call-by-Name and Call-by-Value

Evaluation strategies describe how to compute.

β-Reduction by Name: $(\lambda x.t)u \mapsto_{\beta} t\{x\leftarrow u\}$

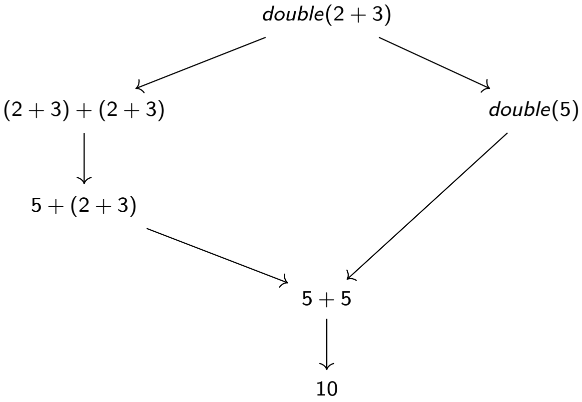
β-Reduction by Value: $(\lambda x.t)v \mapsto_{\beta_v} t\{x\leftarrow v\}$

For values v that are *answers*, i.e. computations that ended.

Call-by-Name vs. Call-by-Value

Let $double := x \mapsto x + x$. How do you calculate $double(2+3)$?

Call-by-Name | Call-by-Value



Feed any argument to any function | Only feed functions with *values*

Call-by-Name vs. Call-by-Value

Computations may never end: $\Omega := (\lambda x.xx)(\lambda x.xx)$

Let a constant function $always\_1 : x \mapsto 1$.
How to compute $always\_1(\Omega)$?

Call-by-Name | Call-by-Value

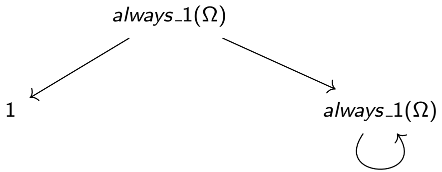
Feed any argument to any function | Only feed functions with *values*

Call-by-Name vs. Call-by-Value

Computations may never end: $\Omega := (\lambda x.xx)(\lambda x.xx)$

Let a constant function $always\_1 : x \mapsto 1$.
How to compute $always\_1(\Omega)$?

Call-by-Name | Call-by-Value



Feed any argument to any function | Only feed functions with *values*

Call-by-Name vs. Call-by-Value

Call-by-Name and Call-by-Value compute quite differently.

Efficiency: Call-by-Value computes *faster* than Call-by-Name.

Erasability: Call-by-Value gets stuck on *erasable* arguments.

Efficiency and Erasability can be combined: Call-by-Need!

Call-by-Name vs. Call-by-Value

Call-by-Name and Call-by-Value compute quite differently.

Efficiency: Call-by-Value computes *faster* than Call-by-Name.

Erasability: Call-by-Value gets stuck on *erasable* arguments.

Efficiency and Erasability can be combined: Call-by-Need!

Evaluation Strategies combining Name and Value

	Duplication by Name *Silly Duplication*	Duplication by Value *Wise Duplication*
Erasure by Name *Wise Erasure*	Call-by-Name	Call-by-Need
Erasure by Value *Silly Erasure*	**Call-by-Silly**	Call-by-Value

Evaluation Strategies combining Name and Value

	Duplication by Name *Silly Duplication*	Duplication by Value *Wise Duplication*
Erasure by Name *Wise Erasure*	Call-by-Name	Call-by-Need
Erasure by Value *Silly Erasure*	**Call-by-Silly**	Call-by-Value

Evaluation Strategies combining Name and Value

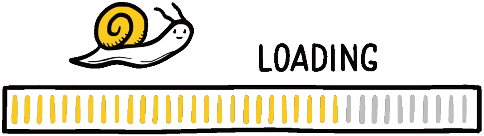
	Duplication by Name *Silly Duplication*	Duplication by Value *Wise Duplication*
Erasure by Name *Wise Erasure*	Call-by-Name	Call-by-Need
Erasure by Value *Silly Erasure*	**Call-by-Silly**	Call-by-Value

Evaluation Strategies combining Name and Value

	Duplication by Name *Silly Duplication*	Duplication by Value *Wise Duplication*
Erasure by Name *Wise Erasure*	Call-by-Name	Call-by-Need
Erasure by Value *Silly Erasure*	**Call-by-Silly**	Call-by-Value

