Quantitative Types for Useful Reduction

Joint work with Delia Kesner and Mariana Milicich

11th International Workshop on Higher Order Rewriting (HOR 2023)

July 4th, 2023

Pablo Barenbaum

(Invited talk)

Outline

The λ-calculus — some questions

The λ-calculus can express all computable functions.

The λ-calculus — some questions

The λ-calculus can express all computable functions.

The End

The λ-calculus — some questions

The λ-calculus can express all computable functions.

The End ?

The λ-calculus — some questions

The λ-calculus can express all computable functions.

~~*The End*~~

Can we implement β-reduction efficiently?

The λ-calculus — some questions

The λ-calculus can express all computable functions.

~~*The End*~~

Can we implement β-reduction efficiently?

Evaluation strategies, abstract machines, optimal reduction, **sharing**, ...

The λ-calculus — some questions

The λ-calculus can express all computable functions.

~~*The End*~~

Can we implement β-reduction efficiently?

Evaluation strategies, abstract machines, optimal reduction, **sharing**, ...

Can we statically guarantee dynamic properties of λ-terms?

The λ-calculus — some questions

The λ-calculus can express all computable functions.

~~*The End*~~

Can we implement β-reduction efficiently?

Evaluation strategies, abstract machines, optimal reduction, **sharing**, ...

Can we statically guarantee dynamic properties of λ-terms?

Type systems: polymorphic/dependent/refinement/**intersection**/... types

The λ-calculus — some questions

The λ-calculus can express all computable functions.

~~*The End*~~

Can we implement β-reduction efficiently?

Evaluation strategies, abstract machines, optimal reduction, **sharing**, …

Can we statically guarantee dynamic properties of λ-terms?

Type systems: polymorphic/dependent/refinement/**intersection**/… types

Can we measure complexity directly in the λ-calculus?

The λ-calculus — some questions

The λ-calculus can express all computable functions.

~~The End~~

Can we implement β-reduction efficiently?

Evaluation strategies, abstract machines, optimal reduction, **sharing**, ...

Can we statically guarantee dynamic properties of λ-terms?

Type systems: polymorphic/dependent/refinement/**intersection**/... types

Can we measure complexity directly in the λ-calculus?

Implicit computational complexity, cost semantics, **useful reduction**, ...

The λ-calculus — some questions

The λ-calculus can express all computable functions.

~~The End~~

Can we implement β-reduction efficiently?

Evaluation strategies, abstract machines, optimal reduction, **sharing**, ...

Can we statically guarantee dynamic properties of λ-terms?

Type systems: polymorphic/dependent/refinement/**intersection**/... types

Can we measure complexity directly in the λ-calculus?

Implicit computational complexity, cost semantics, **useful reduction**, ...

...

Reasonable cost models

Van Emde Boas' *Invariance Thesis*

There is a standard class of machine models that are able to simulate each other with

polynomial	overhead in	**time**
constant	overhead in	**space**

Reasonable cost models

Van Emde Boas' *Invariance Thesis*

There is a standard class of machine models that are able to simulate each other with

polynomial	overhead in	**time**
constant	overhead in	**space**

These are called reasonable models of computation.

Reasonable cost models

Van Emde Boas' *Invariance Thesis*

There is a standard class of machine models that are able to simulate each other with

polynomial	overhead in	**time**
constant	overhead in	**space**

These are called reasonable models of computation.

Reasonable models include Turing machines and RAMs.

Reasonable cost models

Van Emde Boas' *Invariance Thesis*

There is a standard class of machine models that are able to simulate each other with

polynomial	overhead in	**time**
constant	overhead in	**space**

These are called reasonable models of computation.

Reasonable models include Turing machines and RAMs.

Is the λ-calculus reasonable?

`Lawall, Mairson, Asperti, Guerrini, Dal Lago, Accattoli, ...`

Reasonable cost models

Van Emde Boas' *Invariance Thesis*
There is a standard class of machine models that are able to simulate each other with

polynomial	overhead in	**time**
constant	overhead in	**space**

These are called reasonable models of computation.

Reasonable models include Turing machines and RAMs.

Is the λ-calculus reasonable?

`Lawall, Mairson, Asperti, Guerrini, Dal Lago, Accattoli, ...`

(In this talk: we focus on **time** complexity).

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

$$t_0 \stackrel{\text{def}}{=} z \qquad t_{n+1} \stackrel{\text{def}}{=} \Delta\, t_n \qquad \text{where } \Delta \stackrel{\text{def}}{=} \lambda x.\, x\, x$$

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

$$t_0 \stackrel{\text{def}}{=} z \qquad t_{n+1} \stackrel{\text{def}}{=} \Delta\, t_n \qquad \text{where } \Delta \stackrel{\text{def}}{=} \lambda x.\, x\, x$$

$$t_1 = \Delta\, z \qquad \to z\, z$$

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

$$t_0 \stackrel{\text{def}}{=} z \qquad t_{n+1} \stackrel{\text{def}}{=} \Delta\, t_n \qquad \text{where } \Delta \stackrel{\text{def}}{=} \lambda x.\, x\, x$$

$$\begin{array}{lll} t_1 = \Delta\, z & \to z\, z & \\ t_2 = \Delta\, (\Delta\, z) & \to \Delta\, (z\, z) & \to z\, z\, (z\, z) \end{array}$$

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

$$t_0 \stackrel{\text{def}}{=} z \qquad t_{n+1} \stackrel{\text{def}}{=} \Delta\, t_n \qquad \text{where } \Delta \stackrel{\text{def}}{=} \lambda x.\, x\, x$$

$$\begin{aligned}
t_1 &= \Delta\, z && \to z\, z \\
t_2 &= \Delta\, (\Delta\, z) && \to \Delta\, (z\, z) \to z\, z\, (z\, z) \\
t_3 &= \Delta\, (\Delta\, (\Delta\, z)) && \to \Delta\, (\Delta\, (z\, z)) \to \Delta\, (z\, z\, (z\, z)) \to z\, z\, (z\, z)\, (z\, z\, (z\, z))
\end{aligned}$$

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

$$t_0 \stackrel{\text{def}}{=} z \qquad t_{n+1} \stackrel{\text{def}}{=} \Delta\, t_n \qquad \text{where } \Delta \stackrel{\text{def}}{=} \lambda x.\, x\, x$$

$$\begin{array}{llllll}
t_1 = \Delta\, z & \to z\, z & & & \\
t_2 = \Delta\,(\Delta\, z) & \to \Delta\,(z\, z) & \to z\, z\,(z\, z) & & \\
t_3 = \Delta\,(\Delta\,(\Delta\, z)) & \to \Delta\,(\Delta\,(z\, z)) & \to \Delta\,(z\, z\,(z\, z)) & \to z\, z\,(z\, z)\,(z\, z\,(z\, z)) \\
\vdots & & & &
\end{array}$$

t_n is of size $\Theta(n)$ and reduces in n steps to a term of size $\Theta(2^n)$.

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

What does this mean?

- In Turing machines, space can grow at most linearly with time.

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

What does this mean?

- In Turing machines, space can grow at most linearly with time.
- So TMs **cannot** simulate β-reduction with polynomial overhead...

Is the λ-calculus reasonable?

Can n steps of β-reduction be simulated in $O(n^k)$ time?

The size explosion problem

Terms may grow exponentially with β-reduction.

What does this mean?

- In Turing machines, space can grow at most linearly with time.
- So TMs **cannot** simulate β-reduction with polynomial overhead...

  ...as long as one represents λ-terms naively as trees.
- But one can use better representations for λ-terms.

  `Accattoli, Dal Lago,` *et al.*`'s work`

  `More to come...`

Evaluation strategies

The number of steps to normal form depends on the evaluation strategy.

Example

$\mathtt{I} \stackrel{\text{def}}{=} \lambda x.\, x$

Evaluation strategies

The number of steps to normal form depends on the evaluation strategy.

