LAMBDA CALCULI THROUGH THE LENS OF LINEAR LOGIC

Lesson 5: **Observational Equivalence By Means of Intersection Types**

Delia KESNER
Email : kesner@irif.fr
URL : www.irif.fr/~kesner

Université Paris Cité and CNRS

Observational Equivalence

Call-by-Name

Call-by-Value

?

Call-by-Need

Linear Call-by-Value

Neededness

CbN/CbV
Useful Evaluation

Observational Equivalence

Call-by-Name

Call-by-Value

$$S_1 \cong S_2$$

Call-by-Need

Linear Call-by-Value

Neededness

CbN/CbV
Useful Evaluation

Call-by-Name

Call-by-Value

$$\mathbf{t} \cong_{\mathcal{S}_1} \mathbf{u} \text{ iff } \mathbf{t} \cong_{\mathcal{S}_2} \mathbf{u}$$

Call-by-Need

Linear Call-by-Value

Neededness

CbN/CbV
Useful Evaluation

Call-by-Name

Call-by-Value

$\mathbf{t} \cong_{\mathcal{S}} \mathbf{u}$: for all context C,
C⟪$\mathbf{t}$⟫ terminates iff C⟪$\mathbf{u}$⟫ terminates

Call-by-Need

Linear Call-by-Value

Neededness

CbN/CbV
Useful Evaluation

Call-by-Name Different from Call-by-Value

$$\ncong$$

Call-by-Name Different from Call-by-Value

$\ncong$

Sometimes similar,

$$(\lambda x.8)(4+3) \rightarrow_{\texttt{CbN}} 8$$
$$(\lambda x.8)(4+3) \rightarrow_{\texttt{CbV}} (\lambda x.8)7 \rightarrow_{\texttt{CbV}} 8$$

Call-by-Name Different from Call-by-Value

Sometimes similar,

$$(\lambda x.8)(4+3) \rightarrow_{\texttt{CbN}} 8$$
$$(\lambda x.8)(4+3) \rightarrow_{\texttt{CbV}} (\lambda x.8)7 \rightarrow_{\texttt{CbV}} 8$$

But different:

$$(\lambda x.8)\Omega \rightarrow_{\texttt{CbN}} 8$$
$$(\lambda x.8)\Omega \rightarrow_{\texttt{CbV}} (\lambda x.8)\Omega \rightarrow_{\texttt{CbV}} \ldots\ldots\infty$$

Call-by-Need Equivalent to Call-by-Name

$$\cong$$

Call-by-Need Equivalent to Call-by-Name

$\cong$

Example:

$$\texttt{Twice}\ (4+3) \rightarrow_{\texttt{CbN}} (4+3)+(4+3) \rightarrow_{\texttt{CbN}} 7+(4+3) \rightarrow_{\texttt{CbN}} 7+7 \rightarrow_{\texttt{CbN}} 14$$
$$\texttt{Twice}\ (4+3) \rightarrow_{\texttt{CbNeed}} \texttt{Twice}\ 7 \rightarrow_{\texttt{CbNeed}} 7+7 \rightarrow_{\texttt{CbNeed}} 14$$

where $\texttt{Twice} = \lambda x.x + x$.

Call-by-Need Different from Call-by-Value

$\ncong$

Call-by-Need Different from Call-by-Value

$\ncong$

A consequence of the two previous results.

Call-by-Need Equivalent to Neededness

Call-by-Need Equivalent to Neededness

(Syntactical) **call-by-need** is similar to **(semantical)** **neededness**

$$(\lambda x.x)(4+3) \rightarrow_{\texttt{CbNeed}} (\lambda x.x)7 \rightarrow_{\texttt{CbNeed}} 7$$
$$(\lambda x.x)(4+3) \rightarrow_{\texttt{Neededness}} 4+3 \rightarrow_{\texttt{Neededness}} 7$$

State of the Art

- **Ariola-Felleisen'97**:
  Call-by-Name is obs. equivalent to **Call-by-Need**
- **Barendregt-Kennaway-Klop-Sleep'87**, **Ariola-Felleisen'97**:
  Call-by-Need is obs. equivalent to **Nedeedness**
- **Accattoli-Ugo Dal Lago'14**:
  Call-by-Name is obs. equivalent to **Useful Call-by-Name**
- **Accattoli-Sacerdotti Coen'15**, **Accattoli-Guerrieri'17**:
  Call-by-Value is obs. equivalent to **Useful Call-by-Value** (definition postponed)

State of the Art

- **Ariola-Felleisen'97**:
  Call-by-Name is obs. equivalent to **Call-by-Need**
- **Barendregt-Kennaway-Klop-Sleep'87**, **Ariola-Felleisen'97**:
  Call-by-Need is obs. equivalent to **Nedeedness**
- **Accattoli-Ugo Dal Lago'14**:
  Call-by-Name is obs. equivalent to **Useful Call-by-Name**
- **Accattoli-Sacerdotti Coen'15**, **Accattoli-Guerrieri'17**:
  Call-by-Value is obs. equivalent to **Useful Call-by-Value** (definition postponed)

State of the Art

- **Ariola-Felleisen'97**:
  Call-by-Name is obs. equivalent to **Call-by-Need**
- **Barendregt-Kennaway-Klop-Sleep'87**, **Ariola-Felleisen'97**:
  Call-by-Need is obs. equivalent to **Nedeedness**
- **Accattoli-Ugo Dal Lago'14**:
  Call-by-Name is obs. equivalent to **Useful Call-by-Name**
- **Accattoli-Sacerdotti Coen'15**, **Accattoli-Guerrieri'17**:
  Call-by-Value is obs. equivalent to **Useful Call-by-Value** (definition postponed)

Syntactical proofs
Difficult to adapt to other variants

State of the Art

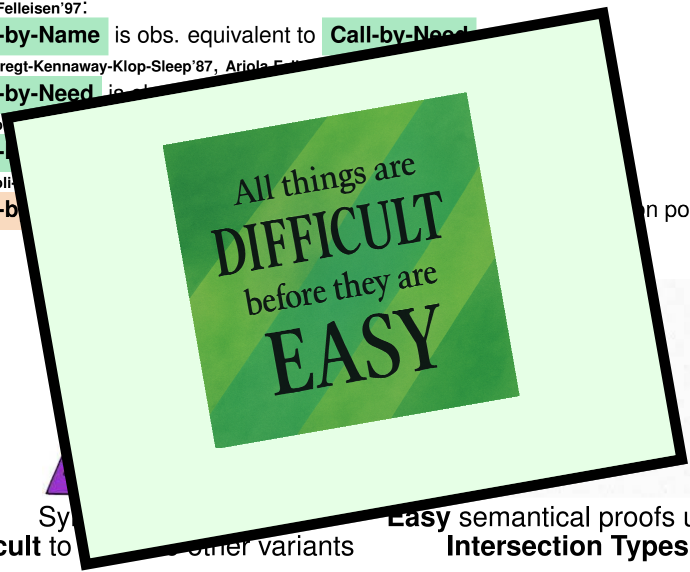
- Ariola-Felleisen'97:
  Call-by-Name is obs. equivalent to **Call-by-Need**
- Barendregt-Kennaway-Klop-Sleep'87, Ariola-Felleisen'97:
  Call-by-Need is obs. equivalent to **Nedeedness**
- Accattoli-Ugo Dal Lago'14:
  Call-by-Name is obs. equivalent to **Useful Call-by-Name**
- Accattoli-Sacerdotti Coen'15, Accattoli-Guerrieri'17:
  Call-by-Value is obs. equivalent to **Useful Call-by-Value** (definition postponed)

Syntactical proofs
Difficult to adapt to other variants

Easy semantical proofs using
Intersection Types

State of the Art

- Ariola-Felleisen'97:
  Call-by-Name is obs. equivalent to **Call-by-Need**
- Barendregt-Kennaway-Klop-Sleep'87, Ariola-F…
  Call-by-Need is ob…
- Accatto…
  Call-…
- Accattoli-…
  Call-b… …n postponed)



Sy… **Difficult** to … o other variants

Easy semantical proofs using **Intersection Types**

Call-by-Name versus Call-by-Value

		
Understood in the theory of programming		
Used in practice		

		
Understood in the theory of programming	☺	☹
Used in practice	☹	☺

Today we focus on **Call-by-Value**

$\boxed{\textbf{Correspondence}}$

Typability $\Longleftrightarrow$ **Termination**

$\boxed{\textbf{Correspondence}}$

Typability $\Longleftrightarrow$ **Termination**

- Applicable to **different** notions of termination.

$$\boxed{\textbf{Correspondence}}$$

$$\textbf{Typability} \quad \Longleftrightarrow \quad \textbf{Termination}$$

- Applicable to **different** notions of termination.
- Which give rise to **different** intersection type system.

$$\boxed{\textbf{Correspondence}}$$

$$\textbf{Typability} \underset{\textbf{Soundness}}{\Longrightarrow} \textbf{Termination}$$

- Applicable to **different** notions of termination.
- Which give rise to **different** intersection type system.
- **Soundness**: based on a **simple arithmetical** proof of **subject reduction**.

$\boxed{\textbf{Correspondence}}$

Typability $\Longleftarrow$ **Termination**

Completeness

- Applicable to **different** notions of termination.
- Which give rise to **different** intersection type system.
- **Soundness**: based on a **simple arithmetical** proof of **subject reduction**.
- **Completeness**: based on **typability of normal forms** and **subject expansion**.

Correspondence

Typability $\Longleftrightarrow$ ...ation

- Applicable...
- Whic...
- **Sound**... ...**on**.
- **Comple**... ...**ansion**.

From now on:

(Static) Typing System $\mathbb{T}$

and

(Dynamic) Calculus $\mathbb{C}$

Outline

Agenda

Observational Equivalence for Call-by-Name/Call-by-Need

Call-by-Name	Call-by-Need	Neededness
`CbN`	`CbNeed`	`Neededness`

?

Observational Equivalence for Call-by-Name/Call-by-Need

Call-by-Name	**Call-by-Need**	**Neededness**
`CbN`	`CbNeed`	`Neededness`

$$S_1 \cong S_2$$

?

Observational Equivalence for Call-by-Name/Call-by-Need

Call-by-Name	Call-by-Need	Neededness
`CbN`	`CbNeed`	`Neededness`

$$\mathbf{t} \cong_{\mathcal{S}_1} \mathbf{u} \iff \mathbf{t} \cong_{\mathcal{S}_2} \mathbf{u}$$

?

Observational Equivalence for Call-by-Name/Call-by-Need

Call-by-Name	**Call-by-Need**	**Neededness**
`CbN`	`CbNeed`	`Neededness`

$\mathbf{t} \cong_{\mathcal{S}} \mathbf{u}$: for all context C,

C⟪$\mathbf{t}$⟫ $\mathcal{S}$-terminates $\iff$ C⟪$\mathbf{u}$⟫ $\mathcal{S}$-terminates

?