Evaluation Strategies combining Name and Value

	Duplication by Name *Silly Duplication*	Duplication by Value *Wise Duplication*
Erasure by Name *Wise Erasure*	Call-by-Name	Call-by-Need
Erasure by Value *Silly Erasure*	**Call-by-Silly**	Call-by-Value



Evaluation Strategies: duplication and erasure rules

Duplication by Name: $(\lambda x.t)u \mapsto_\beta t\{x \leftarrow u\}$ where $x \in \mathtt{fv}(t)$

Duplication by Value: $(\lambda x.t)v \mapsto_\beta t\{x \leftarrow v\}$ where $x \in \mathtt{fv}(t)$

Erasure by Name: $(\lambda x.t)u \mapsto_\beta t\{x \leftarrow u\}$ where $x \notin \mathtt{fv}(t)$

Erasure by Value: $(\lambda x.t)v \mapsto_\beta t\{x \leftarrow v\}$ where $x \notin \mathtt{fv}(t)$

Main results about Call-by-Silly:

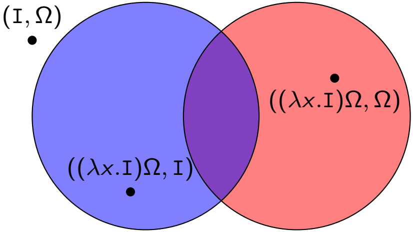
- Rewriting Properties and Multi Types
- CbSilly induces the same contextual equivalence than CbV
  - Mirroring the main theorem about CbNeed
  - Helpful to prove CbV contextual equivalence
  - CbV contextual equivalence is blind wrto efficiency
- Quantitative study of types: Call-by-Silly really is inefficient

Main results about Call-by-Silly:

- Rewriting Properties and Multi Types
- CbSilly induces the same contextual equivalence than CbV
  - Mirroring the main theorem about CbNeed
  - Helpful to prove CbV contextual equivalence
  - CbV contextual equivalence is blind wrto efficiency
- Quantitative study of types: Call-by-Silly really is inefficient

Contextual Equivalences

CbN Contextual Equivalence CbV Contextual Equivalence



$$\mathtt{I} =_{CbN} (\lambda x.\mathtt{I})\Omega =_{CbV} \Omega$$

$$\mathtt{I} \neq_{CbN} \Omega \neq_{CbV} \mathtt{I}$$

Contextual Equivalences

CbN Contextual Equivalence

⇕ Kesner [FoSSaCS 2016]

CbNeed Contextual Equivalence

CbV Contextual Equivalence

⇕ **This Paper**

CbSilly Contextual Equivalence

Through Intersection Types!

Contextual Equivalences

CbN Contextual Equivalence

Kesner [FoSSaCS 2016] $\Updownarrow$

CbNeed Contextual Equivalence

CbV Contextual Equivalence

$\Updownarrow$ **This Paper**

CbSilly Contextual Equivalence

Through Non Idempotent Intersection Types!

$\rightsquigarrow$ Quantitative Knowledge

Contextual Equivalences

CbN Contextual Equivalence $\Updownarrow$ Kesner [FoSSaCS 2016] CbNeed Contextual Equivalence

CbV Contextual Equivalence $\Updownarrow$ **This Paper** **CbSilly Contextual Equivalence**

Through Intersection Types!

$\rightsquigarrow$ Quantitative Knowledge

For Ctx Equivalence:

Duplication does not matter, Erasure matters.

Outline

Silly Substitution Calculus

$$
\begin{array}{rrcl}
\textsc{Terms} & t, u, s & ::= & x \mid \lambda x.t \mid tu \mid t[x{\leftarrow}u] \\
\textsc{Values} & v, v' & ::= & \lambda x.t \\
\textsc{Sub. Ctxs} & S, S' & ::= & \langle\cdot\rangle \mid S[x{\leftarrow}u] \\
\textsc{Weak contexts} & W & ::= & \langle\cdot\rangle \mid Wt \mid tW \mid t[x{\leftarrow}W] \mid W[x{\leftarrow}u]
\end{array}
$$

$$
\begin{array}{rll}
(\lambda x.t)u & \mapsto_{\mathtt{m}} & t[x{\leftarrow}u] \\
\ldots x \ldots [x{\leftarrow}u] & \mapsto_{\mathtt{e}} & \ldots u \ldots [x{\leftarrow}u] \\
t[x{\leftarrow}v] & \mapsto_{\mathtt{gcv}} & t \quad \text{if } x \notin \mathtt{fv}(t)
\end{array}
$$

$\rightarrow_{\mathtt{sil}}$ is the weak contextual closure of $\mapsto_{\mathtt{m}}$, $\mapsto_{\mathtt{e}_W}$ and $\mapsto_{\mathtt{gcv}}$.

