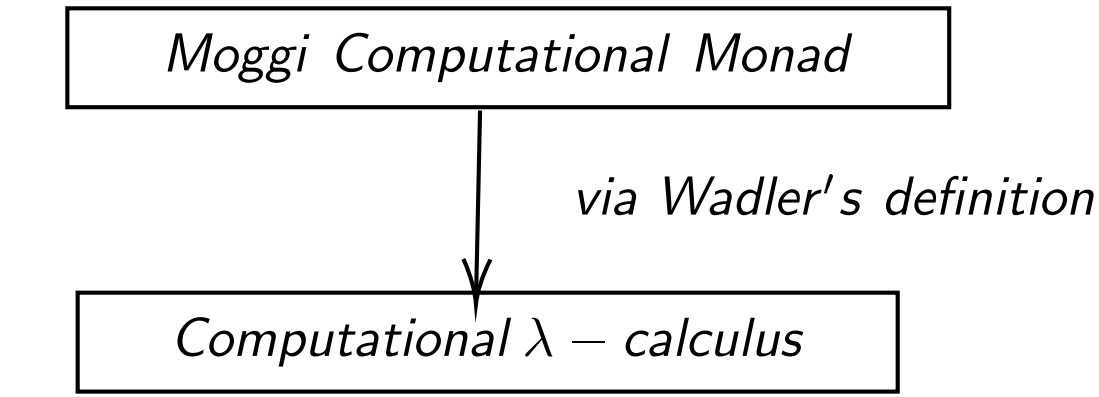
On the reduction of the type-free computational λ-calculus

Ugo de'Liguoro, Riccardo Treglia

University of Turin, Italy

IWC 2020 30/06/2020

Overview



Calculus	*Intersection type assignment*
Reduction relation	*Subject convertibility*
Contexts	*Monadic Interpretation*
Confluence	D_∞ *model*
Factorization	*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.

Why effects in functional languages

Why effects in functional languages

- Semantic theories are an established field providing foundations to software analysis and verification

Why effects in functional languages

- Semantic theories are an established field providing foundations to software analysis and verification
- We have nice theories of the semantics for functional programming languages, but procedural languages commonly used for both sequential and concurrent programming have non-functional features, collectively called **effects**

Why effects in functional languages

- Semantic theories are an established field providing foundations to software analysis and verification
- We have nice theories of the semantics for functional programming languages, but procedural languages commonly used for both sequential and concurrent programming have non-functional features, collectively called **effects**
- Since Algol'60 project various λ-**calculi** have been used as metalanguages providing semantics to programming languages in general, and a successful approach among others is based on the notion of **computational monad** [Moggi89, 91]

Why effects in functional languages

- Semantic theories are an established field providing foundations to software analysis and verification
- We have nice theories of the semantics for functional programming languages, but procedural languages commonly used for both sequential and concurrent programming have non-functional features, collectively called **effects**
- Since Algol'60 project various λ-**calculi** have been used as metalanguages providing semantics to programming languages in general, and a successful approach among others is based on the notion of **computational monad** [Moggi89, 91]
- **Monads** have been studied and implemented mainly for typed languages; here we address the issue of **modelling effects in the untyped case**.

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

$unit_D : D \to TD$ (Haskell: **return**);

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

Axioms

$$(unit\ d) \star f = f\ d$$

$$a \star unit = a$$

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

Untyped computational λ-calculus: λ_c^u

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

$$D = D \to TD$$

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

$$\begin{array}{rrcll} \mathit{Val}: & V, W & ::= & x \mid \lambda x.M & \text{(values)} \\ \mathit{Com}: & L, M, N & ::= & \mathit{unit}\ V \mid M \star V & \text{(computations)} \end{array}$$

having types:

$$x : D \vdash x : D \qquad \frac{x : D \vdash M : TD}{\vdash \lambda x.M : D \to TD = D}$$

$$\frac{\vdash V : D}{\vdash \mathit{unit}\ V : TD} \qquad \frac{\vdash M : TD \quad \vdash V : D = D \to TD}{\vdash M \star V : TD}$$

Reduction

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

$$\begin{array}{lrcll} \beta_c & \mathit{unit}\ V \star (\lambda x.M) & \longrightarrow & M[V/x] & \\ \mathit{id} & M \star \lambda x.\mathit{unit}\ x & \longrightarrow & M & \\ \mathit{ass} & (L \star \lambda x.M) \star \lambda y.N & \longrightarrow & L \star \lambda x.(M \star \lambda y.N) & \text{for } x \notin FV(N) \end{array}$$