Observational Equivalence for Call-by-Name/Call-by-Need

Call-by-Name
`CbN`

Call-by-Need
`CbNeed`

Neededness
`Neededness`

$\mathbf{t} \cong_{\mathcal{S}} \mathbf{u}$: for all context C,

$$\mathsf{C}\langle\!\langle \mathbf{t} \rangle\!\rangle\ \mathcal{S}\text{-terminates} \iff \mathsf{C}\langle\!\langle \mathbf{u} \rangle\!\rangle\ \mathcal{S}\text{-terminates}$$

$$\mathsf{C}\langle\!\langle \mathbf{t} \rangle\!\rangle \text{ is typable} \iff \mathsf{C}\langle\!\langle \mathbf{u} \rangle\!\rangle \text{ is typable}$$

?

Observational Equivalence for Call-by-Name/Call-by-Need

Call-by-Name	Call-by-Need	Neededness
`CbN`	`CbNeed`	`Neededness`

$\mathbf{t} \cong_{\mathcal{S}} \mathbf{u}$: for all context C,

C⟪$\mathbf{t}$⟫ $\mathcal{S}$-terminates $\iff$ C⟪$\mathbf{u}$⟫ $\mathcal{S}$-terminates

C⟪$\mathbf{t}$⟫ is typable $\iff$ C⟪$\mathbf{u}$⟫ is typable

SAME typing system (called $\mathcal{A}$)
to capture **DIFFERENT** models of `CbN` computation

Observational Equivalence by Means of Types: First Example

Theorem

- $\mathcal{A}$ -*Typability* $\Longleftrightarrow$ `CbN` -*Termination.*
- $\mathcal{A}$ -*Typability* $\Longleftrightarrow$ `CbNeed` -*Termination.*
- $\mathcal{A}$ -*Typability* $\Longleftrightarrow$ `Neededness` -*Termination.*

Observational Equivalence by Means of Types: First Example

Theorem

- $\mathcal{A}$-*Typability* $\Longleftrightarrow$ CbN -*Termination.*
- $\mathcal{A}$-*Typability* $\Longleftrightarrow$ CbNeed -*Te*
- $\mathcal{A}$-*Typability* $\Longleftrightarrow$ N

(K.'16, K.-Viso-Ríos'18)

CbN $\cong$ CbNeed $\cong$ Neededness

Observational Equivalence by Means of Types: First Example

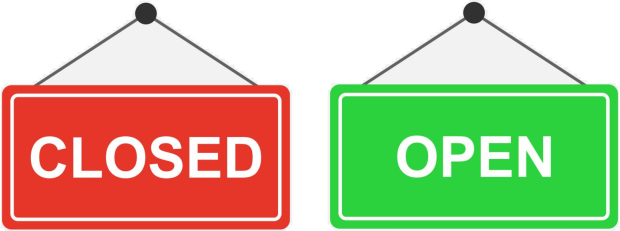
Theorem

- $\mathcal{A}$-*Typability* $\Longleftrightarrow$ `CbN` -*Termination.*
- $\mathcal{A}$-*Typability* $\Longleftrightarrow$ `CbNeed` -*Te*
- $\mathcal{A}$-*Typability* $\Longleftrightarrow$ `N`

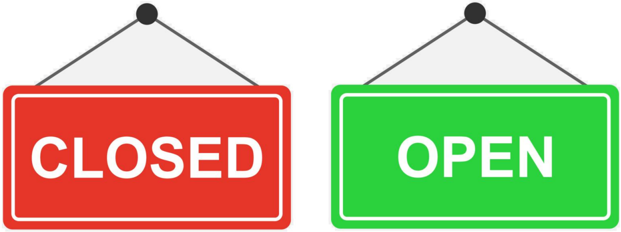
Let's apply these ideas to Call-by-Value

Agenda

Closed Versus Open Terms

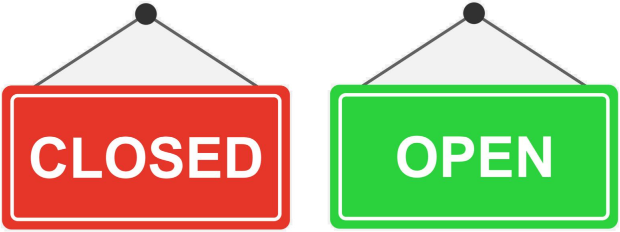


Closed Versus Open Terms



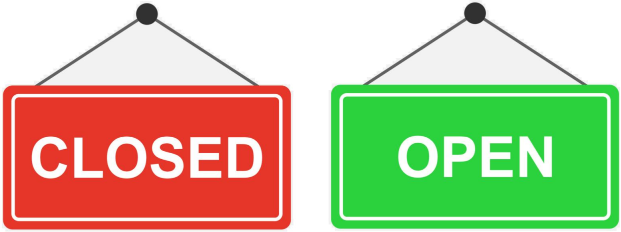
- **Closed** terms (no free variables): *e.g.* $\lambda x.x + 2$

Closed Versus Open Terms



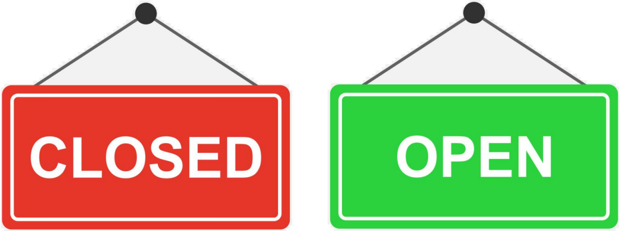
- **Closed** terms (no free variables): *e.g.* $\lambda x.x + 2$
  - Used in **programming languages** (**weak** evaluation)

Closed Versus Open Terms



- **Closed** terms (no free variables): *e.g.* $\lambda x.x + 2$
  - Used in **programming languages** (**weak** evaluation)
- **Open** terms (with free variables): *e.g.* $\lambda x.x + y$

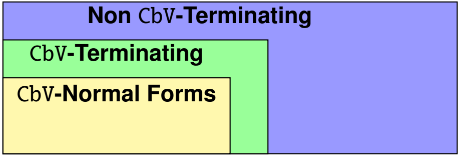
Closed Versus Open Terms



- **Closed** terms (no free variables): *e.g.* $\lambda x.x + 2$
  - Used in **programming languages** (**weak** evaluation)
- **Open** terms (with free variables): *e.g.* $\lambda x.x + y$
  - Used in **proof assistants** (**strong** evaluation)
  - Partial evaluation
  - Symbolic computation
  - Metaprogramming

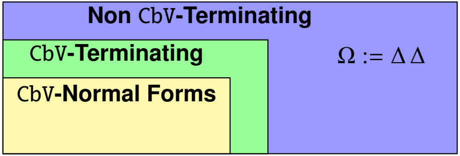
Call-by-Value on Open Terms

Drawback in Plotkin's CbV (Paolini-RonchiDellaRocca'99):



Call-by-Value on Open Terms

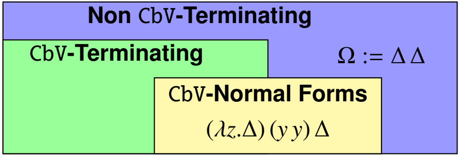
Drawback in Plotkin's `CbV` (Paolini-RonchiDellaRocca'99):



where $\Delta := \lambda x.xx$

Call-by-Value on Open Terms

Drawback in Plotkin's $\mathtt{CbV}$ **(Paolini-RonchiDellaRocca'99):**

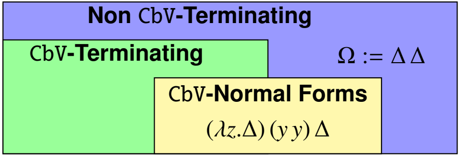


where $\Delta := \lambda x.xx$

terms $y\,t_1 \ldots t_n$ are **structures**

Call-by-Value on Open Terms

Drawback in Plotkin's $\mathtt{CbV}$ **(Paolini-RonchiDellaRocca'99):**



where $\Delta := \lambda x.xx$

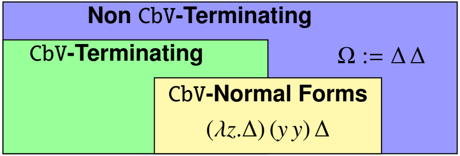
terms $y\,t_1 \ldots t_n$ are **structures**

Towards the Solution:

- **New** Syntax: **explicit substitutions**
- **New** Operational semantics: **reduction at a distance** (**Accattoli-K.'10**)

Call-by-Value on Open Terms

Drawback in Plotkin's CbV **(Paolini-RonchiDellaRocca'99):**



where $\Delta := \lambda x.xx$

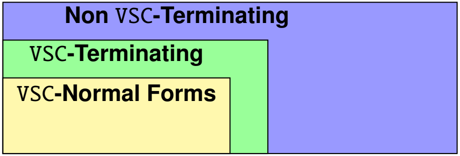
terms $y\,t_1 \ldots t_n$ are **structures**

Towards the Solution:

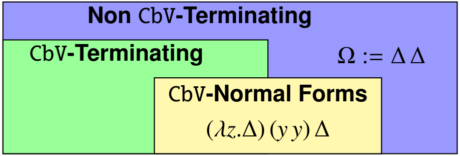
- **New** Syntax: **explicit substitutions**
- **New** Operational semantics: **reduction at a distance** (**Accattoli-K.'10**)

The Resulting New CbV **Setting:**

- The **Value Substitution Calculus** (VSC) (**Accattoli and Paolini'12**)
- The VSC extends Plotkin's CbV



Drawback in Plotkin's $\mathtt{CbV}$ **(Paolini-RonchiDellaRocca'99):**



where $\Delta := \lambda x.xx$

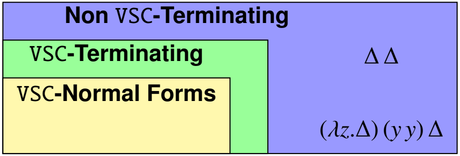
terms $y\,t_1 \ldots t_n$ are **structures**

Towards the Solution:

- **New** Syntax: **explicit substitutions**
- **New** Operational semantics: **reduction at a distance** (**Accattoli-K.'10**)

The Resulting New $\mathtt{CbV}$ **Setting:**

- The **Value Substitution Calculus** ($\mathtt{VSC}$) (**Accattoli and Paolini'12**)
- The $\mathtt{VSC}$ extends Plotkin's $\mathtt{CbV}$



From Plotkin's CbV to the Value Substitution Calculus VSC

Values $\mathtt{v} ::= \mathtt{x} \mid \lambda\mathtt{x}.\mathtt{t}$

Terms $t, u ::= \mathtt{v} \mid \mathtt{t}\,\mathtt{u}$

Values $\quad \mathtt{v} ::= \mathtt{x} \mid \lambda\mathtt{x}.\mathtt{t}$

Terms $\quad t, u ::= \mathtt{v} \mid \mathtt{t}\,\mathtt{u}$

Operational Semantics

Plotkin's CbV :

$$(\lambda x.t)\mathtt{v} \quad \rightarrow_{\beta_{\mathbf{v}}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}$$