Silly Substitution Calculus

$$\begin{array}{rrcl}
\textsc{Terms} & t, u, s & ::= & x \mid \lambda x.t \mid tu \mid t[x{\leftarrow}u] \\
\textsc{Values} & v, v' & ::= & \lambda x.t \\
\textsc{Sub. Ctxs} & S, S' & ::= & \langle\cdot\rangle \mid S[x{\leftarrow}u] \\
\textsc{Weak contexts} & W & ::= & \langle\cdot\rangle \mid Wt \mid tW \mid t[x{\leftarrow}W] \mid W[x{\leftarrow}u]
\end{array}$$

$$\begin{array}{rcl}
(\lambda x.t)u & \mapsto_{\mathtt{m}} & t[x{\leftarrow}u] \\
W\langle\!\langle x\rangle\!\rangle[x{\leftarrow}u] & \mapsto_{\mathtt{e}_W} & W\langle\!\langle u\rangle\!\rangle[x{\leftarrow}u] \\
t[x{\leftarrow}v] & \mapsto_{\mathtt{gcv}} & t \quad \text{if } x \notin \mathtt{fv}(t)
\end{array}$$

$\rightarrow_{\mathtt{sil}}$ is the weak contextual closure of $\mapsto_{\mathtt{m}}$, $\mapsto_{\mathtt{e}_W}$ and $\mapsto_{\mathtt{gcv}}$.

Silly Substitution Calculus

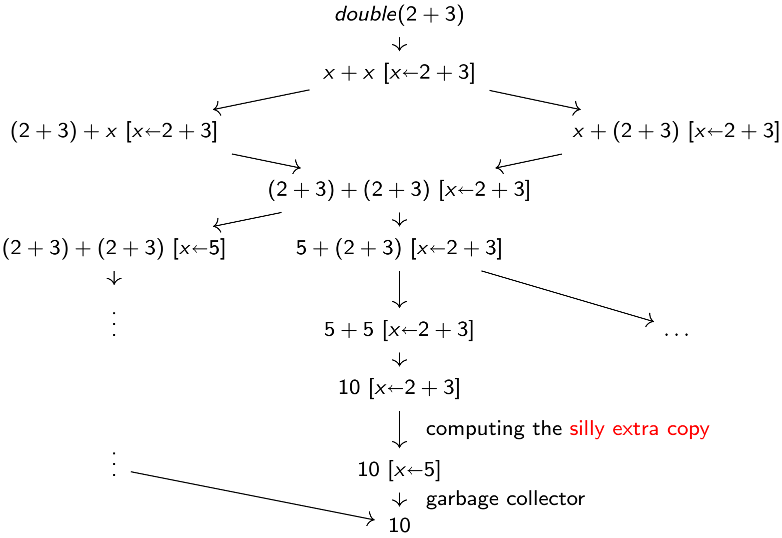
$$
\begin{array}{rrcl}
\textsc{Terms} & t, u, s & ::= & x \mid \lambda x.t \mid tu \mid t[x{\leftarrow}u] \\
\textsc{Values} & v, v' & ::= & \lambda x.t \\
\textsc{Sub. Ctxs} & S, S' & ::= & \langle\cdot\rangle \mid S[x{\leftarrow}u] \\
\textsc{Weak contexts} & W & ::= & \langle\cdot\rangle \mid Wt \mid tW \mid t[x{\leftarrow}W] \mid W[x{\leftarrow}u]
\end{array}
$$

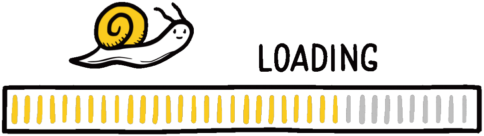
$$
\begin{array}{rcl}
S\langle\lambda x.t\rangle u & \mapsto_{\mathtt{m}} & S\langle t[x{\leftarrow}u]\rangle \\
W\langle\!\langle x\rangle\!\rangle[x{\leftarrow}u] & \mapsto_{\mathtt{e}_W} & W\langle\!\langle u\rangle\!\rangle[x{\leftarrow}u] \\
t[x{\leftarrow}S\langle v\rangle] & \mapsto_{\mathtt{gcv}} & S\langle t\rangle \quad \text{if } x \notin \mathtt{fv}(t)
\end{array}
$$

$\rightarrow_{\mathtt{sil}}$ is the weak contextual closure of $\mapsto_{\mathtt{m}}$, $\mapsto_{\mathtt{e}_W}$ and $\mapsto_{\mathtt{gcv}}$.