Example

$\mathtt{I} \stackrel{\text{def}}{=} \lambda x.\, x$

Call-by-name takes 4 steps to evaluate $(\lambda x.\, x\, x)\,(\mathtt{I}\,\mathtt{I})$

$$(\lambda x.\, x\, x)\,(\mathtt{I}\,\mathtt{I}) \to \mathtt{I}\,\mathtt{I}\,(\mathtt{I}\,\mathtt{I}) \to \mathtt{I}\,(\mathtt{I}\,\mathtt{I}) \to \mathtt{I}\,\mathtt{I} \to \mathtt{I}$$

Evaluation strategies

The number of steps to normal form depends on the evaluation strategy.

Example

$\mathtt{I} \stackrel{\text{def}}{=} \lambda x.\, x$

Call-by-name takes 4 steps to evaluate $(\lambda x.\, x\, x)\,(\mathtt{I}\,\mathtt{I})$

$$(\lambda x.\, x\, x)\,(\mathtt{I}\,\mathtt{I}) \to \mathtt{I}\,\mathtt{I}\,(\mathtt{I}\,\mathtt{I}) \to \mathtt{I}\,(\mathtt{I}\,\mathtt{I}) \to \mathtt{I}\,\mathtt{I} \to \mathtt{I}$$

Call-by-value takes only 3 steps

$$(\lambda x.\, x\, x)(\mathtt{I}\,\mathtt{I}) \to (\lambda x.\, x\, x)\,\mathtt{I} \to \mathtt{I}\,\mathtt{I} \to \mathtt{I}$$

Evaluation strategies

The number of steps to normal form depends on the evaluation strategy.

Example

$\mathtt{I} \stackrel{\text{def}}{=} \lambda x.\, x$

Call-by-name takes 4 steps to evaluate $(\lambda x.\, x\, x)\, (\mathtt{I}\, \mathtt{I})$

$$(\lambda x.\, x\, x)\, (\mathtt{I}\, \mathtt{I}) \to \mathtt{I}\, \mathtt{I}\, (\mathtt{I}\, \mathtt{I}) \to \mathtt{I}\, (\mathtt{I}\, \mathtt{I}) \to \mathtt{I}\, \mathtt{I} \to \mathtt{I}$$

Call-by-value takes only 3 steps

$$(\lambda x.\, x\, x)(\mathtt{I}\, \mathtt{I}) \to (\lambda x.\, x\, x)\, \mathtt{I} \to \mathtt{I}\, \mathtt{I} \to \mathtt{I}$$

To measure the time cost of evaluating a λ-term we must fix a strategy.

Goal of this work

General long-term objective

Develop tools to reason about the complexity of evaluating λ-terms.

Goal of this work

General long-term objective

Develop tools to reason about the complexity of evaluating λ-terms.

State of the art

- Reasoning about the complexity of functional programs is hard.

Goal of this work

General long-term objective

Develop tools to reason about the complexity of evaluating λ-terms.

State of the art

- Reasoning about the complexity of functional programs is hard.
- This is specially true for open and strong reduction.
  (Used by proof assistants such as Coq, Agda, Lean, etc.).

Goal of this work

General long-term objective

Develop tools to reason about the complexity of evaluating λ-terms.

State of the art

- Reasoning about the complexity of functional programs is hard.
- This is specially true for open and strong reduction.
  (Used by proof assistants such as Coq, Agda, Lean, etc.).
- Typical implementation techniques are not reasonable.

Goal of this work

Specific objectives

Study a notion of evaluation: **useful open call-by-value**.

Accattoli & Sacerdoti Coen (2015)

Accattoli & Guerrieri (2017)

Goal of this work

Specific objectives

Study a notion of evaluation: **useful open call-by-value**.

Accattoli & Sacerdoti Coen (2015)

Accattoli & Guerrieri (2017)

CBV is the most widely used strategy in PL implementations.
Open reduction is essential to implement evaluation in proof assistants.
Useful evaluation is the **key** to show that strategies are reasonable.

Goal of this work

Specific objectives

Study a notion of evaluation: **useful open call-by-value**.

Accattoli & Sacerdoti Coen (2015)

Accattoli & Guerrieri (2017)

CBV is the most widely used strategy in PL implementations.
Open reduction is essential to implement evaluation in proof assistants.
Useful evaluation is the **key** to show that strategies are reasonable.

Main goal

Formulate a **quantitative type system** for useful call-by-value.

Outline

How to avoid size explosion?

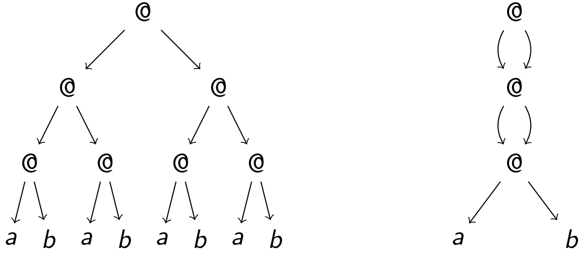
The key to obtain a reasonable cost model is to **avoid size explosion**.

How to avoid size explosion?

The key to obtain a reasonable cost model is to **avoid size explosion**.

Sharing subterms

Represent λ-terms using **directed acyclic graphs** instead of trees.



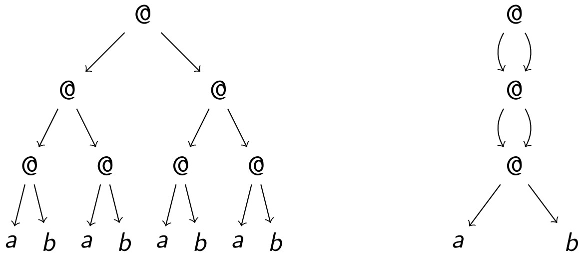
Two representations of a b (a b) (a b (a b)).

How to avoid size explosion?

The key to obtain a reasonable cost model is to **avoid size explosion**.

Sharing subterms

Represent λ-terms using **directed acyclic graphs** instead of trees.



Two representations of $\mathsf{a\,b\,(a\,b)\,(a\,b\,(a\,b))}$.

The shared representation can be written using **explicit substitutions**:

$$(x\,x)[x/y\,y][y/a\,b]$$

The Linear Substitution Calculus

Accattoli & Kesner (∼2010)

$$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid \underbrace{t[x/s]}_{\text{explicit substitution}}$$

The Linear Substitution Calculus

Accattoli & Kesner (∼2010)

$$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid \underbrace{t[x/s]}_{\text{explicit substitution}}$$

Distant beta $\quad (\lambda x.\, t)\mathtt{L}\, s \quad \to \quad t[x/s]\mathtt{L}$

The Linear Substitution Calculus

Accattoli & Kesner (∼2010)

$$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid \underbrace{t[x/s]}_{\text{explicit substitution}}$$

Distant beta $\qquad (\lambda x.\, t)\mathtt{L}\, s \quad \rightarrow \quad t[x/s]\mathtt{L}$

$\mathtt{L}$ stands for an arbitrary list of explicit substitutions.

The Linear Substitution Calculus

Accattoli & Kesner (∼2010)

$$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid \underbrace{t[x/s]}_{\text{explicit substitution}}$$

Distant beta	$(\lambda x.\, t)\mathtt{L}\, s$	$\to$	$t[x/s]\mathtt{L}$
Linear substitution	$(...x...)[x/t]$	$\to$	$(...t...)[x/t]$

The Linear Substitution Calculus

Accattoli & Kesner (∼2010)

$$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid \underbrace{t[x/s]}_{\text{explicit substitution}}$$

Distant beta	$(\lambda x.\, t)\mathtt{L}\, s$	$\to$	$t[x/s]\mathtt{L}$
Linear substitution	$(...x...)[x/t]$	$\to$	$(...t...)[x/t]$

Variables are substituted one occurrence at a time.

