

On the reduction of the type-free computational λ -calculus

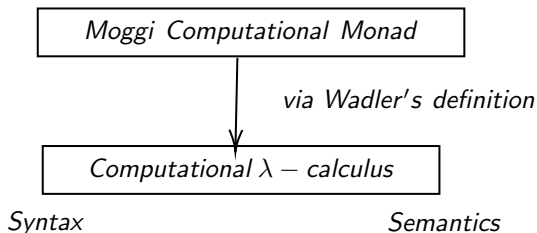
Ugo de'Liguoro,Riccardo Treglia

University of Turin

IWC 2020 30/06/2020



Overview



Calculus

Reduction relation

Contexts

Confluence

Factorization

Intersection type assignment

Subject convertibility

Monadic Interpretation

D_∞ model

Soundness and Completeness via filter model

We study a notion of **reduction** and prove it is **confluent** and **factorizes**; also we relate our calculus to the original work by Moggi.

Computational Monads (after Wadler)

A *monad* is a triple (T, unit, \star)

Types

D is the type of **values**;

TD is the type of **computations** (with **effects**) over D .

Operators

$\text{unit}_D : D \rightarrow TD$ (Haskell: **return**);

$\star_{D,E} : TD \rightarrow (D \rightarrow TE) \rightarrow TE$ (Haskell: $>>=$).

Axioms

$$(\text{unit } d) \star f = f d$$

$$a \star \text{unit} = a$$

$$(a \star f) \star g = a \star \lambda d. (f d \star g)$$

Untyped computational λ -calculus: λ_c^u

We consider Moggi's call-by-value reflexive object:

$$D = D \rightarrow TD$$

hence there are two types, D and TD , and two kinds of terms:

$$\text{Val} : \quad V, W ::= x \mid \lambda x.M \quad (\text{values})$$

$$\text{Com} : \quad L, M, N ::= \text{unit } V \mid M \star V \quad (\text{computations})$$

having types:

$$\begin{array}{c} x : D \vdash x : D \\ \\ \frac{\vdash V : D}{\vdash \text{unit } V : TD} \end{array} \qquad \begin{array}{c} \frac{x : D \vdash M : TD}{\vdash \lambda x.M : D \rightarrow TD = D} \\ \\ \frac{\vdash M : TD \quad \vdash V : D = D \rightarrow TD}{\vdash M \star V : TD} \end{array}$$

Reduction

Orienting monad axioms we get $\longrightarrow \subseteq Com \times Com$ as the compatible closure of:

$$\beta_c \quad unit\ V \star (\lambda x.M) \longrightarrow M[V/x]$$

$$id \quad M \star \lambda x.unit\ x \longrightarrow M$$

$$ass \quad (L \star \lambda x.M) \star \lambda y.N \longrightarrow L \star \lambda x.(M \star \lambda y.N) \quad \text{for } x \notin FV(N)$$

where $M[V/x]$ denotes the capture avoiding substitution of V for all free occurrences of x in M .

To have extensionality we add:

$$(\eta_c) \quad unit\ \lambda x.(unit\ x \star V) \longrightarrow unit\ V \quad \text{if } x \notin FV(V)$$

λ_c^u versus type free λ_c : translation

let is definable:

$$\text{let } x = M \text{ in } N \equiv M \star \lambda x. N$$

No primitive functional application but

$$(\lambda x. M) V \equiv (\text{unit } V) \star (\lambda x. M)$$

$$VM \equiv M \star V$$

$$MN \equiv M \star \lambda z. (N \star z) \text{ for some fresh } z$$

By this we retrieve ordinary call-by-value reduction rule:

$$(\beta_v) \quad (\lambda x. M) V \equiv (\text{unit } V) \star (\lambda x. M) \longrightarrow_{\beta_c} M[V/x]$$

λ_c^u versus type free λ_c : translation

let is definable:

$$\text{let } x = M \text{ in } N \equiv M \star \lambda x.N$$

No primitive functional application but

$$(\lambda x.M)V \equiv (\text{unit } V) \star (\lambda x.M)$$

$$VM \equiv M \star V$$

$$MN \equiv M \star \lambda z.(N \star z) \text{ for some fresh } z$$

Bonus track: *shuffling calculus*

$$V((\lambda x.L)N) \mapsto_{\sigma_3} (\lambda x.VL)N$$

that translated

$$(N \star \lambda x.L) \star V \mapsto_{\sigma_3} N \star \lambda x.(L \star V)$$

and $(N \star \lambda x.L) \star V \mapsto_{\text{ass}} N \star \lambda x.(L \star V)$ if V is a variable

λ_c^u versus type free λ_c : mutual interpretation

Theorem

There exists an interpretation $\llbracket \cdot \rrbracket$ from λ_c into λ_c^u that preserves **reductions**.