The Silly Extra Copy

Let $double := x \mapsto x + x$. How to silly compute $double(2+3)$?





Please wait while I compute $2 + 3$ again

Outline

Silly Multi Types

$$
\begin{array}{lrcl}
\textsc{Linear types} & L, L' & ::= & \mathtt{norm} \mid M \to L \\
\textsc{Multi types} & M, N & ::= & [L_i]_{i \in I} \quad \text{where } I \text{ is a finite set} \\
\textsc{Generic types} & T, T' & ::= & L \mid M
\end{array}
$$

$$
\frac{}{x : [L] \vdash x : L}\ \mathsf{ax} \qquad \frac{(\Gamma_i \vdash t : L_i)_{i \in I}}{\uplus_{i \in I} \Gamma_i \vdash t : [L_i]_{i \in I}}\ \mathsf{many}
$$

$$
\frac{}{\vdash \lambda x.t : \mathtt{norm}}\ \mathsf{ax}_\lambda \qquad \frac{\Gamma \vdash t : M \to L \qquad \Delta \vdash u : M \uplus [\mathtt{norm}]}{\Gamma \uplus \Delta \vdash tu : L}\ @
$$

$$
\frac{\Gamma \vdash t : L}{\Gamma \setminus\!\!\setminus x \vdash \lambda x.t : \Gamma(x) \to L}\ \lambda \qquad \frac{\Gamma \vdash t : L \qquad \Delta \vdash u : \Gamma(x) \uplus [\mathtt{norm}]}{(\Gamma \setminus\!\!\setminus x) \uplus \Delta \vdash t[x \leftarrow u] : L}\ \mathsf{ES}
$$

Example of type derivation

$\mathtt{n} := \mathtt{norm}$

$$\dfrac{\vdash \lambda y.yy : [[L]\to L, L, \mathtt{n}]\to L \qquad \dfrac{\vdash \mathtt{II} : [L]\to L \quad \vdash \mathtt{II} : L \quad \vdash \mathtt{II} : \mathtt{n} \quad \vdash \mathtt{II} : \mathtt{n}}{\vdash \mathtt{II} : [[L]\to L, L, \mathtt{n}] \uplus [\mathtt{n}]}}{\vdash (\lambda y.yy)(\mathtt{II}) : L}$$

Silly Multi Types and the SSC

Through standard methods for non idempotent intersection types, we get that typability is equivalent to silly termination.

Theorem

Let t be a term.

1. *Correctness: if $\pi \triangleright \Gamma \vdash t : L$ then t is $\to_{\mathtt{sil}}$-normalizing.*
2. *Completeness: if t is $\to_{\mathtt{sil}}$-normalizing then there exists $\pi \triangleright \Gamma \vdash t : \mathtt{norm}$.*

Outline

Closed Call-by-Value

$$\begin{array}{lrcl} \text{TERMS} & t, u, s & ::= & x \mid \lambda x.t \mid tu \\ \text{VALUES} & v, v' & ::= & \lambda x.t \end{array}$$

$$\frac{}{(\lambda x.t)v \to_{\beta_v} t\{x{\leftarrow}v\}} \quad \frac{t \to_{\beta_v} t'}{tu \to_{\beta_v} t'u} \quad \frac{t \to_{\beta_v} t'}{ut \to_{\beta_v} ut'}$$

Silly Multi Types and Closed CbV

Theorem

Let t be a closed term.

1. *Correctness: if $\pi \triangleright \ \vdash t : L$ then t is $\to_{\beta_v}$-normalizing.*
2. *Completeness: if t is $\to_{\beta_v}$-normalizing then there exists $\pi \triangleright \ \vdash t : \mathtt{norm}$.*

TFAE, for a closed term t:

- *Typability in Silly Multi Types: $\pi \triangleright \ \vdash t : L$*
- *CbV normalization: $t \to_{\beta_v}^{*} v$*
- *CbSilly normalization: $t \to_{\mathtt{sil}}^{*} v$*

Silly Multi Types and Closed CbV

Theorem

Let t be a closed term.