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

To have extensionality we add:

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

λ_c^u versus type free λ_c: translation

let is definable:

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

No primitive functional application but

$$(\lambda x.M)\,V \equiv (unit\ V) \star (\lambda x.M)$$

$$VM \equiv M \star V$$

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

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

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

λ_c^u versus type free λ_c: translation

let is definable:

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

No primitive functional application but

$$(\lambda x.M)\,V \equiv (unit\ V) \star (\lambda x.M)$$

$$VM \equiv M \star V$$

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

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 $(N \star \lambda x.L) \star V \mapsto\!\!\!\!+_{ass} N \star \lambda x.(L \star V)$ if V is a variable

λ_c^u versus type free λ_c: mutual interpretation

Theorem

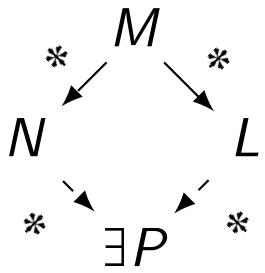
There exists an interpretation $\lfloor \cdot \rfloor$ *from* λ_C *into* λ_c^u *that preserves* **reductions**.

There exists an interpretation $\lceil \cdot \rceil$ *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



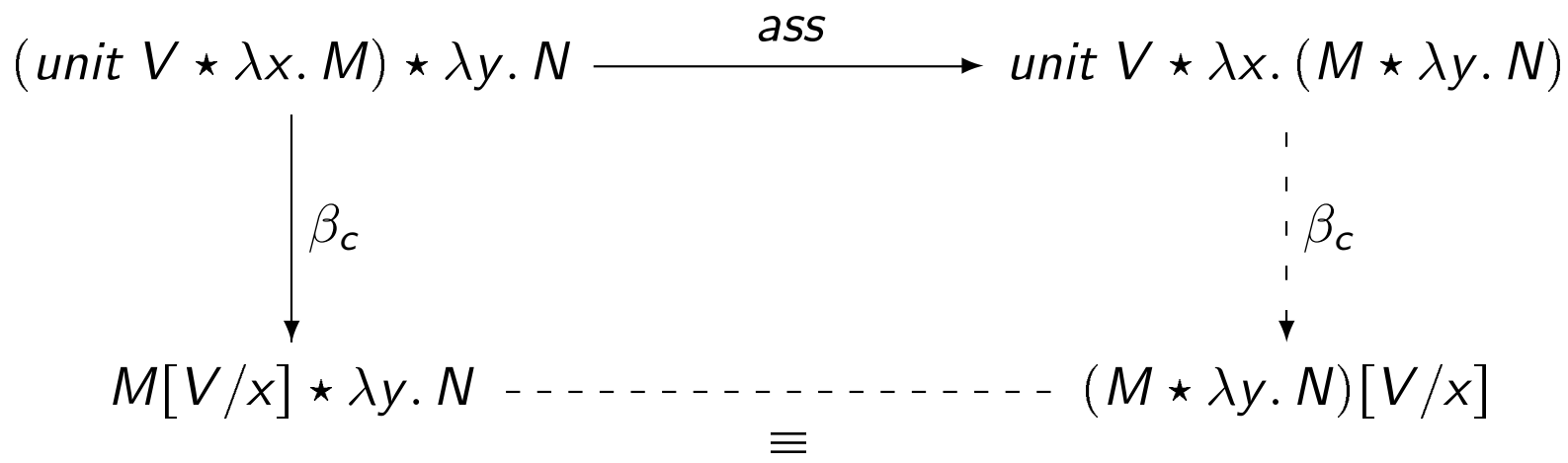
Reduction in λ_c^u has **three** basic rules whose left-hand sides may **overlap** (not orthogonal).
We split the proof in three steps:
1+2. proving confluence of $\beta_c \cup id$ and *ass* separately,
3. eventually combining by means of the commutativity of these relations.