There exists an interpretation $\ulcorner \cdot \urcorner$ from λ_c^u into λ_c that preserves the **convertibility relation**.

The reduction relation λ_c is **confluent**.

This has been shown by Makoto Hamana, *Polymorphic Rewrite Rules: Confluence, Type Inference, and Instance Validation*, 2018.

It is done by checking critical pair using his tool.

Confluence

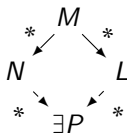


Reduction in λ_c^u has **three** basic rules whose left-hand sides may **overlap** (not orthogonal).

We split the proof in three steps:

- 1+2. proving confluence of $\beta_c \cup id$ and ass separately,
3. eventually combining by means of the commutativity of these relations.

Confluence



Reduction in λ_c^u has **three** basic rules whose left-hand sides may **overlap** (not orthogonal).

We split the proof in three steps:

- 1+2. proving confluence of $\beta_c \cup id$ and *ass* separately,
3. eventually combining by means of the commutativity of these relations.

Adopted strategy:

- call-by-need calculi with the *let* construct (see [2, 6])
- variant of call-by-value λ -calculus in [5],

$\longrightarrow_{\beta_c, id}$ confluence

Define the following relation $\dashv\vdash$, as in the book [7] ch. 10.

Definition

The relation $\dashv\vdash \subseteq \text{Term} \times \text{Term}$ is inductively defined by:

- 1 $x \dashv\vdash x$
- 2 $M \dashv\vdash N \Rightarrow \lambda x. M \dashv\vdash \lambda x. N$
- 3 $V \dashv\vdash V' \Rightarrow \text{unit } V \dashv\vdash \text{unit } V'$
- 4 $M \dashv\vdash M' \text{ and } V \dashv\vdash V' \Rightarrow M \star V \dashv\vdash M' \star V'$
- 5 $M \dashv\vdash M' \text{ and } V \dashv\vdash V' \Rightarrow \text{unit } V \star \lambda x. M \dashv\vdash M'[V'/x]$
- 6 $M \dashv\vdash M' \Rightarrow M \star \lambda x. \text{unit } x \dashv\vdash M'$

Lemma (Substitutivity lemma)

For $M, M' \in \text{Com}$ and $V, V' \in \text{Val}$ and every variable x ,
if $M \dashv\vdash M'$ and $V \dashv\vdash V'$, then $M[V/x] \dashv\vdash M'[V'/x]$.

$\longrightarrow_{\beta_c, id}$ confluence

Define the following relation $\dashv\vdash$, as in the book [7] ch. 10.

Definition

The relation $\dashv\vdash \subseteq \text{Term} \times \text{Term}$ is inductively defined by:

- 1 $x \dashv\vdash x$
- 2 $M \dashv\vdash N \Rightarrow \lambda x. M \dashv\vdash \lambda x. N$
- 3 $V \dashv\vdash V' \Rightarrow \text{unit } V \dashv\vdash \text{unit } V'$
- 4 $M \dashv\vdash M' \text{ and } V \dashv\vdash V' \Rightarrow M \star V \dashv\vdash M' \star V'$
- 5 $M \dashv\vdash M' \text{ and } V \dashv\vdash V' \Rightarrow \text{unit } V \star \lambda x. M \dashv\vdash M'[V'/x]$
- 6 $M \dashv\vdash M' \Rightarrow M \star \lambda x. \text{unit } x \dashv\vdash M'$

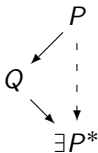
Lemma (Substitutivity lemma)

For $M, M' \in \text{Com}$ and $V, V' \in \text{Val}$ and every variable x ,
if $M \dashv\vdash M'$ and $V \dashv\vdash V'$, then $M[V/x] \dashv\vdash M'[V'/x]$.