1. *Correctness: if $\pi \triangleright \ \vdash t : L$ then t is $\to_{\beta_v}$-normalizing.*
2. *Completeness: if t is $\to_{\beta_v}$-normalizing then there exists $\pi \triangleright \ \vdash t : \mathtt{norm}$.*

TFAE, for a closed term t:

- *Typability in Silly Multi Types: $\pi \triangleright \ \vdash t : L$*
- *CbV normalization: $t \to^*_{\beta_v} v$*
- *CbSilly normalization: $t \to^*_{\mathtt{sil}} v$*

Contextual Equivalences

Definition (Contextual Equivalence)

We define contextual equivalence $\simeq_C^s$ for a rewriting relation $\to_s$:

$t \simeq_C t'$ if for all C contexts such that $C\langle t\rangle$ and $C\langle t'\rangle$ are closed terms, $C\langle t\rangle$ is $\to_s$-normalizing iff $C\langle t'\rangle$ is $\to_s$-normalizing.

Let us consider $\simeq_C^{\beta_v}$, Plotkin's CbV contextual equivalence induced by $\to_{\beta_v}$ and $\simeq_C^{silly}$, the contextual equivalence induced by $\to_{\mathtt{sil}}$.

CbV and Silly induce the same contextual equivalence

Theorem

$\simeq_C^{\beta_v} = \simeq_C^{silly}$

Proof.

$t \simeq_C^{\beta_v} t' \iff$ For all C contexts such that $C\langle t\rangle$ and $C\langle t'\rangle$ are closed terms, $C\langle t\rangle$ is $\to_{\beta_v}$-normalizing iff $C\langle t'\rangle$ is $\to_{\beta_v}$-normalizing.

[On closed terms, $\to_{\beta_v}$-normalization is equivalent to Silly typability]

$\iff$ For all C contexts such that $C\langle t\rangle$ and $C\langle t'\rangle$ are closed terms, $\pi \triangleright \ \vdash C\langle t\rangle : L$ iff $\pi' \triangleright \ \vdash C\langle t'\rangle : L'$.

[(On all terms,) Silly typability is equivalent to silly $\to_{\mathtt{sil}}$-normalization]

$\iff$ For all C contexts such that $C\langle t\rangle$ and $C\langle t'\rangle$ are closed terms, $C\langle t\rangle$ is $\to_{\mathtt{sil}}$-normalizing iff $C\langle t'\rangle$ is $\to_{\mathtt{sil}}$-normalizing.

$\iff$ $t \simeq_C^{silly} t'$

CbSilly helps to prove CbV contextual equivalence

Let i any normal norm that does not look like an abstraction, for example $i = y\mathtt{I}$.
Consider the four following terms:

$$(\lambda x.xx)\, i \qquad (\lambda x.xi)\, i \qquad (\lambda x.ii)\, i \qquad ii$$

How to prove these terms are CbV contextually equivalent?

- $(\lambda x.xx)\, i$, $(\lambda x.xi)\, i$ and $(\lambda x.ii)\, i$ all reduce to the same silly normal form $ii[x{\leftarrow}i]$
- $(\lambda x.xx)\, i =_{\mathtt{sil}} (\lambda x.xi)\, i =_{\mathtt{sil}} (\lambda x.ii)\, i =_{\mathtt{sil}} ii[x{\leftarrow}i]$
- **Proposition:** if $t =_{\mathtt{sil}} u$ then $t \simeq_C^{silly} u$
- $(\lambda x.xx)\, i \simeq_C^{silly} (\lambda x.xi)\, i \simeq_C^{silly} (\lambda x.ii)\, i \simeq_C^{silly} ii[x{\leftarrow}i]$
- $(\lambda x.xx)\, i \simeq_C^{\beta_v} (\lambda x.xi)\, i \simeq_C^{\beta_v} (\lambda x.ii)\, i \simeq_C^{\beta_v} ii[x{\leftarrow}i]$

CbSilly helps to prove CbV contextual equivalence

Let i any normal norm that does not look like an abstraction, for example $i = y\mathtt{I}$.
Consider the four following terms:

$$(\lambda x.xx)\, i \qquad (\lambda x.xi)\, i \qquad (\lambda x.ii)\, i \qquad ii$$

How to prove these terms are CbV contextually equivalent?