Adopted strategy:

- call-by-need calculi with the *let* construct (see [AFM+95, MOTW99])
- variant of call-by-value λ-calculus in [CG14],

Example

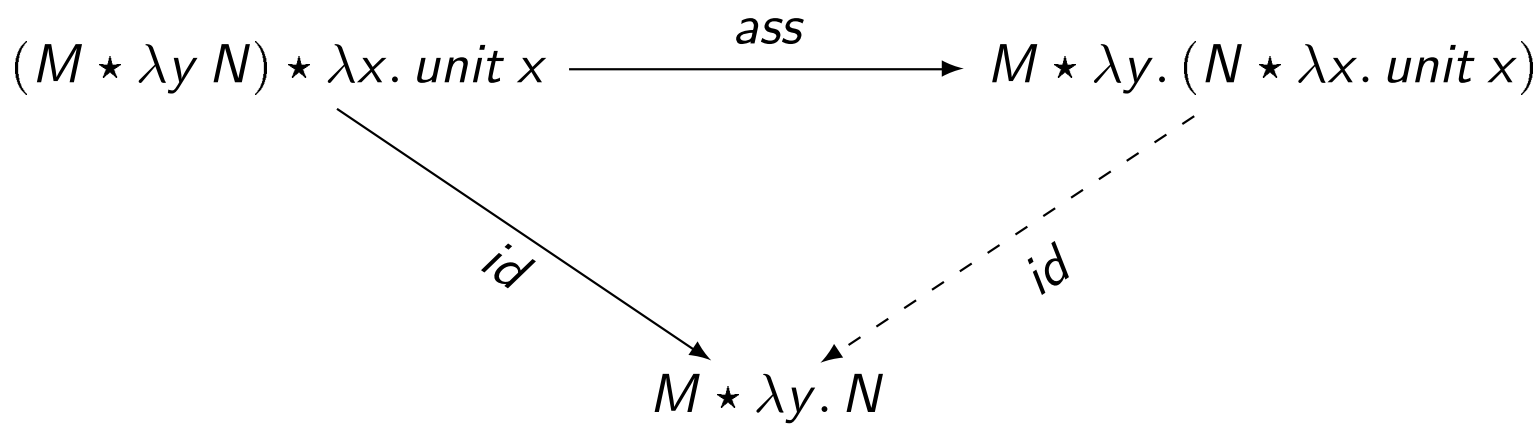
Example



where $x \notin FV(N)$, which is the side condition to rule *ass*; therefore the two terms in the lower line of the diagram are syntactically identical.

Example

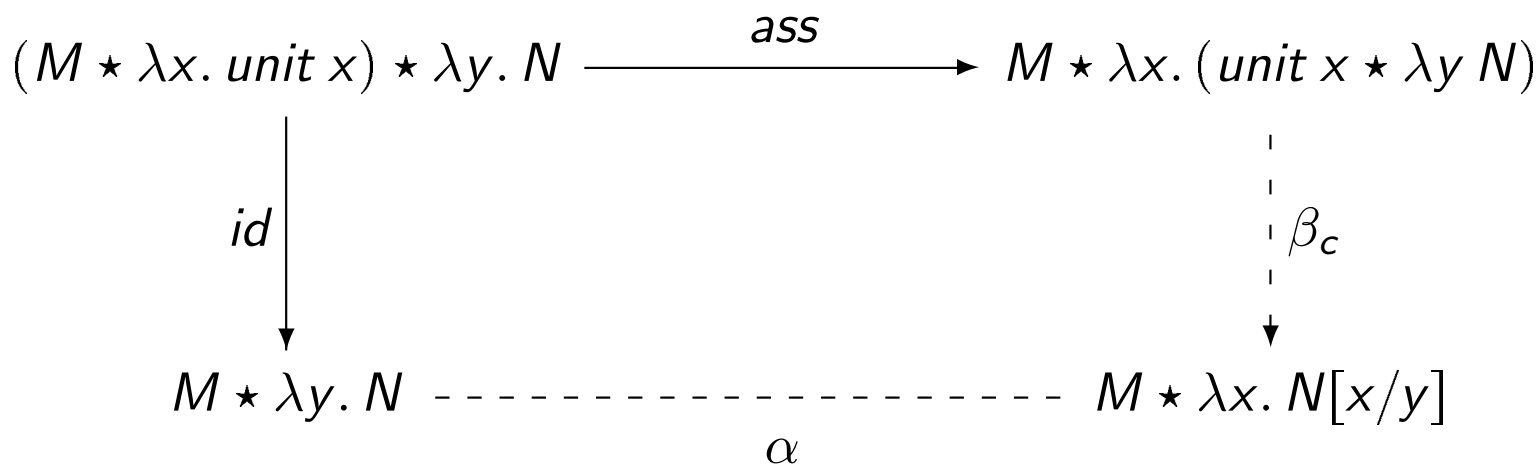
Example



Example

Example

Here the outer reduction by *ass* overlaps with an inner reduction by *id*. This is recovered by means of an inner reduction by β_c:



$\longrightarrow_{\beta_c,id}$ confluence

Define the following relation $\multimap\!\!\rightarrow$, as in the book [Ter03] ch. 10.