Values $\quad v ::= x \mid \lambda x.t$

Terms $\quad t, u ::= v \mid t\,u \mid t[x\backslash u]$ **(explicit substitution)**

Operational Semantics

Plotkin's CbV :

$$(\lambda x.t)v \quad \rightarrow_{\beta_v} \quad t\{\{x\backslash v\}\}$$

From Plotkin's CbV to the Value Substitution Calculus VSC

Values $\mathtt{v} ::= \mathtt{x} \mid \lambda \mathtt{x}.\mathtt{t}$

Terms $t, u ::= \mathtt{v} \mid \mathtt{t}\,\mathtt{u} \mid \mathtt{t}[\mathtt{x}\backslash\mathtt{u}]$ **(explicit substitution)**

Operational Semantics

Beta:
(multiplicative step)

$$(\lambda x.t)u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]$$

(Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}] \quad \rightarrow_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}$$

Distant operational semantics is inspired from linear logic

=

Compact form of commuting conversions

Distant Operational Semantics

Beta:
(multiplicative step)

$$(\lambda x.t)u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]$$

(Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}] \quad \rightarrow_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}$$

Distant operational semantics is inspired from linear logic

=

Compact form of commuting conversions

$t\,\mathtt{L}$ abbreviates $t\,[x_1\backslash u_1]\ldots[x_n\backslash u_n]$ $(n \geq 0)$

Distant Operational Semantics

Beta:
(multiplicative step)

$$(\lambda x.t)u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]$$

(Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}] \quad \rightarrow_{\mathtt{sV}} \quad t\{\{x\backslash \mathtt{v}\}\}$$

Distant operational semantics is inspired from linear logic

=

Compact form of commuting conversions

$t\,\mathtt{L}$ abbreviates $t\,[x_1\backslash u_1]\ldots[x_n\backslash u_n]$ $(n \geq 0)$

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\mathtt{L}\,u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]\,\mathtt{L}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\,\mathtt{L}\,] \quad \rightarrow_{\mathtt{sV}} \quad t\{\{x\backslash \mathtt{v}\}\}\,\mathtt{L}$$

Let $\Delta := \lambda x.xx$. Then,

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\mathtt{L}\; u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]\,\mathtt{L}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\,\mathtt{L}] \quad \rightarrow_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}\,\mathtt{L}$$

Let $\Delta := \lambda x.xx$. Then,

$$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta$$

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\mathtt{L}\ u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]\,\mathtt{L}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\,\mathtt{L}\,] \quad \rightarrow_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}\,\mathtt{L}$$

Let $\Delta := \lambda x.xx$. Then,

$$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \quad \rightarrow_{\mathtt{db}} \quad \Delta[z\backslash y\,y]\,\Delta$$

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\mathtt{L}\,u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]\,\mathtt{L}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathrm{v}\,\mathtt{L}\,] \quad \rightarrow_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathrm{v}\}\!\}\,\mathtt{L}$$

Let $\Delta := \lambda x.xx$. Then,

$$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \quad \rightarrow_{\mathtt{db}} \quad \Delta\,\boxed{[z\backslash y\,y]}\,\Delta$$

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\boxed{\mathtt{L}}\,u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]\,\boxed{\mathtt{L}}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\,\boxed{\mathtt{L}}\,] \quad \rightarrow_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}\,\boxed{\mathtt{L}}$$

Let $\Delta := \lambda x.xx$. Then,

$$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \quad \rightarrow_{\mathtt{db}} \quad \Delta\,\boxed{[z\backslash y\,y]}\,\Delta \quad \rightarrow_{\mathtt{db}} \quad (xx)[x\backslash\Delta][z\backslash y\,y]$$

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\boxed{\mathtt{L}}\,u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]\,\boxed{\mathtt{L}}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\,\boxed{\mathtt{L}}\,] \quad \rightarrow_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}\,\boxed{\mathtt{L}}$$

Let $\Delta := \lambda x.xx$. Then,

$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \quad \rightarrow_{\mathtt{db}} \quad \Delta\,[z\backslash y\,y]\,\Delta \quad \rightarrow_{\mathtt{db}} \quad (xx)[x\backslash\Delta][z\backslash y\,y] \quad \rightarrow_{\mathtt{sV}} \quad (\Delta\,\Delta)[z\backslash y\,y]$

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\mathtt{L}\,u \quad \rightarrow_{\mathtt{db}} \quad t[x\backslash u]\,\mathtt{L}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\,\mathtt{L}\,] \quad \rightarrow_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}\,\mathtt{L}$$

Let $\Delta := \lambda x.xx$. Then,

$$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \rightarrow_{\mathtt{db}} \Delta\,\boxed{[z\backslash y\,y]}\,\Delta \rightarrow_{\mathtt{db}} (xx)[x\backslash\Delta][z\backslash y\,y] \rightarrow_{\mathtt{sV}}$$
$$(\Delta\,\Delta)[z\backslash y\,y] \rightarrow_{\mathtt{db}} (xx)[x\backslash\Delta][z\backslash y\,y]$$

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\boxed{\mathtt{L}}\,u \rightarrow_{\mathtt{db}} t[x\backslash u]\,\boxed{\mathtt{L}}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\,\boxed{\mathtt{L}}\,] \rightarrow_{\mathtt{sV}} t\{\!\{x\backslash \mathtt{v}\}\!\}\,\boxed{\mathtt{L}}$$

Let $\Delta := \lambda x.xx$. Then,

$$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \to_{\mathtt{db}} \Delta\ \boxed{[z\backslash y\,y]}\ \Delta \to_{\mathtt{db}} (xx)[x\backslash\Delta][z\backslash y\,y] \to_{\mathtt{sV}}$$

$$(\Delta\,\Delta)[z\backslash y\,y] \to_{\mathtt{db}} (xx)[x\backslash\Delta][z\backslash y\,y] \to_{\mathtt{sV}} \ldots$$

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\ \boxed{\mathtt{L}}\ u \to_{\mathtt{db}} t[x\backslash u]\ \boxed{\mathtt{L}}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\ \boxed{\mathtt{L}}\ ] \to_{\mathtt{sV}} t\{\!\{x\backslash \mathtt{v}\}\!\}\ \boxed{\mathtt{L}}$$

Let $\Delta := \lambda x.xx$. Then,

$$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \ \rightarrow_{\mathtt{db}}\ \Delta\,\boxed{[z\backslash y\,y]}\,\Delta \ \rightarrow_{\mathtt{db}}\ (xx)[x\backslash\Delta][z\backslash y\,y] \ \rightarrow_{\mathtt{sV}}$$
$$(\Delta\,\Delta)[z\backslash y\,y] \ \rightarrow_{\mathtt{db}}\ (xx)[x\backslash\Delta][z\backslash y\,y] \ \rightarrow_{\mathtt{sV}}\ \ldots$$

$(\mathbf{2})$ $(xx)[x\backslash y\Delta]$ is already a normal form

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\,\boxed{\mathtt{L}}\,u \ \rightarrow_{\mathtt{db}}\ t[x\backslash u]\,\boxed{\mathtt{L}}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathrm{v}\,\boxed{\mathtt{L}}\,] \ \rightarrow_{\mathtt{sV}}\ t\{\!\{x\backslash \mathrm{v}\}\!\}\,\boxed{\mathtt{L}}$$

Let $\Delta := \lambda x.xx$. Then,

$$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \rightarrow_{\mathtt{db}} \Delta\ \boxed{[z\backslash y\,y]}\ \Delta \rightarrow_{\mathtt{db}} (xx)[x\backslash\Delta][z\backslash y\,y] \rightarrow_{\mathtt{sV}}$$
$$(\Delta\,\Delta)[z\backslash y\,y] \rightarrow_{\mathtt{db}} (xx)[x\backslash\Delta][z\backslash y\,y] \rightarrow_{\mathtt{sV}} \ldots$$

$(\mathbf{2})$ $(xx)[x\backslash y\Delta]$ is already a normal form : **Structures Remain Shared**

Distant Operational Semantics

Distant Beta:
(multiplicative step)

$$(\lambda x.t)\ \mathtt{L}\ u \rightarrow_{\mathtt{db}} t[x\backslash u]\ \mathtt{L}$$

Distant (Full) Value Substitution:
(exponential step)

$$t[x\backslash \mathtt{v}\ \mathtt{L}\ ] \rightarrow_{\mathtt{sV}} t\{\!\{x\backslash \mathtt{v}\}\!\}\ \mathtt{L}$$

Let $\Delta := \lambda x.xx$. Then,

$(\mathbf{1})\ t = (\lambda z.\Delta)\,(y\,y)\,\Delta \quad \to_{\mathtt{db}} \quad \Delta\ [z\backslash y\,y]\ \Delta \quad \to \ldots\ ][z\backslash y\,y] \quad \to_{\mathtt{sV}}$

$(\Delta\,\Delta)[z\backslash y\,y] \quad \to_{\mathtt{db}} \quad (xx)[x\backslash\Delta][z\backslash \ldots$

$(\mathbf{2})\ (xx)[x\backslash y\Delta]$ is already a … **ared**

Reduction is WEAK:

outside λ-abstractions

… $t[x\backslash u]\ \mathtt{L}$

…on:

…ntial step)

$t[x\backslash \mathtt{v}\ \mathtt{L}\ ] \quad \to_{\mathtt{sV}} \quad t\{\!\{x\backslash \mathtt{v}\}\!\}\ \mathtt{L}$

(Full) substitution becomes linear

(Full) substitution becomes linear

$t[x \backslash v] \rightarrow_{\mathtt{sV}} \mathtt{t}\{\!\{\mathtt{x} \backslash \mathtt{v}\}\!\}$

(Full) substitution becomes linear

$t[x\backslash \mathtt{v}] \rightarrow_{\mathtt{sV}} \mathtt{t}\{\!\{\mathbf{x}\backslash \mathtt{v}\}\!\}$ becomes

$$\begin{array}{lll} (\ldots x \ldots \boxed{x} \ldots)[x\backslash \mathtt{v}] & \rightarrow_{\mathtt{lsV}} & (\ldots x \ldots \mathtt{v} \ldots)[\mathbf{x}\backslash \mathtt{v}] = \\ (\ldots \boxed{x} \ldots \mathtt{v} \ldots)[\mathbf{x}\backslash \mathtt{v}] & \rightarrow_{\mathtt{lsV}} & (\ldots \mathtt{v} \ldots \mathtt{v} \ldots)[\mathbf{x}\backslash \mathtt{v}] \end{array}$$