The Linear Substitution Calculus

Accattoli & Kesner (∼2010)

$$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid \underbrace{t[x/s]}_{\text{explicit substitution}}$$

Distant beta $\quad (\lambda x.\, t)\mathtt{L}\, s \quad \to \quad t[x/s]\mathtt{L}$

Linear substitution $\quad (\ldots x \ldots)[x/t] \quad \to \quad (\ldots t \ldots)[x/t]$

Example

$$\begin{array}{rcl}
\underline{(\lambda x.\, x\, x)\, \mathtt{I}} & \to & (\underline{x}\, x)[x/\mathtt{I}] \\
& \to & (\underline{\mathtt{I}\, x})[x/\mathtt{I}] \\
& \to & y[y/\underline{x}][x/\mathtt{I}] \\
& \to & \underline{y}[y/\mathtt{I}][x/\mathtt{I}] \\
& \to & \mathtt{I}[y/\mathtt{I}][x/\mathtt{I}]
\end{array}$$

Useful reduction

To avoid size explosion, we must perform substitution **carefully**.

Useful reduction

To avoid size explosion, we must perform substitution **carefully**.

A substitution step is **useful** if it contributes to creating a beta-redex:

$$(\underline{x}\,x)[x/\mathtt{I}] \rightarrow (\underline{\mathtt{I}\,x})[x/\mathtt{I}]$$

Useful reduction

To avoid size explosion, we must perform substitution **carefully**.

A substitution step is **useful** if it contributes to creating a beta-redex:

$$(\underline{x}\,x)[x/\mathtt{I}] \to (\underline{\mathtt{I}\,x})[x/\mathtt{I}]$$

Substituting x by $\mathtt{I}$ creates a redex $\mathtt{I}\,x$.

Useful reduction

To avoid size explosion, we must perform substitution **carefully**.

A substitution step is **useful** if it contributes to creating a beta-redex:

$$(\underline{x}\, x)[x/\mathtt{I}] \rightarrow (\underline{\mathtt{I}\, x})[x/\mathtt{I}]$$

Substituting x by $\mathtt{I}$ creates a redex $\mathtt{I}\, x$.

$$(x\, \underline{x})[x/\mathtt{I}] \rightarrow (x\, \mathtt{I})[x/\mathtt{I}]$$ (Not useful)

Useful reduction

To avoid size explosion, we must perform substitution **carefully**.

A substitution step is **useful** if it contributes to creating a beta-redex:

$$(\underline{x}\, x)[x/\mathtt{I}] \to (\underline{\mathtt{I}\, x})[x/\mathtt{I}]$$

Substituting x by $\mathtt{I}$ creates a redex $\mathtt{I}\, x$.

$$(x\, \underline{x})[x/\mathtt{I}] \to (x\, \mathtt{I})[x/\mathtt{I}] \qquad \text{(Not useful)}$$

A substitution step may *indirectly* contribute to creating a beta-redex:

$$(\underline{x}\, x)[x/y][y/\mathtt{I}] \to (\underline{y}\, x)[x/y][y/\mathtt{I}] \to (\underline{\mathtt{I}\, x})[x/y][y/\mathtt{I}]$$

Useful reduction

To avoid size explosion, we must perform substitution **carefully**.

A substitution step is **useful** if it contributes to creating a beta-redex:

$$(\underline{x}\, x)[x/\mathtt{I}] \to (\underline{\mathtt{I}\, x})[x/\mathtt{I}]$$

Substituting x by $\mathtt{I}$ creates a redex $\mathtt{I}\, x$.

$$(x\, \underline{x})[x/\mathtt{I}] \to (x\, \mathtt{I})[x/\mathtt{I}] \qquad \text{(Not useful)}$$

A substitution step may *indirectly* contribute to creating a beta-redex:

$$(\underline{x}\, x)[x/y][y/\mathtt{I}] \to (\underline{y}\, x)[x/y][y/\mathtt{I}] \to (\underline{\mathtt{I}\, x})[x/y][y/\mathtt{I}]$$

Performing only useful substitution steps indeed avoids size explosion.

Useful reduction

To avoid size explosion, we must perform substitution **carefully**.

A substitution step is **useful** if it contributes to creating a beta-redex:

$$(\underline{x}\,x)[x/\mathtt{I}] \to (\underline{\mathtt{I}\,x})[x/\mathtt{I}]$$

Substituting x by $\mathtt{I}$ creates a redex $\mathtt{I}\,x$.

$$(x\,\underline{x})[x/\mathtt{I}] \to (x\,\mathtt{I})[x/\mathtt{I}] \qquad \text{(Not useful)}$$

A substitution step may *indirectly* contribute to creating a beta-redex:

$$(\underline{x}\,x)[x/y][y/\mathtt{I}] \to (\underline{y}\,x)[x/y][y/\mathtt{I}] \to (\underline{\mathtt{I}\,x})[x/y][y/\mathtt{I}]$$

Performing only useful substitution steps indeed avoids size explosion.

Theorem

Accattoli-Dal Lago

The number of leftmost-outermost β-reduction steps to normal form is a reasonable time cost model.

Outline

Introduction

Useful Reduction

Towards Useful Open Call-by-Value

Useful Open Call-by-Value

A Quantitative Type System for Useful CBV

Closed CBV

Plotkin's call-by-value

Terms

$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s$

Values

$v ::= \lambda x.\, t$

$$(\lambda x.\, t)\, v \to_{\beta_v} t\{x := v\}$$

Closed CBV

Plotkin's call-by-value

Terms

$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s$

Values

$v ::= \lambda x.\, t$

$$(\lambda x.\, t)\, v \to_{\beta_v} t\{x := v\}$$

How should CBV be extended for *open* terms?

Open CBV — first variant

"Naive" open call-by-value

Extend the set of values to allow variables:

$$v ::= \lambda x.\, t \;\mid\; x$$

Open CBV — first variant

"Naive" open call-by-value

Extend the set of values to allow variables:

$$v ::= \lambda x.\, t \;\mid\; x$$

Well-known problem: adequacy fails

Let $\delta = \lambda x.\, x\, x$.

Open CBV — first variant

"Naive" open call-by-value

Extend the set of values to allow variables:

$$v ::= \lambda x.\, t \;\mid\; x$$

Well-known problem: adequacy fails

Let $\delta = \lambda x.\, x\, x$.

The term $(\lambda x.\, \delta)\,(z\, z)\,\delta$ is **stuck**.

(Because $z\, z$ is not a value.)

Open CBV — first variant

"Naive" open call-by-value

Extend the set of values to allow variables:

$$v ::= \lambda x.\, t \;\mid\; x$$

Well-known problem: adequacy fails

Let $\delta = \lambda x.\, x\, x$.

The term $(\lambda x.\, \delta)\,(z\, z)\,\delta$ is **stuck**.

(Because $z\, z$ is not a value.)

But it is **unsolvable**.

Open CBV — second variant

The fireball calculus

Grégoire & Leroy, Della Rocca & Paolini, Accattoli & Sacerdoti Coen

Fireballs	Inert terms
$f ::= \lambda x.\, t \mid i$	$i ::= x\, f_1 \ldots f_n \quad (n \geq 0)$

$$(\lambda x.\, t)\, f \rightarrow_{\beta_f} t\{x := f\}$$

Open CBV — second variant

The fireball calculus

Grégoire & Leroy, Della Rocca & Paolini, Accattoli & Sacerdoti Coen

Fireballs	Inert terms
$f ::= \lambda x.\, t \;\mid\; i$	$i ::= x\, f_1 \ldots f_n \quad (n \geq 0)$

$$(\lambda x.\, t)\, f \rightarrow_{\beta_f} t\{x := f\}$$

Recovers adequacy ✓

Open CBV — second variant

The fireball calculus

Grégoire & Leroy, Della Rocca & Paolini, Accattoli & Sacerdoti Coen

Fireballs
$f ::= \lambda x.\, t \mid i$

Inert terms
$i ::= x\, f_1 \ldots f_n \quad (n \geq 0)$

$$(\lambda x.\, t)\, f \to_{\beta_f} t\{x := f\}$$

Recovers adequacy ✓

Problem: still exhibits **size explosion**

It cannot be used as a cost model.