Definition

The relation $\multimap\!\!\rightarrow \subseteq \mathit{Term} \times \mathit{Term}$ is inductively defined by:

1. $x \multimap\!\!\rightarrow x$
2. $M \multimap\!\!\rightarrow N \Rightarrow \lambda x.M \multimap\!\!\rightarrow \lambda x.N$
3. $V \multimap\!\!\rightarrow V' \Rightarrow \mathit{unit}\ V \multimap\!\!\rightarrow \mathit{unit}\ V'$
4. $M \multimap\!\!\rightarrow M'$ and $V \multimap\!\!\rightarrow V' \Rightarrow M \star V \multimap\!\!\rightarrow M' \star V'$
5. $M \multimap\!\!\rightarrow M'$ and $V \multimap\!\!\rightarrow V' \Rightarrow \mathit{unit}\ V \star \lambda x.M \multimap\!\!\rightarrow M'[V'/x]$
6. $M \multimap\!\!\rightarrow M' \Rightarrow M \star \lambda x.\mathit{unit}\ x \multimap\!\!\rightarrow M'$

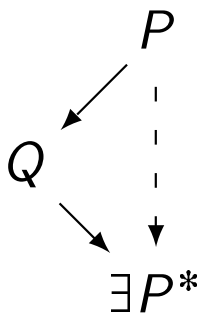
Lemma (Substitutivity lemma)

For $M, M' \in \mathit{Com}$ and $V, V' \in \mathit{Val}$ and every variable x,
if $M \multimap\!\!\rightarrow M'$ and $V \multimap\!\!\rightarrow V'$, then $M[V/x] \multimap\!\!\rightarrow M'[V'/x]$.

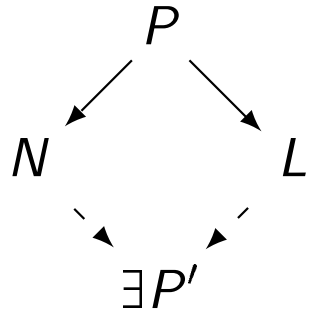
$$\longrightarrow_{\beta_c,id} \subseteq \multimap\!\!\rightarrow \subseteq \overset{*}{\longrightarrow}_{\beta_c,id}$$

$\longrightarrow_{\beta_c, id}$ confluence

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



TP implies the *diamond property*



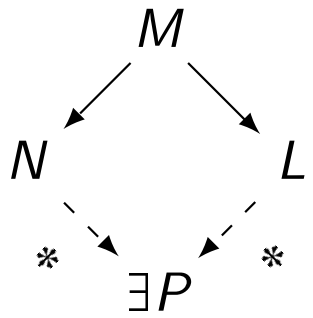
In fact, if TP holds then we can take $P' \equiv P^*$ in DP, since the latter only depends on P. We then define P^* in terms of P as defining an adapted version of Takahashi translation.

Lemma

For all $P, Q \in Term$, if $P \multimap\!\!\!\rightarrow Q$ then $Q \multimap\!\!\!\rightarrow P^*$, namely $\multimap\!\!\!\rightarrow$ satisfies TP.

ass confluence

To prove confluence of *ass*, we use Newman Lemma (see [Bar84], 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 $M \longrightarrow_R M_1 \longrightarrow_R M_2 \longrightarrow_R \cdots$ out of any $M \in Com$.

Lemma

The notion of reduction ass is SN.