From VSC to the Linear Value Substitution Calculus

(Full) substitution becomes linear

$t[x\backslash v] \rightarrow_{sV} t\{\!\{x\backslash v\}\!\}$ becomes

$$(\ldots x \ldots \boxed{x} \ldots)[x\backslash v] \rightarrow_{lsV} (\ldots x \ldots v \ldots)[x\backslash v] =$$
$$(\ldots \boxed{x} \ldots v \ldots)[x\backslash v] \rightarrow_{lsV} (\ldots v \ldots v \ldots)[x\backslash v]$$

The Linear Call-by-Value Substitution Calculus LinearVSC:

(multiplicative step)

$$(\lambda x.t)\, \boxed{L}\, u \rightarrow_{db} t[x\backslash u]\, \boxed{L}$$

(exponential step)

$$C\langle\!\langle \boxed{x} \rangle\!\rangle[x\backslash v\, \boxed{L}\,] \rightarrow_{lsV} C\langle\!\langle v \rangle\!\rangle[x\backslash v]\, \boxed{L}$$

(Full) substitution becomes linear

$t[x\backslash \mathtt{v}] \rightarrow_{\mathtt{sV}} \mathtt{t}\{\{\mathtt{x}\backslash \mathtt{v}\}\}$ becomes

$$(\ldots x \ldots \boxed{x} \ldots)[x\backslash \mathtt{v}] \rightarrow_{\mathtt{lsV}} (\ldots x \ldots \mathtt{v} \ldots)[\mathtt{x}\backslash \mathtt{v}] =$$
$$(\ldots \boxed{x} \ldots \mathtt{v} \ldots)[\mathtt{x}\backslash \mathtt{v}] \rightarrow_{\mathtt{lsV}} (\ldots \mathtt{v} \ldots \mathtt{v} \ldots)[\mathtt{x}\backslash \mathtt{v}]$$

The Linear Call-by-Value Substitution Calculus $\boxed{\mathtt{LinearVSC}}$:

(multiplicative step)

$$(\lambda x.t)\boxed{\mathtt{L}}\, u \rightarrow_{\mathtt{db}} t[x\backslash u]\boxed{\mathtt{L}}$$

(exponential step)

$$\mathtt{C}\langle\langle \boxed{x} \rangle\rangle[x\backslash \mathtt{v}\boxed{\mathtt{L}}] \rightarrow_{\mathtt{lsV}} \mathtt{C}\langle\langle \mathtt{v} \rangle\rangle[\mathtt{x}\backslash \mathtt{v}]\boxed{\mathtt{L}}$$

Example:

Full Substitution

$$(y\ \boxed{x\ x})[x\backslash \mathtt{I}] \rightarrow_{\mathtt{sV}} y\ \mathtt{I}\ \mathtt{I}$$

Linear Substitution

$$(y\ \boxed{x}\ x)[x\backslash \mathtt{I}] \rightarrow_{\mathtt{lsV}} (y\ \mathtt{I}\ \boxed{x})[x\backslash \mathtt{I}]$$
$$\rightarrow_{\mathtt{lsV}} (y\ \mathtt{I}\ \mathtt{I})[x\backslash \mathtt{I}]$$

(Linear) substitution is only allowed if creation of function application:

Towards the Useful Call-by-Value Substitution Calculus

(Linear) substitution is only allowed if creation of function application:

Direct Creation: $\boxed{x}[x\backslash \mathtt{I}]y \to \mathtt{I}[x\backslash \mathtt{I}]y \to_{\mathrm{db}} [z\backslash y][x\backslash \mathtt{I}]y$

Towards the Useful Call-by-Value Substitution Calculus

(Linear) substitution is only allowed if creation of function application:

Direct Creation: $\boxed{x}[x\backslash \mathtt{I}]y \to \mathtt{I}[x\backslash \mathtt{I}]y \to_{\mathtt{db}} [z\backslash y][x\backslash \mathtt{I}]y$

Indirect Creation: $\boxed{x}[x\backslash w][w\backslash \mathtt{I}]y \to \boxed{w}[x\backslash w][w\backslash \mathtt{I}]y$
$\to \mathtt{I}[x\backslash w][w\backslash \mathtt{I}]y \to_{\mathtt{db}} z[z\backslash y][x\backslash w][w\backslash \mathtt{I}]$

Towards the Useful Call-by-Value Substitution Calculus

(Linear) substitution is only allowed if creation of function application:

Direct Creation: $\boxed{x}[x\backslash \mathtt{I}]y \to \mathtt{I}[x\backslash \mathtt{I}]y \to_{\mathrm{db}} [z\backslash y][x\backslash \mathtt{I}]y$

Indirect Creation: $\boxed{x}[x\backslash w][w\backslash \mathtt{I}]y \to \boxed{w}[x\backslash w][w\backslash \mathtt{I}]y$
$\to \mathtt{I}[x\backslash w][w\backslash \mathtt{I}]y \to_{\mathrm{db}} z[z\backslash y][x\backslash w][w\backslash \mathtt{I}]$

No Creation: $\boxed{x}[x\backslash \mathtt{I}] \to \mathtt{I}[x\backslash \mathtt{I}]$

(Linear) substitution is only allowed if creation of function application:

✓ **Direct Creation:** $\boxed{x}[x\backslash \mathtt{I}]y \to \mathtt{I}[x\backslash \mathtt{I}]y \to_{\mathrm{db}} [z\backslash y][x\backslash \mathtt{I}]y$

✓ **Indirect Creation:** $\boxed{x}[x\backslash w][w\backslash \mathtt{I}]y \to \boxed{w}[x\backslash w][w\backslash \mathtt{I}]y$
$\to \mathtt{I}[x\backslash w][w\backslash \mathtt{I}]y \to_{\mathrm{db}} z[z\backslash y][x\backslash w][w\backslash \mathtt{I}]$

✗ **No Creation:** $\boxed{x}[x\backslash \mathtt{I}] \to \mathtt{I}[x\backslash \mathtt{I}]$

- Usefulness was introduced to define **reasonable cost models**.

(Linear) substitution is only allowed if creation of function application:

Direct Creation: $\boxed{x}[x\backslash \mathtt{I}]y \to \mathtt{I}[x\backslash \mathtt{I}]y \to_{\mathrm{db}} [z\backslash y][x\backslash \mathtt{I}]y$

Indirect Creation: $\boxed{x}[x\backslash w][w\backslash \mathtt{I}]y \to \boxed{w}[x\backslash w][w\backslash \mathtt{I}]y$
$\to \mathtt{I}[x\backslash w][w\backslash \mathtt{I}]y \to_{\mathrm{db}} z[z\backslash y][x\backslash w][w\backslash \mathtt{I}]$

No Creation: $\boxed{x}[x\backslash \mathtt{I}] \to \mathtt{I}[x\backslash \mathtt{I}]$

- Usefulness was introduced to define **reasonable cost models**.
- Usefulness can be seen as an **optimized evaluation strategy**.

Towards the Useful Call-by-Value Substitution Calculus

(Linear) substitution is only allowed if creation of function application:

Direct Creation: $\boxed{x}\,[x\backslash \mathtt{I}]y \to \mathtt{I}[x\backslash \mathtt{I}]y \to_{\mathrm{db}} [z\backslash y][x\backslash \mathtt{I}]y$

Indirect Creation: $\boxed{x}\,[x\backslash w][w\backslash \mathtt{I}]y \to \boxed{w}\,[x\backslash w][w\backslash \mathtt{I}]y$
$\to \mathtt{I}[x\backslash w][w\backslash \mathtt{I}]y \to_{\mathrm{db}} z[z\backslash y][x\backslash w][w\backslash \mathtt{I}]$

No Creation: $\boxed{x}\,[x\backslash \mathtt{I}] \to \mathtt{I}[x\backslash \mathtt{I}]$

- Usefulness was introduced to define **reasonable cost models**.
- Usefulness can be seen as an **optimized evaluation strategy**.
- **(Accattoli-DalLago'14)**:
  Original formulation of **useful evaluation** for **Call-by-Name**

Towards the Useful Call-by-Value Substitution Calculus

(Linear) substitution is only allowed if creation of function application:

Direct Creation: $\boxed{x}\,[x\backslash \mathtt{I}]y \to \mathtt{I}[x\backslash \mathtt{I}]y \to_{\mathrm{db}} [z\backslash y][x\backslash \mathtt{I}]y$

Indirect Creation: $\boxed{x}\,[x\backslash w][w\backslash \mathtt{I}]y \to \boxed{w}\,[x\backslash w][w\backslash \mathtt{I}]y$
$\to \mathtt{I}[x\backslash w][w\backslash \mathtt{I}]y \to_{\mathrm{db}} z[z\backslash y][x\backslash w][w\backslash \mathtt{I}]$

No Creation: $\boxed{x}\,[x\backslash \mathtt{I}] \to \mathtt{I}[x\backslash \mathtt{I}]$

- Usefulness was introduced to define **reasonable cost models**.
- Usefulness can be seen as an **optimized evaluation strategy**.
- **(Accattoli-DalLago'14)**:
  Original formulation of **useful evaluation** for **Call-by-Name**
- **(Accattoli-SacerdottiCoen'15, Accattoli-Guerrieri'17)**:
  Original formulations of **useful evaluation** for **Call-by-Value**

Towards the Useful Call-by-Value Substitution Calculus

(Linear) substitution is only allowed if creation of function application:

Direct Creation: $x[x\backslash \mathtt{I}]y \to \mathtt{I}[x\backslash \mathtt{I}]y \to_{\mathrm{db}} [z\backslash y][x\backslash \mathtt{I}]y$

Indirect Creation: $x[x\backslash w][w\backslash \mathtt{I}]y \to w[x\backslash w][w\backslash \mathtt{I}]y$
$\to \mathtt{I}[x\backslash w][w\backslash \mathtt{I}]y \to_{\mathrm{db}} z[z\backslash y][x\backslash w][w\backslash \mathtt{I}]$

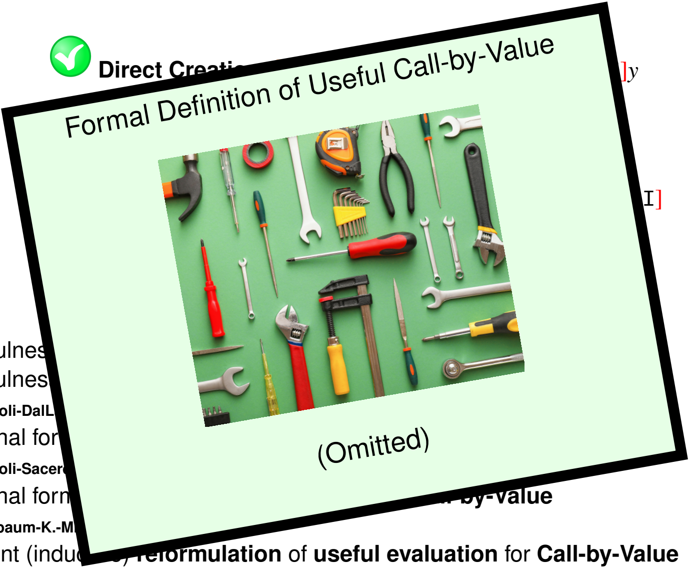
No Creation: $x[x\backslash \mathtt{I}] \to \mathtt{I}[x\backslash \mathtt{I}]$