Taking the same example as before:

$$t_0 \stackrel{\text{def}}{=} z \qquad t_{n+1} \stackrel{\text{def}}{=} \Delta\, t_n \qquad \text{where } \Delta \stackrel{\text{def}}{=} \lambda x.\, x\, x$$

Open CBV — third variant

To avoid size explosion, use explicit substitutions:

Open CBV — third variant

To avoid size explosion, use explicit substitutions:

The value-substitution calculus

Accattoli & Paolini (2012)

Terms	Values
$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid t[x/s]$	$v ::= \lambda x.\, t \mid x$

Distant beta	$(\lambda x.\, t)\mathtt{L}\, s$	$\to$	$t[x/s]\mathtt{L}$
Value substitution	$t[x/v\mathtt{L}]$	$\to$	$t\{x := v\}\mathtt{L}$

Open CBV — third variant

To avoid size explosion, use explicit substitutions:

The value-substitution calculus

Accattoli & Paolini (2012)

Terms	Values
$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid t[x/s]$	$v ::= \lambda x.\, t \mid x$

Distant beta	$(\lambda x.\, t)\mathtt{L}\, s$	$\to$	$t[x/s]\mathtt{L}$
Value substitution	$t[x/v\mathtt{L}]$	$\to$	$t\{x := v\}\mathtt{L}$

Recovers adequacy ✓

The problematic term $(\lambda x.\, \delta)\,(z\, z)\,\delta$ is not stuck anymore.

$$(\lambda x.\, \delta)\,(z\, z)\,\delta \to \delta[x/z\, z]\,\delta \to (y\, y)[y/\delta][x/z\, z] \to \ldots$$

(Recall: $\delta := \lambda x.\, x\, x$)

Open CBV — third variant

To avoid size explosion, use explicit substitutions:

The value-substitution calculus

Accattoli & Paolini (2012)

Terms

$t, s, \ldots ::= x \mid \lambda x.\, t \mid t\, s \mid t[x/s]$

Values

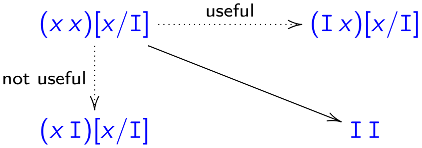
$v ::= \lambda x.\, t \mid x$

Distant beta $\quad (\lambda x.\, t)\mathtt{L}\, s \quad \to \quad t[x/s]\mathtt{L}$

Value substitution $\quad t[x/v\mathtt{L}] \quad \to \quad t\{x := v\}\mathtt{L}$

Problem: substitution is not linear

All occurrences of x are substituted at once.



Linear and Useful CBV

Starting point

The **value-substitution calculus with linear substitution**.

Linear and Useful CBV

Starting point

The **value-substitution calculus with linear substitution**.

In this work

We study two notions of reduction:

$t \overset{\circ}{\to} s$	Linear CBV	substitution steps are unrestricted
$t \overset{\bullet}{\to} s$	Useful CBV	substitution steps must be useful

Linear and Useful CBV

Starting point

The **value-substitution calculus with linear substitution**.

In this work

We study two notions of reduction:

$t \xrightarrow{\circ} s$	Linear CBV	substitution steps are unrestricted
$t \xrightarrow{\bullet} s$	Useful CBV	substitution steps must be useful

Linear and Useful CBV

Starting point

The **value-substitution calculus with linear substitution**.

In this work

We study two notions of reduction:

$t \xrightarrow{\circ} s$	Linear CBV	substitution steps are unrestricted
$t \xrightarrow{\bullet} s$	Useful CBV	substitution steps must be useful

They are strategies for **open** (but not **strong**) CBV evaluation.

These notions have been studied before.

Accattoli, Sacerdoti Coen, Guerrieri, ...

Linear and Useful CBV

Starting point

The **value-substitution calculus with linear substitution**.

In this work

We study two notions of reduction:

$t \xrightarrow{\circ} s$	Linear CBV	substitution steps are unrestricted
$t \xrightarrow{\bullet} s$	Useful CBV	substitution steps must be useful

They are strategies for **open** (but not **strong**) CBV evaluation.

These notions have been studied before.

Accattoli, Sacerdoti Coen, Guerrieri, ...

As part of our work, we reformulate them as **inductive predicates**.

Outline

Linear CBV

Linear CBV – formal definition

Reduction rules:

$$\frac{}{(\lambda x.\, t)\mathtt{L}\, s \overset{\circ}{\rightarrow}_{\mathsf{db}} t[x/s]\mathtt{L}}\ \mathtt{db}^{\circ} \qquad \frac{}{x \overset{\circ}{\rightarrow}_{\mathsf{sub}_{(x,v)}} v}\ \mathtt{sub}^{\circ} \qquad \frac{t \overset{\circ}{\rightarrow}_{\mathsf{sub}_{(x,v)}} t'}{t[x/v\mathtt{L}] \overset{\circ}{\rightarrow}_{\mathsf{lsv}} t'[x/v]\mathtt{L}}\ \mathtt{lsv}^{\circ}$$

Linear CBV

Linear CBV – formal definition

Reduction rules:

$$\frac{}{(\lambda x.\, t)\mathtt{L}\, s \overset{\circ}{\to}_{\mathsf{db}} t[x/s]\mathtt{L}}\ \mathtt{db}^{\circ} \qquad \frac{}{x \overset{\circ}{\to}_{\mathsf{sub}_{(x,v)}} v}\ \mathtt{sub}^{\circ} \qquad \frac{t \overset{\circ}{\to}_{\mathsf{sub}_{(x,v)}} t'}{t[x/v\mathtt{L}] \overset{\circ}{\to}_{\mathsf{lsv}} t'[x/v]\mathtt{L}}\ \mathtt{lsv}^{\circ}$$

Congruence rules:

$$\frac{t \overset{\circ}{\to}_{\rho} t'}{t\, s \overset{\circ}{\to}_{\rho} t'\, s}\ \mathtt{appL}^{\circ} \qquad \frac{s \overset{\circ}{\to}_{\rho} s'}{t\, s \overset{\circ}{\to}_{\rho} t\, s'}\ \mathtt{appR}^{\circ}$$

$$\frac{t \overset{\circ}{\to}_{\rho} t'}{t[x/s] \overset{\circ}{\to}_{\rho} t'[x/s]}\ \mathtt{esL}^{\circ} \qquad \frac{s \overset{\circ}{\to}_{\rho} s'}{t[x/s] \overset{\circ}{\to}_{\rho} t[x/s']}\ \mathtt{esR}^{\circ}$$

Linear CBV

Linear CBV – formal definition

Reduction rules:

$$\frac{}{(\lambda x.\, t)\mathtt{L}\, s \xrightarrow{\circ}_{\mathsf{db}} t[x/s]\mathtt{L}}\ \mathtt{db}^\circ \qquad \frac{}{x \xrightarrow{\circ}_{\mathsf{sub}_{(x,v)}} v}\ \mathtt{sub}^\circ \qquad \frac{t \xrightarrow{\circ}_{\mathsf{sub}_{(x,v)}} t'}{t[x/v\mathtt{L}] \xrightarrow{\circ}_{\mathsf{lsv}} t'[x/v]\mathtt{L}}\ \mathtt{lsv}^\circ$$