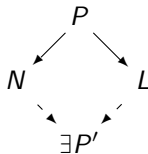
$$\longrightarrow_{\beta_c, id} \subseteq \dashv\vdash \subseteq \overset{*}{\longrightarrow}_{\beta_c, id}$$

→ _{β_c, id} confluence

The next step in the proof is to show that the relation \rightarrow satisfies the *triangle property TP*:



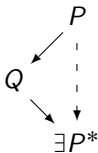
TP implies the *diamond property*



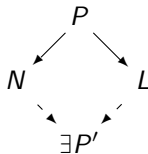
In fact, if *TP* holds then we can take $P' \equiv P^*$ in *DP*, since the latter only depends on P . We then define P^* in terms of P as defining an adapted version of Takahashi translation.

$\longrightarrow_{\beta_c, id}$ confluence

The next step in the proof is to show that the relation \longrightarrow satisfies the *triangle property TP*:



TP implies the *diamond property*



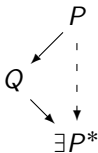
In fact, if *TP* holds then we can take $P' \equiv P^*$ in *DP*, since the latter only depends on P . We then define P^* in terms of P as defining an adapted version of Takahashi translation.

Lemma

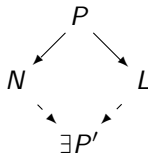
For all $P, Q \in \text{Term}$, if $P \longrightarrow Q$ then $Q \longrightarrow P^*$, namely \longrightarrow satisfies *TP*.

$\longrightarrow_{\beta_c, id}$ confluence

The next step in the proof is to show that the relation \longrightarrow satisfies the *triangle property TP*:



TP implies the *diamond property*



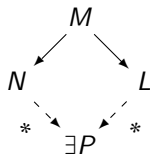
In fact, if *TP* holds then we can take $P' \equiv P^*$ in *DP*, since the latter only depends on P . We then define P^* in terms of P as defining an adapted version of Takahashi translation.

Proposition

$\longrightarrow_{\beta_c, id}$ is confluent.

ass confluence

To prove confluence of *ass*, we use Newman Lemma (see [4], Prop. 3.1.24). A notion of reduction R is *weakly Church-Rosser*, shortly *WCR*, if



Lemma

The notion of reduction ass is WCR.

ass confluence

Recall that a notion of reduction R is *strongly normalizing*, shortly *SN*, if there exists **no** infinite reduction $M \rightarrow_R M_1 \rightarrow_R M_2 \rightarrow_R \dots$ out of any $M \in Com$.

Lemma

The notion of reduction ass is SN.

ass confluence

Recall that a notion of reduction R is *strongly normalizing*, shortly *SN*, if there exists **no** infinite reduction $M \rightarrow_R M_1 \rightarrow_R M_2 \rightarrow_R \dots$ out of any $M \in Com$.

Lemma

The notion of reduction *ass* is SN.

Proof idea

Given $M \in Com$. Let's decorate each occurrence of \star in M with a natural number.

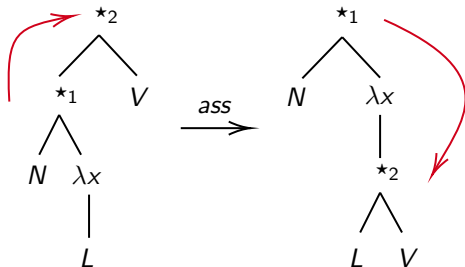
Define \star_i left to \star_j in M iff

- $\exists (L \star_j V)$ subterm of M
- \star_i occurs in L

$$\#M \stackrel{def}{=} |\{(x_i, x_j) \mid x_i \text{ left to } x_j\}|$$

It's easy to see that

$$\#M > \#N \text{ if } M \rightarrow_{ass} N$$



ass confluence

Corollary

The notion of reduction ass is CR.

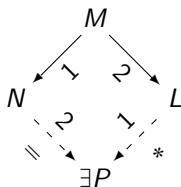
Proof.

By the previous lemma and by Newman Lemma, stating that a notion of reduction which is WCR and SN is CR . □

$\longrightarrow_{\beta_c, id}$ and \longrightarrow_{ass} commute

The following definitions are from [3], Def. 2.7.9.