- $(\lambda x.xx)\, i$, $(\lambda x.xi)\, i$ and $(\lambda x.ii)\, i$ all reduce to the same silly normal form $ii[x \leftarrow i]$
- $(\lambda x.xx)\, i =_{\mathtt{sil}} (\lambda x.xi)\, i =_{\mathtt{sil}} (\lambda x.ii)\, i =_{\mathtt{sil}} ii[x \leftarrow i]$
- **Proposition:** if $t =_{\mathtt{sil}} u$ then $t \simeq_C^{silly} u$
- $(\lambda x.xx)\, i \simeq_C^{silly} (\lambda x.xi)\, i \simeq_C^{silly} (\lambda x.ii)\, i \simeq_C^{silly} ii[x \leftarrow i]$
- $(\lambda x.xx)\, i \simeq_C^{\beta_v} (\lambda x.xi)\, i \simeq_C^{\beta_v} (\lambda x.ii)\, i \simeq_C^{\beta_v} ii[x \leftarrow i]$

CbSilly helps to prove CbV contextual equivalence

Let i any normal norm that does not look like an abstraction, for example $i = y\mathtt{I}$.
Consider the four following terms:

$$(\lambda x.xx)\, i \qquad (\lambda x.xi)\, i \qquad (\lambda x.ii)\, i \qquad ii$$

How to prove these terms are CbV contextually equivalent?

- $(\lambda x.xx)\, i$, $(\lambda x.xi)\, i$ and $(\lambda x.ii)\, i$ all reduce to the same silly normal form $ii[x{\leftarrow}i]$
- $(\lambda x.xx)\, i =_{\mathtt{sil}} (\lambda x.xi)\, i =_{\mathtt{sil}} (\lambda x.ii)\, i =_{\mathtt{sil}} ii[x{\leftarrow}i]$
- **Proposition:** if $t =_{\mathtt{sil}} u$ then $t \simeq_C^{silly} u$
- $(\lambda x.xx)\, i \simeq_C^{silly} (\lambda x.xi)\, i \simeq_C^{silly} (\lambda x.ii)\, i \simeq_C^{silly} ii[x{\leftarrow}i]$
- $(\lambda x.xx)\, i \simeq_C^{\beta_v} (\lambda x.xi)\, i \simeq_C^{\beta_v} (\lambda x.ii)\, i \simeq_C^{\beta_v} ii[x{\leftarrow}i]$

Outline

Conclusion

In the paper:

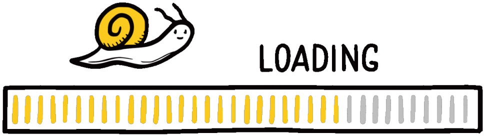
- Introduce a degenerated reduction, call-by-silly
- Mirroring the main theorem about CbN and CbNeed: call-by-silly and call-by-value generate the same contextual equivalence
- Helps to prove CbV contextual equivalence: convertibility in call-by-silly
- Tight types and maximality: the type system exactly measures the reduction length of a call-by-silly strategy. We can also show thanks to the type system that this strategy is maximal in some sense.

Future work:

- Refining CbV contextual equivalence to forbid silly duplications and be aware of efficiency
- Categorical or Game Semantics for CbSilly and CbNeed?

Thank you for your attention!

	Duplication by Name *Silly Duplication*	Duplication by Value *Wise Duplication*
Erasure by Name *Wise Erasure*	Call-by-Name	Call-by-Need
Erasure by Value *Silly Erasure*	**Call-by-Silly**	Call-by-Value



Please wait while I compute $2 + 3$ again

The problem with variable as values

$$x[z\leftarrow ww] \;{}_{\texttt{wgcv}}\!\leftarrow\; x[y\leftarrow z][z\leftarrow ww] \to_{\texttt{e}_W} x[y\leftarrow ww][z\leftarrow ww]$$

Similar issues arise in CbNeed:

$$\begin{array}{ccc} x(\lambda z.wx)[x\leftarrow y][y\leftarrow \mathtt{I}] & \longrightarrow & x(\lambda z.wx)[x\leftarrow \mathtt{I}][y\leftarrow \mathtt{I}] \\ \downarrow & & \vdots \\ y(\lambda z.wx)[x\leftarrow y][y\leftarrow \mathtt{I}] \dashrightarrow \mathtt{I}(\lambda z.wx)[x\leftarrow y][y\leftarrow \mathtt{I}] & & \mathtt{I}(\lambda z.wx)[x\leftarrow \mathtt{I}][y\leftarrow \mathtt{I}] \end{array}$$