Congruence rules:

$$\frac{t \xrightarrow{\circ}_{\rho} t'}{t\, s \xrightarrow{\circ}_{\rho} t'\, s}\ \mathtt{appL}^\circ \qquad \frac{s \xrightarrow{\circ}_{\rho} s'}{t\, s \xrightarrow{\circ}_{\rho} t\, s'}\ \mathtt{appR}^\circ$$

$$\frac{t \xrightarrow{\circ}_{\rho} t'}{t[x/s] \xrightarrow{\circ}_{\rho} t'[x/s]}\ \mathtt{esL}^\circ \qquad \frac{s \xrightarrow{\circ}_{\rho} s'}{t[x/s] \xrightarrow{\circ}_{\rho} t[x/s']}\ \mathtt{esR}^\circ$$

(No congruence rule for abstraction)

Useful CBV

To determine whether a substitution step is **useful**, we need to know:

Useful CBV

To determine whether a substitution step is **useful**, we need to know:

1. Whether a variable is in an applied position or not.

$$(\underline{x}\, z)[x/\mathtt{I}] \xrightarrow{\text{useful}} (\mathtt{I}\, z)[x/\mathtt{I}] \qquad \underline{x}[x/\mathtt{I}] \xrightarrow{\text{not useful}} \mathtt{I}[x/\mathtt{I}]$$

Useful CBV

To determine whether a substitution step is **useful**, we need to know:

1. Whether a variable is in an applied position or not.

$$(\underline{x}\, z)[x/\mathtt{I}] \xrightarrow{\text{useful}} (\mathtt{I}\, z)[x/\mathtt{I}] \qquad \underline{x}[x/\mathtt{I}] \xrightarrow{\text{not useful}} \mathtt{I}[x/\mathtt{I}]$$

$$x \rightarrow_{\mathsf{sub}_{(x,\mathtt{I})}} \mathtt{I} \qquad \text{is useful iff } x \text{ is applied}$$

Useful CBV

To determine whether a substitution step is **useful**, we need to know:

1. Whether a variable is in an applied position or not.

$$(\underline{x}\, z)[x/\mathtt{I}] \xrightarrow{\text{useful}} (\mathtt{I}\, z)[x/\mathtt{I}] \qquad \underline{x}[x/\mathtt{I}] \xrightarrow{\text{not useful}} \mathtt{I}[x/\mathtt{I}]$$

$$x \rightarrow_{\mathsf{sub}_{(x,\mathtt{I})}} \mathtt{I} \quad \text{is useful iff } x \text{ is applied}$$

2. Whether a variable is (indirectly) bound to an abstraction or not.

$$(\underline{x}\, z)[x/y][y/\mathtt{I}] \xrightarrow{\text{useful}} (y\, z)[x/y][y/\mathtt{I}] \qquad (\underline{x}\, z)[x/y] \xrightarrow{\text{not useful}} (y\, z)[x/y]$$

Useful CBV

To determine whether a substitution step is **useful**, we need to know:

1. Whether a variable is in an applied position or not.

$$(\underline{x}\, z)[x/\mathtt{I}] \xrightarrow{\text{useful}} (\mathtt{I}\, z)[x/\mathtt{I}] \qquad \underline{x}[x/\mathtt{I}] \xrightarrow{\text{not useful}} \mathtt{I}[x/\mathtt{I}]$$

$x \rightarrow_{\mathsf{sub}_{(x,\mathtt{I})}} \mathtt{I}$ is useful iff x is applied

2. Whether a variable is (indirectly) bound to an abstraction or not.

$$(\underline{x}\, z)[x/y][y/\mathtt{I}] \xrightarrow{\text{useful}} (y\, z)[x/y][y/\mathtt{I}] \qquad (\underline{x}\, z)[x/y] \xrightarrow{\text{not useful}} (y\, z)[x/y]$$

$x \rightarrow_{\mathsf{sub}_{(x,y)}} y$ is useful iff x is applied and
y is indirectly bound to an abstraction

Useful CBV

The Useful CBV reduction relation is indexed by parameters:

1. A set $\mathcal{A}$ of variables assumed to be *hereditary abstractions*.
2. A set $\mathcal{S}$ of variables assumed to be *structures*.
3. A *positional flag* $\mu \in \{@, \not@\}$.

Useful CBV

The Useful CBV reduction relation is indexed by parameters:

1. A set $\mathcal{A}$ of variables assumed to be *hereditary abstractions*.
2. A set $\mathcal{S}$ of variables assumed to be *structures*.
3. A *positional flag* $\mu \in \{@, \not@\}$.

Hereditary abstractions

Abstractions or variables bound to hereditary abstractions (with ESs).

$$
\begin{array}{lcl}
(\lambda x. y)[y/z] & \in & \mathsf{HAbs}_{\mathcal{A}} \\
x[x/y][y/\mathtt{I}] & \in & \mathsf{HAbs}_{\mathcal{A}} \\
x[x/y] & \in & \mathsf{HAbs}_{\mathcal{A}} \quad \iff \quad y \in \mathcal{A}
\end{array}
$$

Useful CBV

The Useful CBV reduction relation is indexed by parameters:

1. A set $\mathcal{A}$ of variables assumed to be *hereditary abstractions*.
2. A set $\mathcal{S}$ of variables assumed to be *structures*.
3. A *positional flag* $\mu \in \{@, \not@\}$.

Hereditary abstractions

Abstractions or variables bound to hereditary abstractions (with ESs).

$$
\begin{array}{lcl}
(\lambda x.\, y)[y/z] & \in & \mathsf{HAbs}_{\mathcal{A}} \\
x[x/y][y/\mathtt{I}] & \in & \mathsf{HAbs}_{\mathcal{A}} \\
x[x/y] & \in & \mathsf{HAbs}_{\mathcal{A}} \quad \iff \quad y \in \mathcal{A}
\end{array}
$$

Structures

"Rigid" terms that are headed by a free variable in $\mathcal{S}$.

$$
\begin{array}{lclcl}
(x\, y)[y/z] & \in & \mathsf{St}_{\mathcal{S}} & \iff & x \in \mathcal{S} \\
x[x/y\, z]\, z & \in & \mathsf{St}_{\mathcal{S}} & \iff & y \in \mathcal{S} \\
(x\, y)[x/\mathtt{I}] & \notin & \mathsf{St}_{\mathcal{S}} & &
\end{array}
$$

Useful CBV

Useful CBV – formal definition

Reduction rules:

$$\frac{}{(\lambda x.\, t)\mathtt{L}\, s \overset{\bullet}{\rightarrow}_{\mathsf{db},\mathcal{A},\mathcal{S},\mu} t[x/s]\mathtt{L}}\ \mathtt{db}^{\bullet}$$

$$\frac{t \overset{\bullet}{\rightarrow}_{\mathsf{sub}_{(x,v)},\mathcal{A}\cup\{x\},\mathcal{S},\mu} t' \quad v\mathtt{L} \in \mathsf{HAbs}_{\mathcal{A}}}{t[x/v\mathtt{L}] \overset{\bullet}{\rightarrow}_{\mathsf{lsv},\mathcal{A},\mathcal{S},\mu} t'[x/v]\mathtt{L}}\ \mathtt{lsv}^{\bullet}$$

$$\frac{}{x \overset{\bullet}{\rightarrow}_{\mathsf{sub}_{(x,v)},\mathcal{A}\cup\{x\},\mathcal{S},\mathtt{@}} v}\ \mathtt{sub}^{\bullet}$$

Useful CBV – formal definition

Reduction rules:

$$\frac{}{(\lambda x.\, t)\mathtt{L}\, s \xrightarrow{\bullet}_{\mathsf{db},\mathcal{A},\mathcal{S},\mu} t[x/s]\mathtt{L}}\ \mathtt{db}^\bullet$$

$$\frac{t \xrightarrow{\bullet}_{\mathsf{sub}_{(x,v)},\mathcal{A}\cup\{x\},\mathcal{S},\mu} t' \quad v\mathtt{L} \in \mathsf{HAbs}_{\mathcal{A}}}{t[x/v\mathtt{L}] \xrightarrow{\bullet}_{\mathsf{lsv},\mathcal{A},\mathcal{S},\mu} t'[x/v]\mathtt{L}}\ \mathtt{lsv}^\bullet$$

$$\frac{}{x \xrightarrow{\bullet}_{\mathsf{sub}_{(x,v)},\mathcal{A}\cup\{x\},\mathcal{S},@} v}\ \mathtt{sub}^\bullet$$

The substituted variable must be an hereditary abstraction and in an applied position.