Relations \longrightarrow_1 and \longrightarrow_2 *strongly commute* if, for all M, N, L :



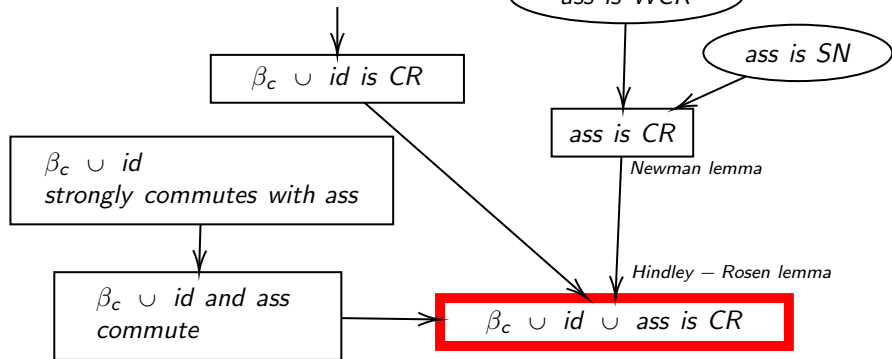
where $\overset{=}{\longrightarrow}_2$ is $\longrightarrow_2 \cup =$, namely at most one reduction step.

Lemma

Reductions $\longrightarrow_{\beta_c, id}$ and \longrightarrow_{ass} strongly commute, then commute.

Complete proof recap

Parallel reduction $\left\{ \begin{array}{l} \longrightarrow_{\beta_c, id} \subseteq \longrightarrow \subseteq \xrightarrow{*} \beta_c, id \\ \text{Takahashi translation } \longrightarrow \text{ has TP} \end{array} \right.$



Factorization

An **abstract reduction system** $(Term, \longrightarrow)$ factorizes via $\longrightarrow_e, \longrightarrow_{\neg e}$ if

$$\longrightarrow = \longrightarrow_e \cup \longrightarrow_{\neg e}$$

and for all $M, N \in Term$, $M \xrightarrow{*} N$ implies that there exists $L \in Term$ such that

$$M \xrightarrow{*}_e L \xrightarrow{*}_{\neg e} N.$$

We abbreviate the last condition by $M \xrightarrow{*}_e \cdot \xrightarrow{*}_{\neg e} N$.

Factorized reduction example

$$\begin{array}{l}
 M \equiv \frac{((unit \lambda y.unit y) \star \lambda w.(unit \lambda x.unit x)) \star \lambda z.unit z}{unit \lambda y.unit y \star \lambda w.(unit \lambda x.unit x \star \lambda z.unit z)} \quad \xrightarrow{\text{ass}} \\
 \frac{unit \lambda y.unit y \star \lambda w.(unit \lambda x.unit x \star \lambda z.unit z)}{unit \lambda x.unit x} \quad \xrightarrow{\beta_c} \\
 M \equiv \frac{((unit \lambda y.unit y) \star \lambda w.(unit \lambda x.unit x)) \star \lambda z.unit z}{(unit \lambda x.unit x \star \lambda z.unit z)} \quad \xrightarrow{\beta_c} \\
 unit \lambda x.unit x \quad \xrightarrow{\beta_c}
 \end{array}$$

Essential vs. Inessential Contexts

Take $\longrightarrow = \longrightarrow_{\lambda C}$

Construct the relations \longrightarrow_e , $\longrightarrow_{\neg e}$, called the *essential* and *inessential* in [1], by closing $\mapsto_{\lambda C}$ under two sorts of contexts:

$$\begin{array}{l} \text{Inessential contexts:} \quad \neg \mathcal{E} ::= \langle \cdot \rangle_C \mid \text{unit } \lambda x. \neg \mathcal{E} \mid M \star \lambda x. \neg \mathcal{E} \mid \neg \mathcal{E} \star V \\ \text{Essential contexts:} \quad \mathcal{E} ::= \langle \cdot \rangle_C \mid \mathcal{E} \star V \end{array}$$

where the hole $\langle \cdot \rangle_C$ can be filled by terms in *Com* only.

Then \longrightarrow_e and $\longrightarrow_{\neg e}$ are the least relations including $\mapsto_{\lambda C}$ such that for all $M, N \in \text{Com}$:

$$M \mapsto_{\lambda C} N \implies \mathcal{E} \langle M \rangle \longrightarrow_e \mathcal{E} \langle N \rangle \quad \text{and} \quad M \mapsto_{\lambda C} N \implies \neg \mathcal{E} \langle M \rangle \longrightarrow_{\neg e} \neg \mathcal{E} \langle N \rangle$$

We highlight that relations \longrightarrow_e and $\longrightarrow_{\neg e}$ are actually **not disjoint**, as *essential* steps are also *inessential*.