The problem with variable as values

$$x[z\leftarrow ww] \;{}_{\texttt{wgcv}}\!\leftarrow\; x[y\leftarrow z][z\leftarrow ww] \to_{\texttt{e}_W} x[y\leftarrow ww][z\leftarrow ww]$$

Similar issues arise in CbNeed:

$$\begin{array}{ccc}
x(\lambda z.wx)[x\leftarrow y][y\leftarrow \texttt{I}] & \longrightarrow & x(\lambda z.wx)[x\leftarrow \texttt{I}][y\leftarrow \texttt{I}] \\
\downarrow & & \vdots \\
y(\lambda z.wx)[x\leftarrow y][y\leftarrow \texttt{I}] \dashrightarrow \texttt{I}(\lambda z.wx)[x\leftarrow y][y\leftarrow \texttt{I}] & & \texttt{I}(\lambda z.wx)[x\leftarrow \texttt{I}][y\leftarrow \texttt{I}]
\end{array}$$

Why Silly Types do not work for Open CbV

For $n \geq 1$, we have the following derivation for the source term $z((\lambda x.y)u)$ in the silly type system:

$$\dfrac{\pi_z \rhd \ \ldots \qquad \dfrac{\left(\dfrac{\dfrac{\dfrac{}{y:[A] \vdash y:A}\,\mathsf{ax}}{y:[A] \vdash \lambda x.y : 0 \to A}\,\lambda \qquad \dfrac{\pi_u \rhd \ \Gamma \vdash u:\texttt{norm}}{\Gamma \vdash u:[\texttt{norm}]}\,\mathsf{many}}{(y:[A], \Gamma \vdash (\lambda x.y)u : A)_{i=1,\ldots,n}}\,@\right) \qquad \dfrac{\dfrac{\dfrac{}{y:[\texttt{norm}] \vdash y:\texttt{norm}}\,\mathsf{ax}}{y:[\texttt{norm}] \vdash \lambda x.y : 0 \to \texttt{norm}}\,\lambda \qquad \ldots}{y:[\texttt{norm}], \Gamma \vdash (\lambda x.y)u : \texttt{no}\ldots}}{y:[A^n, \texttt{norm}], \Gamma^{n+1} \vdash (\lambda x.y)u : [A^n, \texttt{norm}]}}{z:[[A^n, \texttt{norm}] \to B], y:[A^n, \texttt{norm}], \Gamma^{n+1} \vdash z((\lambda x.y)u) : B}\,@$$

where $\pi_z \rhd \ \ldots$ stands for:

$$\dfrac{}{\pi_z \rhd \ z:[[A^n] \to B] \vdash z:[A^n] \to B}\,\mathsf{ax}$$

The term $(\lambda x.zy)u$, instead, can only be typed as follows, the key point being that Γ^{n+1} is replaced by Γ:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z:[[A^n] \to B] \vdash z:[A^n] \to B}\,\mathsf{ax} \qquad \dfrac{\dfrac{}{(y:[A] \vdash y:A)_{i=1,\ldots,n}}\,\mathsf{ax} \qquad \dfrac{}{y:[\texttt{norm}] \vdash y:\texttt{norm}}\,\mathsf{ax}}{y:[A^n, \texttt{norm}] \vdash y:[A^n, \texttt{norm}]}\,\mathsf{many}}{z:[[A^n] \to B], y:[A^n, \texttt{norm}] \vdash zy : B}\,@}{z:[[A^n] \to B], y:[A^n, \texttt{norm}] \vdash \lambda x.zy : 0 \to B}\,\lambda \qquad \dfrac{\pi_u \rhd \ \Gamma \vdash u:\ldots}{\Gamma \vdash u:[\texttt{no}\ldots}}{z:[[A^n] \to B], y:[A^n, \texttt{norm}], \Gamma \vdash (\lambda x.zy)u : B}$$