- Usefulness was introduced to define **reasonable cost models**.
- Usefulness can be seen as an **optimized evaluation strategy**.
- **(Accattoli-DalLago'14)**:
  Original formulation of **useful evaluation** for **Call-by-Name**
- **(Accattoli-SacerdottiCoen'15, Accattoli-Guerrieri'17)**:
  Original formulations of **useful evaluation** for **Call-by-Value**
- **(Barenbaum-K.-Milicich'25)**:
  Recent (inductive) **reformulation** of **useful evaluation** for **Call-by-Value**

(Linear) substitution is only allowed if creation of function application:

Direct Creatio ...]y

I]



- Usefulnes
- Usefulnes
- **(Accattoli-DalL**
  Original for
- **(Accattoli-Sacer**
  Original form ... **by-Value**
- **(Barenbaum-K.-M**
  Recent (induc... **reformulation** of **useful evaluation** for **Call-by-Value**

Three Different Call-by-Value Operational Semantics

- `VSC`: The Call-by-Value Substitution Calculus
  (Accattoli and Paolini'12)
  - Uses **full** substitution (and thus erases values)
- `LinearVSC`: The Linear Call-by-Value Substitution Calculus
  - Uses **linear** substitution (and thus is non-erasing)
  - Implementation oriented.
- `UsefulVSC`: The Useful Call-by-Value Substitution Calculus
  (Barenbaum-K.-Milicich'25)
  - Uses **linear** and **useful** substitution
  - Call-by-Value optimization.

Three Different Call-by-Value Operational Semantics

- `VSC`: The Call-by-Value Substitution Calculus

  (Accattoli and Paolini'12)
  - Uses **full** substitution (and thus erases values)
- `LinearVSC`: The Linear Call-by-Value Substitution Calculus
  - Uses **linear** substitution (and thus is non-erasing)
  - Implementation oriented.
- `UsefulVSC`: The Useful Call-by-Value Substitution Calculus

  (Barenbaum-K.-Milicich'25)
  - Uses **linear** and **useful** substitution
  - Call-by-Value optimization.

Three Different Call-by-Value Operational Semantics

- `VSC`: The Call-by-Value Substitution Calculus

  (Accattoli and Paolini'12)
  - Uses **full** substitution (and thus erases values)
- `LinearVSC`: The Linear Call-by-Value Substitution Calculus
  - Uses **linear** substitution (and thus is non-erasing)
  - Implementation oriented.
- `UsefulVSC`: The Useful Call-by-Value Substitution Calculus

  (Barenbaum-K.-Milicich'25)
  - Uses **linear** and **useful** substitution
  - Call-by-Value optimization.

Agenda

Observational Equivalence for Call-by-Value

Call-by-Value
`VSC`

Linear Call-by-Value
`LinearVSC`

Useful Call-by-Value
`UsefulVSC`

?

Call-by-Value	Linear Call-by-Value	Useful Call-by-Value
`VSC`	`LinearVSC`	`UsefulVSC`

?

SAME typing system (called $\mathcal{V}$)
to capture **DIFFERENT** models of CbV computation

Based on Ehrhard's Relational Semantics for Plotkin's $\mathtt{CbV}$

Grammar:

$$
\begin{array}{llll}
\textbf{(Types)} & \mathtt{A} & ::= & \iota \mid \mathcal{M} \mid \mathcal{M} \Rightarrow \mathtt{A} \\
\textbf{(Multi-Types)} & \mathcal{M} & ::= & [\mathtt{A_i}]_{i \in I} \; I \text{ finite set}
\end{array}
$$

Grammar:

$$\begin{array}{llll} \textbf{(Types)} & \mathtt{A} & ::= & \iota \mid \mathcal{M} \mid \mathcal{M} \Rightarrow \mathtt{A} \\ \textbf{(Multi-Types)} & \mathcal{M} & ::= & [\mathtt{A_i}]_{i \in I} \ I \text{ finite set} \end{array}$$

Judgements:

Multi-Type $\mathcal{M}_{\mathbf{i}}$ for each variable x_i

$$x_1 : \mathcal{M}_{\mathbf{1}}, \ldots, x_n : \mathcal{M}_{\mathbf{n}} \vdash \mathbf{t} : \mathtt{A}$$

Type $\mathtt{A}$ for the term $\mathbf{t}$

The Typing Rules:

$$\frac{}{x : \mathcal{M} \vdash x : \mathcal{M}}\ (\mathtt{ax}) \qquad \frac{(\Gamma_i; x : \mathcal{M}_\mathbf{i} \vdash t : \mathtt{A}_\mathbf{i})_{i\in I}}{\sqcup_{i\in I}\ \Gamma_i \vdash \lambda x.t : [\mathcal{M}_\mathbf{i} \Rightarrow \mathtt{A}_\mathbf{i}]_{i\in I}}\ (\mathtt{fun})$$

$$\frac{\Gamma \vdash t : [\mathcal{M} \Rightarrow \mathtt{A}] \qquad \Delta \vdash u : \mathcal{M}}{\Gamma\ \sqcup\ \Delta \vdash tu : \mathtt{A}}\ (\mathtt{app}) \qquad \frac{\Gamma; x : \mathcal{M} \vdash t : \mathtt{A} \qquad \Delta \vdash u : \mathcal{M}}{\Gamma\ \sqcup\ \Delta \vdash t[x\backslash u] : \mathtt{A}}\ (\mathtt{es})$$

The Typing Rules:

$$\frac{}{x : \mathcal{M} \vdash x : \mathcal{M}}\ (\mathtt{ax}) \qquad \frac{(\Gamma_i; x : \mathcal{M}_\mathtt{i} \vdash t : \mathtt{A_i})_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash \lambda x.t : [\mathcal{M}_\mathtt{i} \Rightarrow \mathtt{A_i}]_{i \in I}}\ (\mathtt{fun})$$

$$\frac{\Gamma \vdash t : [\mathcal{M} \Rightarrow \mathtt{A}] \qquad \Delta \vdash u : \mathcal{M}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{A}}\ (\mathtt{app}) \qquad \frac{\Gamma; x : \mathcal{M} \vdash t : \mathtt{A} \qquad \Delta \vdash u : \mathcal{M}}{\Gamma \sqcup \Delta \vdash t[x \backslash u] : \mathtt{A}}\ (\mathtt{es})$$

- **Relevant** axiom (no weakening).

The Typing Rules:

$$\frac{}{x : \mathcal{M} \vdash x : \mathcal{M}}\ (\mathtt{ax}) \qquad \frac{(\Gamma_i; x : \mathcal{M}_\mathbf{i} \vdash t : \mathtt{A_i})_{i\in I}}{\boxed{\sqcup_{i\in I} \Gamma_i} \vdash \lambda x.t : [\mathcal{M}_\mathbf{i} \Rightarrow \mathtt{A_i}]_{i\in I}}\ (\mathtt{fun})$$

$$\frac{\Gamma \vdash t : [\mathcal{M} \Rightarrow \mathtt{A}] \qquad \Delta \vdash u : \mathcal{M}}{\boxed{\Gamma \sqcup \Delta} \vdash tu : \mathtt{A}}\ (\mathtt{app}) \qquad \frac{\Gamma; x : \mathcal{M} \vdash t : \mathtt{A} \qquad \Delta \vdash u : \mathcal{M}}{\boxed{\Gamma \sqcup \Delta} \vdash t[x\backslash u] : \mathtt{A}}\ (\mathtt{es})$$

- **Multiplicative** rule.
- **Operation** $(x : \mathcal{M}_1) \sqcup (x : \mathcal{M}_2)$ yields multiset **union** for multi-types $x : \mathcal{M}_1 \sqcup \mathcal{M}_2$.

Type System $\mathcal{V}$

The Typing Rules:

$$\frac{}{x : \mathcal{M} \vdash x : \mathcal{M}}\ (\mathtt{ax}) \qquad \frac{(\Gamma_i ; x : \mathcal{M}_{\mathtt{i}} \vdash t : \mathtt{A_i})_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash \lambda x.t : \boxed{[\mathcal{M}_{\mathtt{i}} \Rightarrow \mathtt{A_i}]_{i \in I}}}\ (\mathtt{fun})$$

$$\frac{\Gamma \vdash t : [\mathcal{M} \Rightarrow \mathtt{A}] \qquad \Delta \vdash u : \mathcal{M}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{A}}\ (\mathtt{app}) \qquad \frac{\Gamma ; x : \mathcal{M} \vdash t : \mathtt{A} \qquad \Delta \vdash u : \mathcal{M}}{\Gamma \sqcup \Delta \vdash t[x \backslash u] : \mathtt{A}}\ (\mathtt{es})$$

- **Erasing** abstractions are typed with $[[\,] \Rightarrow \mathtt{A}]$.

The Typing Rules:

$$\frac{}{x : \mathcal{M} \vdash x : \mathcal{M}}\ (\mathtt{ax}) \qquad \frac{(\Gamma_i ; x : \mathcal{M}_\mathbf{i} \vdash t : \mathtt{A}_\mathbf{i})_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash \lambda x.t : [\mathcal{M}_\mathbf{i} \Rightarrow \mathtt{A}_\mathbf{i}]_{i \in I}}\ (\mathtt{fun})$$

$$\frac{\Gamma \vdash t : [\mathcal{M} \Rightarrow \mathtt{A}] \qquad \Delta \vdash u : \mathcal{M}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{A}}\ (\mathtt{app}) \qquad \frac{\Gamma ; x : \mathcal{M} \vdash t : \mathtt{A} \qquad \Delta \vdash u : \mathcal{M}}{\Gamma \sqcup \Delta \vdash t[x \backslash u] : \mathtt{A}}\ (\mathtt{es})$$

- **Syntax Directed** system.