Useful CBV – formal definition

Congruence rules:

$$\frac{t \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S},@} t'}{t\,s \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S},\mu} t'\,s}\ \mathtt{appL}^\bullet \qquad \frac{t \in \mathsf{St}_\mathcal{S} \quad s \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S},\not@} s'}{t\,s \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S},\mu} t\,s'}\ \mathtt{appR}^\bullet$$

$$\frac{s \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S},\not@} s'}{t[x/s] \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S},\mu} t[x/s']}\ \mathtt{esR}^\bullet$$

$$\frac{t \overset{\bullet}{\to}_{\rho,\mathcal{A}\cup\{x\},\mathcal{S},\mu} t' \quad s \in \mathsf{HAbs}_\mathcal{A}}{t[x/s] \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S},\mu} t'[x/s]}\ \mathtt{esLAbs}^\bullet$$

$$\frac{t \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S}\cup\{x\},\mu} t' \quad s \in \mathsf{St}_\mathcal{S}}{t[x/s] \overset{\bullet}{\to}_{\rho,\mathcal{A},\mathcal{S},\mu} t'[x/s]}\ \mathtt{esLStruct}^\bullet$$

Useful CBV

Useful CBV – formal definition

Congruence rules:

$$\frac{t \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},@} t'}{t\,s \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t'\,s}\ \mathtt{appL}^\bullet \qquad \frac{t \in \mathsf{St}_{\mathcal{S}} \quad s \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\not@} s'}{t\,s \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t\,s'}\ \mathtt{appR}^\bullet$$

$$\frac{s \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\not@} s'}{t[x/s] \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t[x/s']}\ \mathtt{esR}^\bullet$$

$$\frac{t \xrightarrow{\bullet}_{\rho,\mathcal{A}\cup\{x\},\mathcal{S},\mu} t' \quad s \in \mathsf{HAbs}_{\mathcal{A}}}{t[x/s] \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t'[x/s]}\ \mathtt{esLAbs}^\bullet$$

$$\frac{t \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S}\cup\{x\},\mu} t' \quad s \in \mathsf{St}_{\mathcal{S}}}{t[x/s] \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t'[x/s]}\ \mathtt{esLStruct}^\bullet$$

In $t[x/s]$, evaluate s until an hereditary abstraction or a structure.
Keep track of whether variables are hereditary abstractions or structures.

Useful CBV

Useful CBV – formal definition

Congruence rules:

$$\frac{t \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},@} t'}{t\,s \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t'\,s}\ \mathtt{appL}^{\bullet} \qquad \frac{t \in \mathsf{St}_{\mathcal{S}} \quad s \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\not@} s'}{t\,s \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t\,s'}\ \mathtt{appR}^{\bullet}$$

$$\frac{s \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\not@} s'}{t[x/s] \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t[x/s']}\ \mathtt{esR}^{\bullet}$$

$$\frac{t \xrightarrow{\bullet}_{\rho,\mathcal{A}\cup\{x\},\mathcal{S},\mu} t' \quad s \in \mathsf{HAbs}_{\mathcal{A}}}{t[x/s] \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t'[x/s]}\ \mathtt{esLAbs}^{\bullet}$$

$$\frac{t \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S}\cup\{x\},\mu} t' \quad s \in \mathsf{St}_{\mathcal{S}}}{t[x/s] \xrightarrow{\bullet}_{\rho,\mathcal{A},\mathcal{S},\mu} t'[x/s]}\ \mathtt{esLStruct}^{\bullet}$$

In $t[x/s]$, evaluate s until an hereditary abstraction or a structure.
Keep track of whether variables are hereditary abstractions or structures.
This leads to complex invariants in the proofs.

Rewriting properties of Useful CBV

Theorem

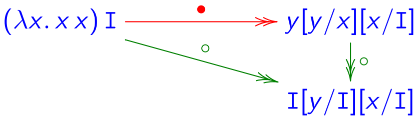
Useful CBV computes the same normal forms as Linear CBV, up to performing some (useless) substitution steps.

Rewriting properties of Useful CBV

Theorem

Useful CBV computes the same normal forms as Linear CBV,
up to performing some (useless) substitution steps.

Example

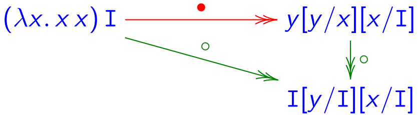


Rewriting properties of Useful CBV

Theorem

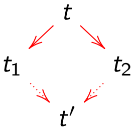
Useful CBV computes the same normal forms as Linear CBV, up to performing some (useless) substitution steps.

Example



Theorem

Useful CBV is not deterministic but enjoys the **diamond property**:



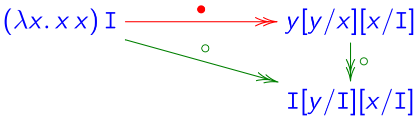
$(t_1 \neq t_2)$

Rewriting properties of Useful CBV

Theorem

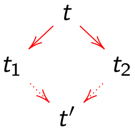
Useful CBV computes the same normal forms as Linear CBV,
up to performing some (useless) substitution steps.

Example



Theorem

Useful CBV is not deterministic but enjoys the **diamond property**:



$(t_1 \neq t_2)$

Note: all reductions to normal form have the same length.

Outline

Intersection types

$$\frac{\Gamma \vdash t : \tau \quad \Gamma \vdash t : \sigma}{\Gamma \vdash t : \tau \cap \sigma}$$

Coppo & Dezani-Ciancaglini (1978)

Intersection types

$$\frac{\Gamma \vdash t : \tau \quad \Gamma \vdash t : \sigma}{\Gamma \vdash t : \tau \cap \sigma}$$

Coppo & Dezani-Ciancaglini (1978)

- These systems enjoy both subject reduction **and** subject expansion. They can be interpreted as semantic models.

Intersection types

$$\frac{\Gamma \vdash t : \tau \quad \Gamma \vdash t : \sigma}{\Gamma \vdash t : \tau \cap \sigma}$$

Coppo & Dezani-Ciancaglini (1978)

- These systems enjoy both subject reduction **and** subject expansion.
  They can be interpreted as semantic models.
- These systems characterize notions of normalization.
  A term is typable *if and only if* it is normalizing.

Intersection types

$$\frac{\Gamma \vdash t : \tau \quad \Gamma \vdash t : \sigma}{\Gamma \vdash t : \tau \cap \sigma}$$

Coppo & Dezani-Ciancaglini (1978)

- These systems enjoy both subject reduction **and** subject expansion.
  They can be interpreted as semantic models.
- These systems characterize notions of normalization.
  A term is typable *if and only if* it is normalizing.
  Typability is undecidable.

Non-idempotent intersection types

The intersection type constructor becomes **non-idempotent**:

$$\tau \cap \tau \not\asymp \tau$$

Gardner (1994), De Carvalho (2007)

Non-idempotent intersection types

The intersection type constructor becomes **non-idempotent**:

$$\tau \cap \tau \not\asymp \tau$$

Gardner (1994), De Carvalho (2007)

- **Notation:** $A_1 \cap \ldots \cap A_n$ can be written as a multiset $[A_1, \ldots, A_n]$.