Proof idea

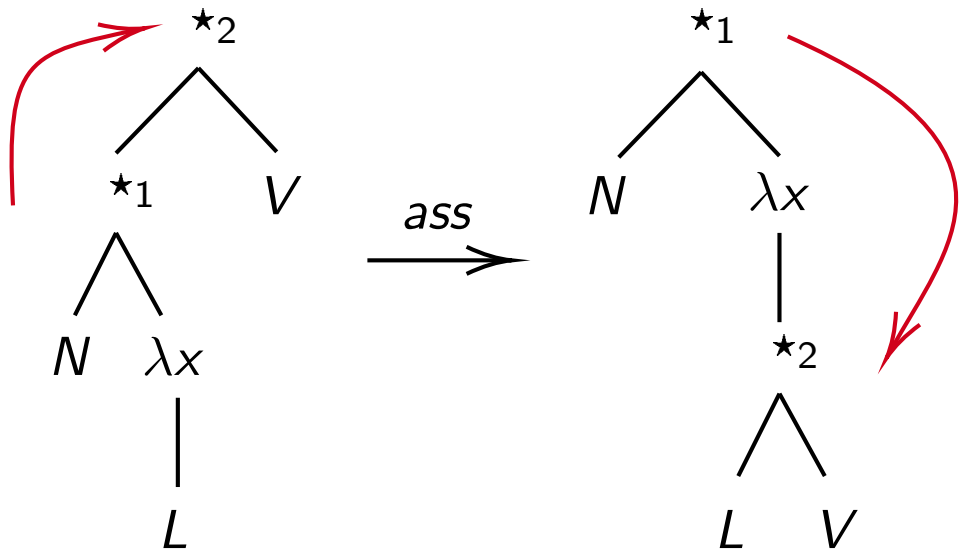
Given $M \in Com$. Let's decorate each occurrence of $\star$ in M with a natural number.

Define $\star_i$ left to $\star_j$ in M iff

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

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

It's easy to see that
$\sharp M > \sharp N$ if $M \longrightarrow_{ass} N$



ass confluence

Corollary

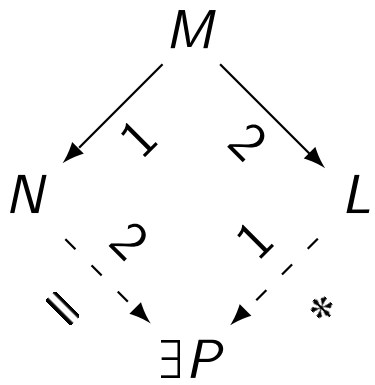
The notion of reduction ass is CR.

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 [BN98], Def. 2.7.9.
Relations $\longrightarrow_1$ and $\longrightarrow_2$ *strongly commute* if, for all M, N, L:

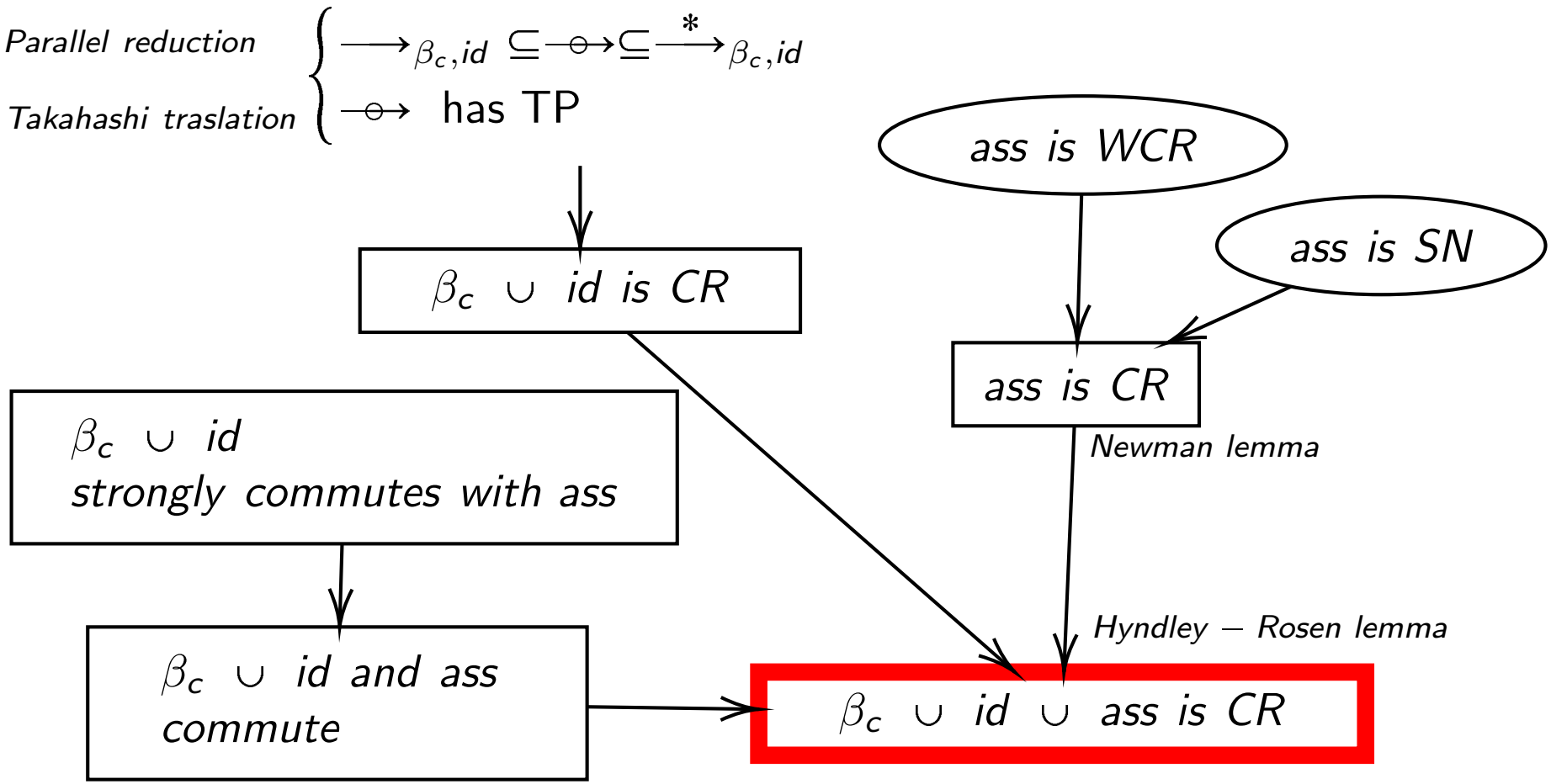


where $\overset{=}{\longrightarrow}_2$ is $\longrightarrow_2 \cup =$, namely at most one reduction step.

Lemma

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

Complete proof recap



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

$$
\begin{array}{ll}
M \equiv \underline{((unit\ \lambda y.unit\ y) \star \lambda w.(unit\ \lambda x.unit\ x)) \star \lambda z.unit\ z} & \longrightarrow_{ass} \\
unit\ \lambda y.unit\ y \star \lambda w.\underline{(unit\ \lambda x.unit\ x \star \lambda z.unit\ z)} & \longrightarrow_{\beta_c} \\
\underline{unit\ \lambda y.unit\ y \star \lambda w.(unit\ \lambda z.unit\ z)} & \longrightarrow_{\beta_c} \\
unit\ \lambda x.unit\ x &
\end{array}
$$

$$
\begin{array}{ll}
M \equiv \underline{((unit\ \lambda y.unit\ y) \star \lambda w.(unit\ \lambda x.unit\ x))} \star \lambda z.unit\ z & \longrightarrow_{\beta_c} \\
\underline{(unit\ \lambda x.unit\ x \star \lambda z.unit\ z)} & \longrightarrow_{\beta_c} \\
unit\ \lambda x.unit\ x &
\end{array}
$$

Essential vs. Inessential Contexts

Take $\longrightarrow = \longrightarrow_{\lambda \mathbf{C}}$
Construct the relations $\longrightarrow_e, \longrightarrow_{\neg e}$, called the *essential* and *inessential* in [AFG19], by closing $\mapsto_{\lambda \mathbf{C}}$ under two sorts of contexts:

$$\begin{array}{rrcl} \textit{Inessential contexts:} & \neg\mathcal{E} & ::= & \langle \cdot_{\mathrm{C}} \rangle \mid \mathit{unit}\ \lambda x.\neg\mathcal{E} \mid M \star \lambda x.\neg\mathcal{E} \mid \neg\mathcal{E} \star V \\ \textit{Essential contexts:} & \mathcal{E} & ::= & \langle \cdot_{\mathrm{C}} \rangle \mid \mathcal{E} \star V \end{array}$$

where the hole $\langle \cdot_{\mathrm{C}} \rangle$ can be filled by terms in *Com* only.