Example: Church Numerals

Let $\underline{\mathbf{3}} := \lambda f.\lambda x.f(f(f\,x)))$ and let $\mathtt{B} := [\,] \Rightarrow [\,]$.

$$
\dfrac{
  \dfrac{}{f : [\mathtt{B}] \vdash f : [\mathtt{B}]}
  \qquad
  \dfrac{
    \dfrac{}{f : [\mathtt{B}] \vdash f : [\mathtt{B}]}
    \qquad
    \dfrac{
      \dfrac{}{f : [\mathtt{B}] \vdash f : [\mathtt{B}]}
      \qquad
      \dfrac{}{x : [\,] \vdash x : [\,]}
    }{f : [\mathtt{B}], x : [\,] \vdash f\,x : [\,]}
  }{f : [\mathtt{B}, \mathtt{B}], x : [\,] \vdash f(f\,x)) : [\,]}
}{
  \dfrac{f : [\mathtt{B}, \mathtt{B}, \mathtt{B}], x : [\,] \vdash f(f(f\,x))) : [\,]}{
    \dfrac{f : [\mathtt{B}, \mathtt{B}, \mathtt{B}] \vdash \lambda x.f(f(f\,x))) : [\,] \Rightarrow [\,]}{\vdash \underline{\mathbf{3}} : [\mathtt{B}, \mathtt{B}, \mathtt{B}] \Rightarrow [\,] \Rightarrow [\,]}
  }
}
$$

Let $\underline{\mathbf{3}} := \lambda f.\lambda x.\, f\,(\,f\,(\,f\,x)))$ and let $\mathtt{B} := [\,] \Rightarrow [\,]$.

$$
\dfrac{
  \dfrac{}{f : [\mathtt{B}] \vdash f : [\mathtt{B}]}
  \qquad
  \dfrac{
    \dfrac{}{f : [\mathtt{B}] \vdash f : [\mathtt{B}]}
    \qquad
    \dfrac{
      \dfrac{}{f : [\mathtt{B}] \vdash f : [\mathtt{B}]}
      \qquad
      \dfrac{}{x : [\,] \vdash x : [\,]}
    }{f : [\mathtt{B}], x : [\,] \vdash f\,x : [\,]}
  }{f : [\mathtt{B}, \mathtt{B}], x : [\,] \vdash f(f\,x)) : [\,]}
}{
  \dfrac{
    \dfrac{f : [\mathtt{B}, \mathtt{B}, \mathtt{B}], x : [\,] \vdash f(f(f\,x))) : [\,]}{f : [\mathtt{B}, \mathtt{B}, \mathtt{B}] \vdash \lambda x.f(f(f\,x))) : [\,] \Rightarrow [\,]}
  }{\vdash \underline{\mathbf{3}} : [\,\mathtt{B}\,,\,\mathtt{B}\,,\,\mathtt{B}\,] \Rightarrow [\,] \Rightarrow [\,]}
}
$$

SAME typing system
to capture **DIFFERENT** models of CbV computation

Theorem

$\mathcal{V}$-*Typability* $\iff$ VSC-*Termination.*

$\mathcal{V}$-*Typability* $\iff$ LinearVSC-*Termination.*

$\mathcal{V}$-*Typability* $\iff$ UsefulVSC-*Termination.*

Call-by-Value Observational Equivalence by Means of Types

SAME typing system
to capture **DIFFERENT** models of ... mputation

Theorem

- $\mathcal{V}$-*Typability* ...
- $\mathcal{V}$-*Typ*...
- $\mathcal{V}$-*Typa*...

(Bucciarelli-K.-Ríos-Viso'20, Barenbaum-K.-Milicich'25)

`VSC` $\cong$ `LinearVSC` $\cong$ `UsefulVSC`

Agenda

Call-by-Value, Linear Call-by-Value and Useful Call-by-Value are **qualitatively** equivalent

`VSC` `LinearVSC` `UsefulVSC`

Call-by-Value, Linear Call-by-Value and Useful Call-by-Value are

qualitatively equivalent

but not **quantitavely** equivalent

`VSC` `LinearVSC` `UsefulVSC`

`VSC`

`LinearVSC`

`UsefulVSC`

Intersection Types and Quantitative Analysis

Call-by-Value, Linear Call-by-Value and Useful Call-by-Value are
qualitatively equivalent
but not **quantitavely** equivalent

`VSC` `LinearVSC` `UsefulVSC`

`VSC`

`LinearVSC`

`UsefulVSC`

Normalization sequences have **different** lengths:

Call-by-Value, Linear Call-by-Value and Useful Call-by-Value are
qualitatively equivalent
but not **quantitavely** equivalent

VSC LinearVSC UsefulVSC

VSC

LinearVSC

UsefulVSC

Normalization sequences have **different** lengths:

VSC $(\mathbf{3}) : (xyx)[x\backslash \mathtt{I}] \rightarrow \mathtt{I}y\mathtt{I} \rightarrow z[z\backslash y]\mathtt{I} \rightarrow y\mathtt{I}$

LinearVSC $(\mathbf{5}) : (xyx)[x\backslash \mathtt{I}] \rightarrow (xy\mathtt{I})[x\backslash \mathtt{I}] \rightarrow (\mathtt{I}y\mathtt{I})[x\backslash \mathtt{I}] \rightarrow (z[z\backslash y]\mathtt{I})[x\backslash \mathtt{I}] \rightarrow (y[z\backslash y]\mathtt{I})[x\backslash \mathtt{I}]$

UsefulVSC $(\mathbf{2}) : (xyx)[x\backslash \mathtt{I}] \rightarrow (\mathtt{I}yx)[x\backslash \mathtt{I}] \rightarrow (z[z\backslash y]x)[x\backslash \mathtt{I}]$

Call-by-Value, Linear Call-by-Value and Useful Call-by-Value are
qualitatively equivalent
but not **quantitavely** equivalent

VSC LinearVSC UsefulVSC

VSC

LinearVSC

UsefulVSC

Normalization sequences have **different** lengths:

$$\texttt{VSC}\ (\mathbf{3}) : (xyx)[x\backslash \texttt{I}] \to \texttt{I}y\texttt{I} \to z[z\backslash y]\texttt{I} \to y\texttt{I}$$

$$\texttt{LinearVSC}\ (\mathbf{5}) : (xyx)[x\backslash \texttt{I}] \to (xy\texttt{I})[x\backslash \texttt{I}] \to (\texttt{I}y\texttt{I})[x\backslash \texttt{I}] \to (z[z\backslash y]\texttt{I})[x\backslash \texttt{I}] \to (y[z\backslash y]\texttt{I})[x\backslash \texttt{I}]$$

$$\texttt{UsefulVSC}\ (\mathbf{2}) : (xyx)[x\backslash \texttt{I}] \to (\texttt{I}yx)[x\backslash \texttt{I}] \to (z[z\backslash y]x)[x\backslash \texttt{I}]$$

The corresponding **normal forms** are all equivalent **modulo unfolding**.

Characterizing Evaluation by Means of **Qualitative**/**Quantitative** Types

Qualitative Intersection Types

Qualitative Intersection Types ✓ ✗

Typability $\Longleftrightarrow$ **Termination**

Quantitative Intersection Types

(de Carvalho, Bernadet-Lengrand)

Typability $\Longleftrightarrow$ **Termination**

- $\Gamma \vdash^{(C_1,\ldots,C_n)} t : A \quad \Longleftrightarrow \quad t \; \underbrace{\rightarrow \ldots \rightarrow}_{\text{length } C_1 + \ldots + C_n}$ **normal form**

Quantitative Intersection Types

(de Carvalho, Bernadet-Lengrand)

Typability $\iff$ **Termination**

- $\Gamma \vdash^{(C_1,\dots,C_n)} t : A \iff t \underbrace{\to \dots \to}_{\text{length } C_1 + \dots + C_n}$ **normal form**
- $\Gamma \vdash^{(m+e)} t : A \overset{n=1}{\iff} t \underbrace{\to \dots \to}_{\text{length } m + e}$ **normal form**

m counts **multiplicative** and e counts **exponential** steps.

Quantitative Intersection Types (with SPLIT MEASURES)

(Accattoli-GrahamLengrand-K. ICFP'18) 

Typability $\Longleftrightarrow$ **Termination**

- $\Gamma \vdash^{(C_1,\ldots,C_n)} t : A \quad \Longleftrightarrow \quad t \underbrace{\rightarrow \ldots \rightarrow}_{\text{length } C_1 + \ldots + C_n}$ **normal form**

- $\Gamma \vdash^{(m+e)} t : A \quad \overset{n=1}{\Longleftrightarrow} \quad t \underbrace{\rightarrow \ldots \rightarrow}_{\text{length } m + e}$ **normal form**

m counts **multiplicative** and e counts **exponential** steps.

Characterizing Evaluation by Means of **Qualitative/Quantitative** Types

Quantitative Intersection Types (with SPLIT MEASURES)

(Accattoli-GrahamLengrand-K. ICFP'18) 

Typability $\Longleftrightarrow$ **Termination**

- $\Gamma \vdash^{(C_1,\dots,C_n)} t : A \Longleftrightarrow t \underbrace{\rightarrow \dots \rightarrow}_{\text{length } C_1 + \dots + C_n}$ **normal form**

- $\Gamma \vdash^{(\mathbf{m}+\mathbf{e})} t : A \overset{n=1}{\Longleftrightarrow} t \underbrace{\rightarrow \dots \rightarrow}_{\text{length } \mathbf{m} + \mathbf{e}}$ **normal form**

- $\Gamma \vdash^{(\mathbf{m},\mathbf{e})} t : A \overset{n=2}{\Longleftrightarrow} t \underbrace{\rightarrow \dots \rightarrow}_{\text{length } \mathbf{m} + \mathbf{e}}$ **normal form**

$\mathbf{m}$ counts **multiplicative** and $\mathbf{e}$ counts **exponential** steps.

Typability Characterizes Quantitative Properties of Languages

This scheme applies to

This scheme applies to

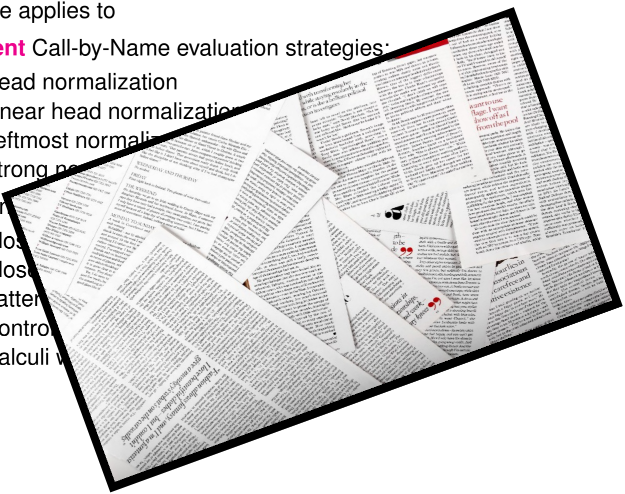
- **Different** Call-by-Name evaluation strategies:
  - Head normalization
  - Linear head normalization
  - Leftmost normalization
  - Strong normalization

This scheme applies to

- **Different** Call-by-Name evaluation strategies:
  - Head normalization
  - Linear head normalization
  - Leftmost normalization
  - Strong normalization
- Different **Expressiveness Features** :
  - Closed Call-by-Value
  - Closed Call-by-Need
  - Pattern matching features
  - Control operators
  - Calculi with effects