Non-idempotent intersection types

The intersection type constructor becomes **non-idempotent**:

$$\tau \cap \tau \not\asymp \tau$$

Gardner (1994), De Carvalho (2007)

- **Notation:** $A_1 \cap \ldots \cap A_n$ can be written as a multiset $[A_1, \ldots, A_n]$.
- Each expression is given as many types as times it is "used".

$$f : [(\mathtt{Int} \to \mathtt{Int}), (\mathtt{Int} \to \mathtt{Int})] \vdash f\,(f\,1) : \mathtt{Int}$$

Non-idempotent intersection types

The intersection type constructor becomes **non-idempotent**:

$$\tau \cap \tau \not\asymp \tau$$

Gardner (1994), De Carvalho (2007)

- **Notation:** $A_1 \cap \ldots \cap A_n$ can be written as a multiset $[A_1, \ldots, A_n]$.
- Each expression is given as many types as times it is "used".

$$f : [(\mathtt{Int} \to \mathtt{Int}), (\mathtt{Int} \to \mathtt{Int})] \vdash f\,(f\,1) : \mathtt{Int}$$

- These systems still characterize notions of normalization.

  (Weak, strong, CBN, CBV, CBNeed, classical calculi, effects, etc.)

Non-idempotent intersection types

The intersection type constructor becomes **non-idempotent**:

$$\tau \cap \tau \not\asymp \tau$$

Gardner (1994), De Carvalho (2007)

- **Notation:** $A_1 \cap \ldots \cap A_n$ can be written as a multiset $[A_1, \ldots, A_n]$.
- Each expression is given as many types as times it is "used".

$$f : [(\mathtt{Int} \to \mathtt{Int}), (\mathtt{Int} \to \mathtt{Int})] \vdash f\,(f\,1) : \mathtt{Int}$$

- These systems still characterize notions of normalization.
  (Weak, strong, CBN, CBV, CBNeed, classical calculi, effects, etc.)
- Typing derivations provide **upper bounds** for reduction lengths.

Non-idempotent intersection types

The intersection type constructor becomes **non-idempotent**:

$$\tau \cap \tau \not\asymp \tau$$

Gardner (1994), De Carvalho (2007)

- **Notation:** $A_1 \cap \ldots \cap A_n$ can be written as a multiset $[A_1, \ldots, A_n]$.
- Each expression is given as many types as times it is "used".

$$f : [(\mathtt{Int} \to \mathtt{Int}), (\mathtt{Int} \to \mathtt{Int})] \vdash f\,(f\,1) : \mathtt{Int}$$

- These systems still characterize notions of normalization.
  (Weak, strong, CBN, CBV, CBNeed, classical calculi, effects, etc.)
- Typing derivations provide **upper bounds** for reduction lengths.
- They have been refined to provide **exact bounds**.
  Accattoli, Kesner & Lengrand (2018)

Quantitative type system for Useful CBV

$$\begin{array}{rlcl} \text{Arrow types} & \tau & ::= & \mathcal{M}^? \rightarrow \mathcal{M} \\ \text{Types} & \mathcal{M} & ::= & \underbrace{\mathbf{n}}_{\text{structures}} \mid \underbrace{[\tau_k]_{k \in K}}_{\text{hereditary abstractions}} \\ \text{Optional types} & \mathcal{M}^? & ::= & \texttt{none} \mid \mathcal{M} \end{array}$$

Quantitative type system for Useful CBV

Arrow types	τ	$::=$	$\mathcal{M}^{?} \rightarrow \mathcal{M}$
Types	$\mathcal{M}$	$::=$	$\underbrace{\mathbf{n}}_{\text{structures}} \mid \underbrace{[\tau_k]_{k \in K}}_{\text{hereditary abstractions}}$
Optional types	$\mathcal{M}^{?}$	$::=$	$\texttt{none} \mid \mathcal{M}$
Judgments		:	$x_1 : \mathcal{M}^{?}_1, \ldots, x_n : \mathcal{M}^{?}_n \vdash^{(m,e)} t : \mathcal{M}$

Quantitative type system for Useful CBV

$$\begin{array}{rlcl}
\text{Arrow types} & \tau & ::= & \mathcal{M}^? \to \mathcal{M} \\
\text{Types} & \mathcal{M} & ::= & \underbrace{\mathbf{n}}_{\text{structures}} \mid \underbrace{[\tau_k]_{k \in K}}_{\text{hereditary abstractions}} \\
\text{Optional types} & \mathcal{M}^? & ::= & \mathtt{none} \mid \mathcal{M} \\
\text{Judgments} & & : & x_1 : \mathcal{M}^?_1, \ldots, x_n : \mathcal{M}^?_n \vdash^{(m,e)} t : \mathcal{M}
\end{array}$$

Typing rules

$$\frac{n = \#(\mathcal{M})}{x : \mathcal{M} \vdash^{(0,n)} x : \mathcal{M}}\ \mathtt{var} \qquad \frac{(\Gamma_i; x : \mathcal{M}^?_i \vdash^{(m_i,e_i)} t : \mathcal{N}_i)_{i \in I}}{+_{i \in I}\Gamma_i \vdash^{(+_{i \in I} m_i, +_{i \in I} e_i)} \lambda x.\, t : [\mathcal{M}^?_i \to \mathcal{N}_i]_{i \in I}}\ \mathtt{abs}$$

$$\frac{\Gamma \vdash^{(m,e)} t : \mathbf{n} \qquad \Delta \vdash^{(m',e')} s : \mathtt{tt}}{\Gamma + \Delta \vdash^{(m+m',e+e')} t\, s : \mathbf{n}}\ \mathtt{appPersistent}$$

$$\frac{\Gamma \vdash^{(m,e)} t : [\mathcal{M}^? \to \mathcal{N}] \qquad \mathcal{M}^? \lhd \mathcal{M} \qquad \Delta \vdash^{(m',e')} s : \mathcal{M}}{\Gamma + \Delta \vdash^{(1+m+m',e+e')} t\, s : \mathcal{N}}\ \mathtt{appConsuming}$$

$$\frac{\Gamma; x : \mathcal{M}^? \vdash^{(m,e)} t : \mathcal{N} \qquad \mathcal{M}^? \lhd \mathcal{M} \qquad \Delta \vdash^{(m',e')} s : \mathcal{M}}{\Gamma + \Delta \vdash^{(m+m',e+e')} t[x/s] : \mathcal{N}}\ \mathtt{es}$$

Quantitative type system for Useful CBV

Arrow types	τ	$::=$	$\mathcal{M}^? \to \mathcal{M}$
Types	$\mathcal{M}$	$::=$	$\underbrace{\mathbf{n}}_{\text{structures}} \mid \underbrace{[\tau_k]_{k \in K}}_{\text{hereditary abstractions}}$
Optional types	$\mathcal{M}^?$	$::=$	none $\mid \mathcal{M}$
Judgments		$:$	$x_1 : \mathcal{M}^?_1, \ldots, x_n : \mathcal{M}^?_n \vdash^{(m,e)} t : \mathcal{M}$

Typing rules

$$\frac{n = \#(\mathcal{M})}{x : \mathcal{M} \vdash^{(0,n)} x : \mathcal{M}}\ \texttt{var} \qquad \frac{(\Gamma_i; x : \mathcal{M}^?_i \vdash^{(m_i,e_i)} t : \mathcal{N}_i)_{i \in I}}{+_{i \in I}\Gamma_i \vdash^{(+_{i \in I} m_i, +_{i \in I} e_i)} \lambda x. t : [\mathcal{M}^?_i \to \mathcal{N}_i]_{i \in I}}\ \texttt{abs}$$

$$\frac{\Gamma \vdash^{(m,e)} t : \mathbf{n} \qquad \Delta \vdash^{(m',e')} s : \mathbb{t}}{\Gamma + \Delta \vdash^{(m+m',e+e')} t\, s : \mathbf{n}}\ \texttt{appPersistent}$$

$$\frac{\Gamma \vdash^{(m,e)} t : [\mathcal{M}^? \to \mathcal{N}] \qquad \mathcal{M}^? \lhd \mathcal{M} \qquad \Delta \vdash^{(m',e')} s : \mathcal{M}}{\Gamma + \Delta \vdash^{(1+m+m',e+e')} t\, s : \mathcal{N}}\ \texttt{appConsuming}$$

$$\frac{\Gamma; x : \mathcal{M}^? \vdash^{(m,e)} t : \mathcal{N} \qquad \mathcal{M}^? \lhd \mathcal{M} \qquad \Delta \vdash^{(m',e')} s : \mathcal{M}}{\Gamma + \Delta \vdash^{(m+m',e+e')} t[x/s] : \mathcal{N}}\ \texttt{es}$$