Techniques in between

One of the strengths of the method in [1] is the fact that it does not rely on the *DP* of the parallel reduction $\Rightarrow_{\lambda C}$ but on the **substitutivity property** of an indexed refined $\xRightarrow{n-1}_{\lambda C}$

Definition (Indexed parallel reduction)

The relation $\xRightarrow{n} \subseteq \text{Term} \times \text{Term}$ is inductively defined by:

- 1 $x \xRightarrow{0} x$
- 2 $M \xRightarrow{n} N \Rightarrow \lambda x. M \xRightarrow{n} \lambda x. N$
- 3 $V \xRightarrow{n} V' \Rightarrow \text{unit } V \xRightarrow{n} \text{unit } V'$
- 4 $M \xRightarrow{n} M' \text{ and } V \xRightarrow{m} V' \Rightarrow M * V \xRightarrow{n+m} M' * V'$
- 5 $M \xRightarrow{n} M' \text{ and } V \xRightarrow{m} V' \Rightarrow \text{unit } V * \lambda x. M \xRightarrow{n+|M'|_x \cdot m+1} M'[V'/x]$
- 6 $M \xRightarrow{n} M' \Rightarrow M * \lambda x. \text{unit } x \xRightarrow{n} M'$
- 7 $L \xRightarrow{n} L' \text{ and } M \xRightarrow{m} M' \text{ and } N \xRightarrow{p} N' \Rightarrow (L * \lambda x. M) * \lambda y. N \xRightarrow{n+m+p} L' * \lambda x. (M' * \lambda y. N')$

and of another auxiliary relations $\Rightarrow_{\neg e}$, roughly the parallel reduction built over $\longrightarrow_{\neg e}$

Factorization proof

Proposition ($\lambda\mathbf{C}$ Macro-step system)

- 1 *Merge:* if $M \Rightarrow_{\neg e} \cdot \longrightarrow_e M'$ then $M \Rightarrow_{\lambda\mathbf{C}} M'$
- 2 *Indexed split:* if $M \xRightarrow{n} M'$, then $M \Rightarrow_{\neg e} M'$, or $n > 0$ and $M \longrightarrow_e \cdot \xRightarrow{n-1} M'$
- 3 *Split:* If $M \Rightarrow_{\lambda\mathbf{C}} M'$, then $M \xrightarrow{*}_e \cdot \Rightarrow_{\neg e} M'$.

Then $(Term, \longrightarrow_e \cup \longrightarrow_{\neg e})$ is a **macro-step system** with respect to $\Rightarrow_{\lambda\mathbf{C}}$ and $\Rightarrow_{\neg e}$.

Factorization proof

Proposition ($\lambda\mathbf{C}$ Macro-step system)

- 1 Merge: if $M \Rightarrow_{\neg e} \cdot \longrightarrow_e M'$ then $M \Rightarrow_{\lambda\mathbf{C}} M'$
- 2 Indexed split: if $M \xrightarrow{n} M'$, then $M \Rightarrow_{\neg e} M'$, or $n > 0$ and $M \longrightarrow_e \cdot \xrightarrow{n-1} M'$
- 3 Split: If $M \Rightarrow_{\lambda\mathbf{C}} M'$, then $M \xrightarrow{*}_e \cdot \Rightarrow_{\neg e} M'$.

Then $(Term, \longrightarrow_e \cup \longrightarrow_{\neg e})$ is a **macro-step system** with respect to $\Rightarrow_{\lambda\mathbf{C}}$ and $\Rightarrow_{\neg e}$.