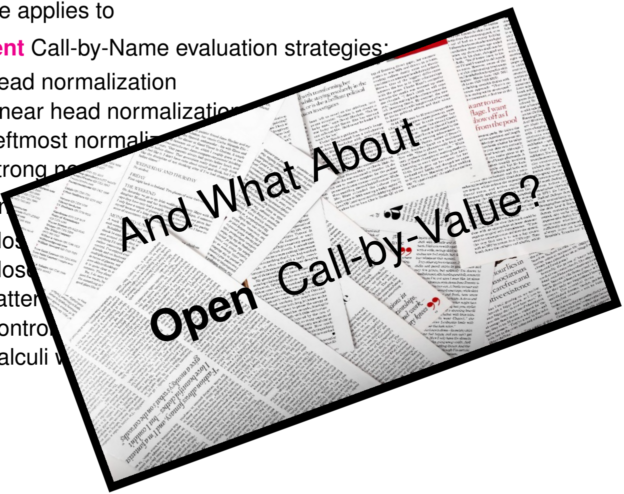
This scheme applies to

- **Different** Call-by-Name evaluation strategies:
  - Head normalization
  - Linear head normalizatio
  - Leftmost normali
  - Strong n
- Differen
  - Clos
  - Close
  - Patter
  - Contro
  - Calculi w



This scheme applies to

- **Different** Call-by-Name evaluation strategies:
  - Head normalization
  - Linear head normalizatio
  - Leftmost normali
  - Strong n
- Differen
  - Clos
  - Close
  - Patter
  - Contro
  - Calculi w



The State of the Art

- **(Accattoli-Guerrieri ESOP 19)** Exact split counting for VSC
  - Free variables/constants are not considered as values

The State of the Art

- **(Accattoli-Guerrieri ESOP 19)** Exact split counting for `VSC`
  - Free variables/constants are not considered as values
- **(K.-Viso CSL'22)** Exact split counting for `VSC`
  - Is based on the consuming/persistent model (very verbose)
  - Does not extend to `LinearVSC`

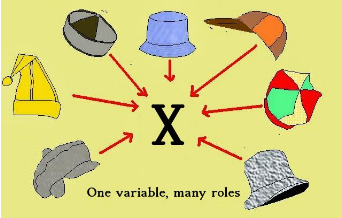
The State of the Art

- **(Accattoli-Guerrieri ESOP 19)** Exact split counting for `VSC`
  - Free variables/constants are not considered as values
- **(K.-Viso CSL'22)** Exact split counting for `VSC`
  - Is based on the consuming/persistent model (very verbose)
  - Does not extend to `LinearVSC`
- **(Barenbaum-K.-Milicich'24)**: System $\mathcal{U}$
  - Exact split counting for `UsefulVSC`

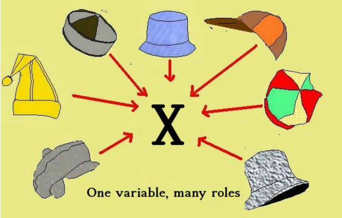
The State of the Art

- **(Accattoli-Guerrieri ESOP 19)** Exact split counting for `VSC`
  - Free variables/constants are not considered as values
- **(K.-Viso CSL'22)** Exact split counting for `VSC`
  - Is based on the consuming/persistent model (very verbose)
  - Does not extend to `LinearVSC`
- **(Barenbaum-K.-Milicich'24)**: System $\mathcal{U}$
  - Exact split counting for `UsefulVSC`
- **(Barenbaum-K.'25)**: (New) System $\mathcal{T}$
  - Values include free variables/constants
  - The same system admits:
    - Exact split counting for `VSC`
    - Exact split counting for `LinearVSC`

Main Difficulties

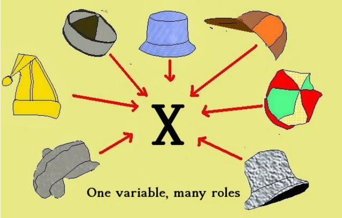


Main Difficulties



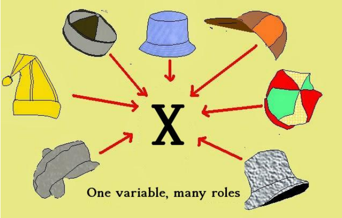
- The variable as a value e.g. $t[y\backslash x]$

Main Difficulties



- The variable as a value e.g. $t[y\backslash \boxed{x}]$
- The variable bound to a value, e.g. $t[\boxed{x}\backslash \mathrm{v}]$

Main Difficulties



- The variable as a value e.g. $t[y \backslash \boxed{x}]$
- The variable bound to a value, e.g. $t[\boxed{x} \backslash \mathrm{v}]$
- The variable bound to a structure, e.g. $t[\boxed{x} \backslash y\ z]$

Exact Counting for `UsefulVSC`
System $\mathcal{U}$

Grammar:

(**Countable Types**)	α	$::=$	$\mathtt{A}^{?} \Rightarrow \mathtt{A}$
(**Multi-types**)	$\mathcal{M}$	$::=$	$[\alpha_{\mathtt{i}}]_{i \in I}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{s} \mid \mathcal{M}$
(**Optional Types**)	$\mathtt{A}^{?}$	$::=$	$\perp \mid \mathtt{A}$

Grammar:

(**Countable Types**)	α	::=	$\mathtt{A}^{?} \Rightarrow \mathtt{A}$
(**Multi-types**)	$\mathcal{M}$	::=	$[\alpha_{\mathbf{i}}]_{i \in I}$
(**Types**)	$\mathtt{A}$	::=	$\mathtt{s} \mid \mathcal{M}$
(**Optional Types**)	$\mathtt{A}^{?}$	::=	$\perp \mid \mathtt{A}$

Three constants:

[ ] types inaccessible **values**

$\mathtt{s}$ types **structures**

$\perp$ **absence**

Grammar:

(**Countable Types**)	α	$::=$	$\mathtt{A}^? \Rightarrow \mathtt{A}$
(**Multi-types**)	$\mathcal{M}$	$::=$	$[\alpha_{\mathbf{i}}]_{i \in I}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{s} \mid \mathcal{M}$
(**Optional Types**)	$\mathtt{A}^?$	$::=$	$\bot \mid \mathtt{A}$

Judgements:

$$x_1 : \mathtt{A}^?_{\mathbf{1}}, \ldots, x_n : \mathtt{A}^?_{\mathbf{n}} \quad \vdash^{(\mathbf{m},\mathbf{e})} \quad \mathbf{t} : \mathtt{A}$$

Grammar:

(**Countable Types**)	α	$::=$	$\mathtt{A}^? \Rightarrow \mathtt{A}$
(**Multi-types**)	$\mathcal{M}$	$::=$	$[\alpha_{\mathtt{i}}]_{i \in I}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{s} \mid \mathcal{M}$
(**Optional Types**)	$\mathtt{A}^?$	$::=$	$\perp \mid \mathtt{A}$

Judgements:

Opt. Type $\mathtt{A}_{\mathtt{i}}^?$ for each variable x_i

$\downarrow$

$$x_1 : \mathtt{A}_{\mathtt{1}}^?, \ldots, x_n : \mathtt{A}_{\mathtt{n}}^? \quad \vdash^{(\mathbf{m},\mathbf{e})} \quad \mathbf{t} : \mathtt{A}$$

Grammar:

(**Countable Types**)	α	$::=$	$\mathtt{A}^? \Rightarrow \mathtt{A}$
(**Multi-types**)	$\mathcal{M}$	$::=$	$[\alpha_{\mathbf{i}}]_{i \in I}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{s} \mid \mathcal{M}$
(**Optional Types**)	$\mathtt{A}^?$	$::=$	$\bot \mid \mathtt{A}$

Judgements:

Opt. Type $\mathtt{A}^?_{\mathbf{i}}$ for each variable x_i
($x : \bot$ if x is not used in the typing context)

$\downarrow$

$$x_1 : \mathtt{A}^?_{\mathbf{1}}, \ldots, x_n : \mathtt{A}^?_{\mathbf{n}} \quad \vdash^{(\mathbf{m},\mathbf{e})} \quad \mathbf{t} : \mathtt{A}$$

Grammar:

(**Countable Types**)	α	$::=$	$\mathtt{A}^{?} \Rightarrow \mathtt{A}$
(**Multi-types**)	$\mathcal{M}$	$::=$	$[\alpha_{\mathbf{i}}]_{i \in I}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{s} \mid \mathcal{M}$
(**Optional Types**)	$\mathtt{A}^{?}$	$::=$	$\bot \mid \mathtt{A}$

Judgements:

Type $\mathtt{A}$ for the term $\mathbf{t}$

$$x_1 : \mathtt{A}^{?}_{\mathbf{1}}, \ldots, x_n : \mathtt{A}^{?}_{\mathbf{n}} \quad \vdash^{(\mathbf{m},\mathbf{e})} \quad \mathbf{t} : \mathtt{A}$$

Grammar:

(**Countable Types**)	α	$::=$	$\mathtt{A}^? \Rightarrow \mathtt{A}$
(**Multi-types**)	$\mathcal{M}$	$::=$	$[\alpha_{\mathbf{i}}]_{i \in I}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathtt{s} \mid \mathcal{M}$
(**Optional Types**)	$\mathtt{A}^?$	$::=$	$\bot \mid \mathtt{A}$

Judgements:

Counter **m** captures the **number** of **multiplicative**-steps to normal form
Counter **e** captures the **number** of **exponential**-steps to normal form

$$x_1 : \mathtt{A}^?_{\mathbf{1}}, \ldots, x_n : \mathtt{A}^?_{\mathbf{n}} \quad \vdash^{(\mathbf{m},\mathbf{e})} \quad \mathbf{t} : \mathtt{A}$$

Characterizing Useful Call-by-Value Normalization with Split Measures

Grammar:

(**Countable Types**)	α	$::=$	$\mathtt{A}^? \Rightarrow \mathtt{A}$
(**Multi-types**)	$\mathcal{M}$	$::=$	$[\alpha_{\mathbf{i}}]_{i \in I}$
(**Types**)	$\mathtt{A}$	$::=$	$\mathbb{s} \mid \mathcal{M}$
(**Optional Types**)	$\mathtt{A}^?$	$::=$	$\bot \mid \mathtt{A}$

Judgements:

Counter **m** captures the **number** of **multiplicative**-steps to normal form
Counter **e** captures the **number** of **exponential**-steps to normal form

$$x_1 : \mathtt{A}^?_{\mathbf{1}}, \ldots, x_n : \mathtt{A}^?_{\mathbf{n}} \quad \vdash^{(\mathbf{m},\mathbf{e})} \quad \mathbf{t} : \mathtt{A}$$

Tightness: A type derivation of judgement $\Gamma \vdash^{(\mathbf{m},\mathbf{e})} t : \mathtt{A}$ is **tight** (e.g. minimal) iff all the types in Γ and $\mathtt{A}$ are **constants**.