The Call-by-Silly Strategy

CALL-BY-NAME STRATEGY $\to_{\mathtt{n}}$

NAME CTXS $N, N' ::= \langle\cdot\rangle \mid Nt \mid N[x{\leftarrow}t]$	$\to_{\mathtt{nm}} := N\langle\mapsto_{\mathtt{m}}\rangle$
ROOT GC $t[x{\leftarrow}s] \mapsto_{\mathtt{gc}} t$ if $x \notin \mathtt{fv}(t)$	$\to_{\mathtt{ne}} := N\langle\mapsto_{\mathtt{e}_N}\rangle$
	$\to_{\mathtt{ngc}} := N\langle\mapsto_{\mathtt{gc}}\rangle$
	$\to_{\mathtt{n}} := \to_{\mathtt{nm}} \cup \to_{\mathtt{ne}} \cup \to_{\mathtt{ngcv}}$

CALL-BY-SILLY STRATEGY $\to_{\mathtt{y}}$

ANSWERS $a, a' ::= v \mid a[x{\leftarrow}a']$	$\to_{\mathtt{ym}} := Y\langle\mapsto_{\mathtt{m}}\rangle$
AUX. CTXS $A, A' ::= \langle\cdot\rangle \mid a[x{\leftarrow}A] \mid A[x{\leftarrow}t]$	$\to_{\mathtt{ygcv}} := Y\langle\mapsto_{\mathtt{gcv}}\rangle$
SILLY CTXS $Y, Y' ::= A\langle N\rangle$	$\to_{\mathtt{ye_{AY}}} := A\langle\mapsto_{\mathtt{e}_Y}\rangle$
	$\to_{\mathtt{ye_{YN}}} := Y\langle\mapsto_{\mathtt{e}_N}\rangle$
	$\to_{\mathtt{y}} := \to_{\mathtt{ym}} \cup \to_{\mathtt{ye_{AY}}} \cup \to_{\mathtt{ye_{YN}}} \cup \to_{\mathtt{ygcv}}$

Figure 4 The call-by-name and call-by-silly strategies.

The Call-by-Silly Strategy: example

CbN evaluation:

$(\lambda y.yy)(\mathtt{II})$

$\to_{\mathtt{ym}}$ $yy[y\leftarrow \mathtt{II}]$		$\to_{\mathtt{ye_{YN}}}$ $(\mathtt{II})y[y\leftarrow \mathtt{II}]$
$\to_{\mathtt{ym}}$ $(x[x\leftarrow \mathtt{I}])y[y\leftarrow \mathtt{II}]$		$\to_{\mathtt{ye_{YN}}}$ $(\mathtt{I}[x\leftarrow \mathtt{I}])y[y\leftarrow \mathtt{II}]$
$\to_{\mathtt{ym}}$ $z[z\leftarrow y][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{II}]$		$\to_{\mathtt{ye_{YN}}}$ $y[z\leftarrow y][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{II}]$
		$\to_{\mathtt{ye_{YN}}}$ $\mathtt{II}[z\leftarrow y][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{II}]$
$\to_{\mathtt{ym}}$ $z'[z'\leftarrow \mathtt{I}][z\leftarrow y][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{II}]$		$\to_{\mathtt{ye_{YN}}}$ $\mathtt{I}[z'\leftarrow \mathtt{I}][z\leftarrow y][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{II}]$

CbS extension:

	$\to_{\mathtt{ye_{AY}}}$ $\mathtt{I}[z'\leftarrow \mathtt{I}][z\leftarrow \mathtt{II}][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{II}]$
$\to_{\mathtt{ym}}$ $\mathtt{I}[z'\leftarrow \mathtt{I}][z\leftarrow z'[z'\leftarrow \mathtt{I}]][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{II}]$	$\to_{\mathtt{ye_{YN}}}$ $\mathtt{I}[z'\leftarrow \mathtt{I}][z\leftarrow \mathtt{I}[z'\leftarrow \mathtt{I}]][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{II}]$
$\to_{\mathtt{ym}}$ $\mathtt{I}[z'\leftarrow \mathtt{I}][z\leftarrow \mathtt{I}[z'\leftarrow \mathtt{I}]][x\leftarrow \mathtt{I}][y\leftarrow z'[z'\leftarrow \mathtt{I}]]$	$\to_{\mathtt{ye_{YN}}}$ $\mathtt{I}[z'\leftarrow \mathtt{I}][z\leftarrow \mathtt{I}[z'\leftarrow \mathtt{I}]][x\leftarrow \mathtt{I}][y\leftarrow \mathtt{I}[z'\leftarrow \mathtt{I}]]$