Quantitative type system for Useful CBV

$$\begin{array}{rlll} \text{Arrow types} & \tau & ::= & \mathcal{M}^? \to \mathcal{M} \\ \text{Types} & \mathcal{M} & ::= & \underbrace{\mathbf{n}}_{\text{structures}} \mid \underbrace{[\tau_k]_{k\in K}}_{\text{hereditary abstractions}} \\ \text{Optional types} & \mathcal{M}^? & ::= & \texttt{none} \mid \mathcal{M} \\ \text{Judgments} & & : & x_1 : \mathcal{M}^?_1, \ldots, x_n : \mathcal{M}^?_n \vdash^{(m,e)} t : \mathcal{M} \end{array}$$

Typing rules

$$\frac{n = \#(\mathcal{M})}{x : \mathcal{M} \vdash^{(0,n)} x : \mathcal{M}}\ \texttt{var} \qquad \frac{(\Gamma_i; x : \mathcal{M}^?_i \vdash^{(m_i,e_i)} t : \mathcal{N}_i)_{i\in I}}{+_{i\in I}\Gamma_i \vdash^{(+_{i\in I}m_i, +_{i\in I}e_i)} \lambda x.\, t : [\mathcal{M}^?_i \to \mathcal{N}_i]_{i\in I}}\ \texttt{abs}$$

$$\frac{\Gamma \vdash^{(m,e)} t : \mathbf{n} \qquad \Delta \vdash^{(m',e')} s : \mathtt{tt}}{\Gamma + \Delta \vdash^{(m+m',e+e')} t\,s : \mathbf{n}}\ \texttt{appPersistent}$$

$$\frac{\Gamma \vdash^{(m,e)} t : [\mathcal{M}^? \to \mathcal{N}] \qquad \mathcal{M}^? \lhd \mathcal{M} \qquad \Delta \vdash^{(m',e')} s : \mathcal{M}}{\Gamma + \Delta \vdash^{(1+m+m',e+e')} t\,s : \mathcal{N}}\ \texttt{appConsuming}$$

$$\frac{\Gamma; x : \mathcal{M}^? \vdash^{(m,e)} t : \mathcal{N} \qquad \mathcal{M}^? \lhd \mathcal{M} \qquad \Delta \vdash^{(m',e')} s : \mathcal{M}}{\Gamma + \Delta \vdash^{(m+m',e+e')} t[x/s] : \mathcal{N}}\ \texttt{es}$$

Quantitative type system for Useful CBV

$$
\begin{array}{rlll}
\text{Arrow types} & \tau & ::= & \mathcal{M}^? \to \mathcal{M} \\
\text{Types} & \mathcal{M} & ::= & \underbrace{\mathbf{n}}_{\text{structures}} \mid \underbrace{[\tau_k]_{k \in K}}_{\text{hereditary abstractions}} \\
\text{Optional types} & \mathcal{M}^? & ::= & \texttt{none} \mid \mathcal{M} \\
\text{Judgments} & & : & x_1 : \mathcal{M}_1^?, \ldots, x_n : \mathcal{M}_n^? \vdash^{(m,e)} t : \mathcal{M}
\end{array}
$$

Typing rules

$$
\frac{n = \#(\mathcal{M})}{x : \mathcal{M} \vdash^{(0,n)} x : \mathcal{M}}\ \texttt{var}
\qquad
\frac{(\Gamma_i; x : \mathcal{M}_i^? \vdash^{(m_i,e_i)} t : \mathcal{N}_i)_{i \in I}}{+_{i \in I}\Gamma_i \vdash^{(+_{i \in I} m_i, +_{i \in I} e_i)} \lambda x. t : [\mathcal{M}_i^? \to \mathcal{N}_i]_{i \in I}}\ \texttt{abs}
$$

$$
\frac{\Gamma \vdash^{(m,e)} t : \mathbf{n} \qquad \Delta \vdash^{(m',e')} s : \mathbb{t}}{\Gamma + \Delta \vdash^{(m+m',e+e')} t\, s : \mathbf{n}}\ \texttt{appPersistent}
$$

$$
\frac{\Gamma \vdash^{(m,e)} t : [\mathcal{M}^? \to \mathcal{N}] \qquad \mathcal{M}^? \lhd \mathcal{M} \qquad \Delta \vdash^{(m',e')} s : \mathcal{M}}{\Gamma + \Delta \vdash^{(1+m+m',e+e')} t\, s : \mathcal{N}}\ \texttt{appConsuming}
$$

$$
\frac{\Gamma; x : \mathcal{M}^? \vdash^{(m,e)} t : \mathcal{N} \qquad \mathcal{M}^? \lhd \mathcal{M} \qquad \Delta \vdash^{(m',e')} s : \mathcal{M}}{\Gamma + \Delta \vdash^{(m+m',e+e')} t[x/s] : \mathcal{N}}\ \texttt{es}
$$

Quantitative type system for Useful CBV

Example

$$
\dfrac{
  \dfrac{
    \dfrac{}{x : [[\,] \to [\,]] \vdash^{(0,\textcolor{red}{1})} x : [[\,] \to [\,]]}
    \qquad
    \dfrac{}{x : [\,] \vdash^{(0,\textcolor{blue}{0})} x : [\,]}
  }{x : [[\,] \to [\,]] \vdash^{(\textcolor{green}{1},1)} x\,x : [\,]}
  \qquad
  \dfrac{
    \dfrac{}{y : [\,] \vdash^{(0,0)} y : [\,]}
  }{\vdash^{(0,0)} \mathtt{I} : [[\,], [\,] \to [\,]]}
}{\vdash^{(1,1)} (\textcolor{red}{x}\,\textcolor{blue}{x})[x/\mathtt{I}] : [\,]}
$$

Quantitative type system for Useful CBV

Theorem (Soundness and completeness)

The following are equivalent:

1. There is a tight derivable typing judgment $\Gamma \vdash^{(m,e)} t : \tau$.
2. t normalizes in exactly m beta steps and e substitution steps (in the Useful CBV strategy).

Quantitative type system for Useful CBV

Theorem (Soundness and completeness)

The following are equivalent:

1. There is a tight derivable typing judgment $\Gamma \vdash^{(m,e)} t : \tau$.
2. t normalizes in exactly m beta steps and e substitution steps (in the Useful CBV strategy).

The proof follows the already well-known strategies:

- Soundness ($1 \Rightarrow 2$): substitution lemma, subject reduction.
- Completeness ($2 \Rightarrow 1$): anti-substitution lemma, subject expansion.

(But it is very intricate).

Outline

Conclusion

Summary

- Compositional specifications of Linear CBV and Useful CBV.
- Useful CBV is reasonable and implements Linear CBV.
- A sound and complete quantitative type system for Useful CBV.

Conclusion

Summary

- Compositional specifications of Linear CBV and Useful CBV.
- Useful CBV is reasonable and implements Linear CBV.
- A sound and complete quantitative type system for Useful CBV.
  Surprisingly simple (relative to the complex operational semantics).

Future work

- Capture further optimizations used by abstract machines.
  Accattoli & Guerrieri (2017)
- Extend to **strong** CBV/CBNeed.