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

$$M \mapsto_{\lambda \mathbf{C}} N \Longrightarrow \mathcal{E}\langle M \rangle \longrightarrow_e \mathcal{E}\langle N \rangle \qquad \text{and} \qquad M \mapsto_{\lambda \mathbf{C}} N \Longrightarrow \neg\mathcal{E}\langle M \rangle \longrightarrow_{\neg e} \neg\mathcal{E}\langle N \rangle$$

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 [AFG19] is the fact that it does not rely on the *DP* of the parallel reduction $\Rightarrow_{\lambda \mathbf{C}}$ but on the **substitutivity property** of an indexed refined $\overset{n-1}{\Longrightarrow}_{\lambda \mathbf{C}}$

Definition (Indexed parallel reduction)

The relation $\overset{n}{\Rightarrow} \subseteq \mathit{Term} \times \mathit{Term}$ is inductively defined by:

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

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* $M \overset{n}{\Rightarrow} M'$, *then* $M \Rightarrow_{\neg e} M'$, *or* $n > 0$ *and* $M \longrightarrow_e \cdot \overset{n-1}{\Longrightarrow} M'$
3. *Split: If* $M \Rightarrow_{\lambda\mathbf{C}} M'$, *then* $M \overset{*}{\longrightarrow}_e \cdot \Rightarrow_{\neg e} M'$.

Then $(\mathit{Term}, \longrightarrow_e \cup \longrightarrow_{\neg e})$ 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* $M \overset{n}{\Rightarrow} M'$, *then* $M \Rightarrow_{\neg e} M'$, *or* $n > 0$ *and* $M \longrightarrow_e \cdot \overset{n-1}{\Longrightarrow} M'$
3. *Split: If* $M \Rightarrow_{\lambda\mathbf{C}} M'$, *then* $M \overset{*}{\longrightarrow}_e \cdot \Rightarrow_{\neg e} M'$.

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

By Split and Merge

$\Rightarrow_{\lambda\mathbf{C}} \subseteq \overset{*}{\longrightarrow}_e \cdot \Rightarrow_{\neg e}$

$\Rightarrow_{\neg e} \cdot \longrightarrow_e \subseteq \overset{*}{\longrightarrow}_e \cdot \Rightarrow_{\neg e}$

By construction

$\Rightarrow_{\lambda\mathbf{C}} \subseteq \overset{*}{\longrightarrow}_{\lambda\mathbf{C}}$

$\Rightarrow_{\neg e} \subseteq \overset{*}{\longrightarrow}_{\neg e}$

We have

$$(\longrightarrow_e \cup \longrightarrow_{\neg e})^* \subseteq \overset{*}{\longrightarrow}_e \cdot \Rightarrow^*_{\neg e} \subseteq \overset{*}{\longrightarrow}_e \cdot \overset{*}{\longrightarrow}_{\neg e}$$

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* $M \overset{n}{\Rightarrow} M'$, *then* $M \Rightarrow_{\neg e} M'$, *or* $n > 0$ *and* $M \longrightarrow_e \cdot \overset{n-1}{\Longrightarrow} M'$
3. *Split: If* $M \Rightarrow_{\lambda\mathbf{C}} M'$, *then* $M \overset{*}{\longrightarrow}_e \cdot \Rightarrow_{\neg e} M'$.

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* $\longrightarrow_e, \longrightarrow_{\neg e}$

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

References I

Z.M. Ariola and Arvind.
Properties of a First-order Functional Language with Sharing.
Theoretical Computer Science, 146:69–108, 1995.

E. Albert, P. Arenas, M. Codish, S. Genaim, G. Puebla, and D. Zanardini.
Termination Analysis of Java Bytecode.
In G. Barthe and F. de Boer, editors, *Proceedings of Formal Methods for Open Object-Based Distributed Systems (FMOODS'08), Oslo, Norway, 2008*, volume 5051 of *Lecture Notes in Computer Science*, pages 2–18. Springer Verlag, 2008.

Arvind, L. Augusston, J Hicks, R. S. Nikhil, S. Peyton-Jones, J. Stoy, and W Williams.
pH: A Parallel Haskell.
Technical report, MIT Laboratory for Computer Science, September 1993.

F. Alessi, F. Barbanera, and M. Dezani-Ciancaglini.
Intersection Types and Computational Rules.
Electronic Notes in Theoretical Computer Science, 84, 2003.