By Split and Merge

$$\begin{aligned} \Rightarrow_{\lambda\mathbf{C}} \subseteq \xrightarrow{*}_e \cdot \Rightarrow_{\neg e} \\ \Rightarrow_{\neg e} \cdot \longrightarrow_e \subseteq \xrightarrow{*}_e \cdot \Rightarrow_{\neg e} \end{aligned}$$

By construction

$$\begin{aligned} \Rightarrow_{\lambda\mathbf{C}} \subseteq \xrightarrow{*}_{\lambda\mathbf{C}} \\ \Rightarrow_{\neg e} \subseteq \xrightarrow{*}_{\neg e} \end{aligned}$$

We have

$$(\longrightarrow_e \cup \longrightarrow_{\neg e})^* \subseteq \xrightarrow{*}_e \cdot \Rightarrow_{\neg e}^* \subseteq \xrightarrow{*}_e \cdot \xrightarrow{*}_{\neg e}$$

Factorization proof

Proposition ($\lambda\mathbf{C}$ Macro-step system)

- 1 Merge: if $M \Rightarrow_{\neg e} \cdot \longrightarrow_e M'$ then $M \Rightarrow_{\lambda\mathbf{C}} M'$
- 2 Indexed split: if $M \xRightarrow{n} M'$, then $M \Rightarrow_{\neg e} M'$, or $n > 0$ and $M \longrightarrow_e \cdot \xRightarrow{n-1} M'$
- 3 Split: If $M \Rightarrow_{\lambda\mathbf{C}} M'$, then $M \xrightarrow{*}_e \cdot \Rightarrow_{\neg e} M'$.

Then $(Term, \longrightarrow_e \cup \longrightarrow_{\neg e})$ is a **macro-step system** with respect to $\Rightarrow_{\lambda\mathbf{C}}$ and $\Rightarrow_{\neg e}$.

Every Macro-step system satisfies factorization:

Theorem (Factorization)

The reduction system $(Term, \longrightarrow_{\lambda\mathbf{C}})$ factorizes via $\longrightarrow_e, \longrightarrow_{\neg e}$

$$M \xrightarrow{*}_{\lambda\mathbf{C}} M' \Rightarrow M \xrightarrow{*}_e \cdot \xrightarrow{*}_{\neg e} M'$$

Factorization proof

Proposition ($\lambda\mathbf{C}$ Macro-step system)

- 1 Merge: if $M \Rightarrow_{\neg e} \cdot \longrightarrow_e M'$ then $M \Rightarrow_{\lambda\mathbf{C}} M'$
- 2 Indexed split: if $M \xRightarrow{n} M'$, then $M \Rightarrow_{\neg e} M'$, or $n > 0$ and $M \longrightarrow_e \cdot \xRightarrow{n-1} M'$
- 3 Split: If $M \Rightarrow_{\lambda\mathbf{C}} M'$, then $M \xrightarrow{*}_e \cdot \Rightarrow_{\neg e} M'$.

Then $(Term, \longrightarrow_e \cup \longrightarrow_{\neg e})$ is a **macro-step system** with respect to $\Rightarrow_{\lambda\mathbf{C}}$ and $\Rightarrow_{\neg e}$.

Every Macro-step system satisfies factorization:

Theorem (Factorization)

The reduction system $(Term, \longrightarrow_{\lambda\mathbf{C}})$ factorizes via $\longrightarrow_e, \longrightarrow_{\neg e}$

$$M \xrightarrow{*}_{\lambda\mathbf{C}} M' \Rightarrow M \xrightarrow{*}_e \cdot \xrightarrow{*}_{\neg e} M'$$

Thanks



Beniamino Accattoli, Claudia Faggian, and Giulio Guerrieri.

Factorization and normalization, essentially.

In *APLAS 2019: Programming Languages and Systems*, volume 11893 of *Lecture Notes in Computer Science*, page 159–180. Springer Verlag, 12 2019.



Zena M. Ariola, Matthias Felleisen, John Maraist, Martin Odersky, and Philip Wadler.

The call-by-need lambda calculus.

In Ron K. Cytron and Peter Lee, editors, *Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23-25, 1995*, pages 233–246. ACM Press, 1995.



Franz Baader and Tobias Nipkow.

Term rewriting and all that.

Cambridge University Press, 1998.



H. Barendregt.

The Lambda Calculus: its Syntax and Semantics.

North-Holland, Amsterdam, revised edition, 1984.



Alberto Carraro and Giulio Guerrieri.

A semantical and operational account of call-by-value solvability.

In Anca Muscholl, editor, *Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014*, volume 8412 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 2014.



John Maraist, Martin Odersky, David N. Turner, and Philip Wadler.

Call-by-name, call-by-value, call-by-need and the linear lambda calculus.

Theor. Comput. Sci., 228(1-2):175–210, 1999.



Terese.

Term Rewriting Systems, volume 55 of *Cambridge Tracts in Theoretical Computer Science*.

Cambridge University Press, 2003.