Type System $\mathcal{V}$

(Ehrhard'12)

$$\frac{}{x : \mathtt{A} \vdash x : \mathtt{A}}$$

$$\frac{(\Gamma_i; x : \mathtt{B}_i \vdash t : \mathtt{A}_i)_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash \lambda x.t : [\mathtt{B}_i \Rightarrow \mathtt{A}_i]_{i \in I}}$$

$$\frac{\Gamma \vdash t : [\mathtt{B} \Rightarrow \mathtt{A}] \qquad \Delta \vdash u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{A}}$$

$$\frac{\Gamma; x : \mathtt{B} \vdash t : \mathtt{A} \qquad \Delta \vdash u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash t[x \backslash u] : \mathtt{A}}$$

Type System $\mathcal{U}$

(Barenbaum-K.-Milicich'24)

$$\frac{}{x : \mathtt{A} \vdash x : \mathtt{A}}$$

$$\frac{(\Gamma_i; x : \mathtt{B}_{\mathtt{i}}^{?} \vdash t : \mathtt{A}_i)_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash \lambda x.t : [\mathtt{B}_{\mathtt{i}}^{?} \Rightarrow \mathtt{A}_i]_{i \in I}}$$

$$\frac{\Gamma \vdash t : [\mathtt{B}^{?} \Rightarrow \mathtt{A}] \quad \mathtt{B}^{?} \ll \mathtt{B} \quad \Delta \vdash u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{A}}$$

$$\frac{\Gamma; x : \mathtt{B}^{?} \vdash t : \mathtt{A} \quad \mathtt{B}^{?} \ll \mathtt{B} \quad \Delta \vdash u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash t[x \backslash u] : \mathtt{A}}$$

Type System $\mathcal{U}$

(Barenbaum-K.-Milicich'24)

$$\frac{}{x : \mathtt{A} \vdash \quad x : \mathtt{A}}$$

$$\frac{(\Gamma_i; x : \mathtt{B}^?_{\mathbf{i}} \vdash \quad t : \mathtt{A}_i)_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash \quad \lambda x.t : [\mathtt{B}^? \ldots}$$

Operator _ ≪ _ introduces a sort of **weakening**:

$$\bot \ll \mathtt{s}$$
$$\bot \ll [\,]$$
$$\mathcal{M} \ll \mathcal{M}$$

Type System $\mathcal{U}$

(Barenbaum-K.-Milicich'24)

$$\frac{}{x : \mathtt{A} \vdash x : \mathtt{A}}$$

$$\frac{(\Gamma_i; x : \mathtt{B}_i^? \vdash t : \mathtt{A}_i)_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash \lambda x.t : [\mathtt{B}_i^? \Rightarrow \mathtt{A}_i]_{i \in I}}$$

$$\frac{\Gamma \vdash t : [\mathtt{B}^? \Rightarrow \mathtt{A}] \quad \mathtt{B}^? \ll \mathtt{B} \quad \Delta \vdash u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{A}}$$

$$\frac{\Gamma; x : \mathtt{B}^? \vdash t : \mathtt{A} \quad \mathtt{B}^? \ll \mathtt{B} \quad \Delta \vdash u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash t[x \backslash u] : \mathtt{A}}$$

Type System $\mathcal{U}$

(Barenbaum-K.-Milicich'24)

$$\frac{}{x : \mathtt{A} \vdash x : \mathtt{A}}$$

$$\frac{(\Gamma_i ; x : \mathtt{B}_i^? \vdash t : \mathtt{A}_i)_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash \lambda x.t : [\mathtt{B}_i^? \Rightarrow \mathtt{A}_i]_{i \in I}} \qquad \frac{\Gamma \vdash t : \mathbb{s} \quad \Delta \vdash u : \mathtt{A} \in \{\mathbb{s}, [\,]\}}{\Gamma \sqcup \Delta \vdash tu : \mathbb{s}}$$

$$\frac{\Gamma \vdash t : [\mathtt{B}^? \Rightarrow \mathtt{A}] \quad \mathtt{B}^? \ll \mathtt{B} \quad \Delta \vdash u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash tu : \mathtt{A}}$$

$$\frac{\Gamma ; x : \mathtt{B}^? \vdash t : \mathtt{A} \quad \mathtt{B}^? \ll \mathtt{B} \quad \Delta \vdash u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash t[x \backslash u] : \mathtt{A}}$$

Type System $\mathcal{U}$

(Barenbaum-K.-Milicich'24)

$$\frac{\mathbf{n} = \#\text{ arrows in } \mathtt{A}}{x : \mathtt{A} \vdash^{(\mathbf{0},\mathbf{n})} x : \mathtt{A}}$$

$$\frac{(\Gamma_i; x : \mathtt{B}^?_{\mathbf{i}} \vdash^{(\mathbf{m_i},\mathbf{e_i})} t : \mathtt{A}_i)_{i \in I}}{\sqcup_{i \in I} \Gamma_i \vdash^{(+_{\mathbf{i}\in\mathbf{I}}\ \mathbf{m_i}, +_{\mathbf{i}\in\mathbf{I}}\ \mathbf{e_i})} \lambda x.t : [\mathtt{B}^?_{\mathbf{i}} \Rightarrow \mathtt{A}_i]_{i \in I}} \qquad \frac{\Gamma \vdash^{(\mathbf{m},\mathbf{e})} t : \mathtt{s} \qquad \Delta \vdash^{(\mathbf{m'},\mathbf{e'})} u : \mathtt{A} \in \{\mathtt{s}, [\,]\}}{\Gamma \sqcup \Delta \vdash^{(\mathbf{m+m'},\mathbf{e+e'})} tu : \mathtt{s}}$$

$$\frac{\Gamma \vdash^{(\mathbf{m},\mathbf{e})} t : [\mathtt{B}^? \Rightarrow \mathtt{A}] \qquad \mathtt{B}^? \ll \mathtt{B} \qquad \Delta \vdash^{(\mathbf{m'},\mathbf{e'})} u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash^{(\mathbf{1+m+m'},\mathbf{e+e'})} tu : \mathtt{A}}$$

$$\frac{\Gamma; x : \mathtt{B}^? \vdash^{(\mathbf{m},\mathbf{e})} t : \mathtt{A} \qquad \mathtt{B}^? \ll \mathtt{B} \qquad \Delta \vdash^{(\mathbf{m'},\mathbf{e'})} u : \mathtt{B}}{\Gamma \sqcup \Delta \vdash^{(\mathbf{m+m'},\mathbf{e+e'})} t[x \backslash u] : \mathtt{A}}$$

A **tight** type derivation for the term $\mathbf{t} := (x\ y\ x)[x \backslash \mathtt{I}]$

$$\frac{}{y : \mathtt{s} \vdash (x\ y\ x)[x \backslash \mathtt{I}] : \mathtt{s}}$$

A **tight** type derivation for the term $\mathbf{t} := (x\ y\ x)[x \backslash \mathtt{I}]$

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{x : [\mathbb{s} \Rightarrow \mathbb{s}] \vdash x : [\mathbb{s} \Rightarrow \mathbb{s}]}
      \qquad
      \dfrac{}{y : \mathbb{s} \vdash y : \mathbb{s}}
    }{x : [\mathbb{s} \Rightarrow \mathbb{s}], y : \mathbb{s} \vdash x\ y : \mathbb{s}}
    \qquad
    \dfrac{}{x : [\,] \vdash x : [\,]}
  }{x : [\mathbb{s} \Rightarrow \mathbb{s}], y : \mathbb{s} \vdash x\ y\ x : \mathbb{s}}
  \qquad
  \dfrac{\dfrac{}{w : \mathbb{s} \vdash w : \mathbb{s}}}{\emptyset \vdash \mathtt{I} : [\mathbb{s} \Rightarrow \mathbb{s}]}
}{y : \mathbb{s} \vdash (x\ y\ x)[x \backslash \mathtt{I}] : \mathbb{s}}
$$

A **tight** type derivation for the term $\mathbf{t} := (x\ y\ x)[x\backslash \mathtt{I}]$

$$
\cfrac{
  \cfrac{
    \cfrac{
      \cfrac{}{x : [\mathtt{s} \Rightarrow \mathtt{s}] \vdash x : [\mathtt{s} \Rightarrow \mathtt{s}]}
      \qquad
      \cfrac{}{y : \mathtt{s} \vdash y : \mathtt{s}}
    }{x : [\mathtt{s} \Rightarrow \mathtt{s}], y : \mathtt{s} \vdash x\ y : \mathtt{s}}
    \qquad
    \cfrac{}{x : [\,] \vdash x : [\,]}
  }{x : [\mathtt{s} \Rightarrow \mathtt{s}], y : \mathtt{s} \vdash x\ y\ x : \mathtt{s}}
  \qquad
  \cfrac{
    \cfrac{}{w : \mathtt{s} \vdash w : \mathtt{s}}
  }{\emptyset \vdash \mathtt{I} : [\mathtt{s} \Rightarrow \mathtt{s}]}
}{y : \mathtt{s} \vdash (x\ y\ x)[x\backslash \mathtt{I}] : \mathtt{s}}
$$

Evaluation of $\mathbf{t}$ to normal form: **1 multiplicative** step and **1 exponential** steps:

$$(x\ y\ x)[x\backslash \mathtt{I}] \to (\mathtt{I}\ y\ x)[x\backslash \mathtt{I}] \to (w[w\backslash y]\ x)[x\backslash \mathtt{I}]$$

Example

A **tight** type derivation for the term $\mathbf{t} := (x\ y\ x)[x \backslash \mathtt{I}]$

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{x : [\mathtt{s} \Rightarrow \mathtt{s}] \vdash^{(0,1)} x : [\mathtt{s} \Rightarrow \mathtt{s}]}
      \qquad
      \dfrac{}{y : \mathtt{s} \vdash^{(0,0)} y : \mathtt{s}}
    }{x : [\mathtt{s} \Rightarrow \mathtt{s}], y : \mathtt{s} \vdash^{(1,1)} x\ y : \mathtt{s}}
    \qquad
    \dfrac{}{x : [\,] \vdash^{(0,0)} x : [\,]}
  }{x : [\mathtt{s} \Rightarrow \mathtt{s}], y : \mathtt{s} \vdash^{(1,1)} x\ y\ x : \mathtt{s}}
  \qquad
  \dfrac{
    \dfrac{}{w : \mathtt{s} \vdash^{(0,0)} w : \mathtt{s}}
  }{\emptyset \vdash^{(0,0)} \mathtt{I} : [\mathtt{s} \Rightarrow \mathtt{s}]}
}{y : \mathtt{s} \vdash^{(1,1)} (x\ y\ x)[x \backslash \mathtt{I}] : \mathtt{s}}
$$

Evaluation of $\mathbf{t}$ to normal form: **1 multiplicative** step and **1 exponential** steps:

$$(x\ y\ x)[x \backslash \mathtt{I}] \rightarrow (\mathtt{I}\ y\ x)[x \backslash \mathtt{I}] \rightarrow (w[w \backslash y]\ x)[x \backslash \mathtt{I}]$$

Qualitative Versus Quantitative Subject Reduction

Let $\mathbf{t} \rightarrow_{\mathtt{UsefulVSC}} \mathbf{t}'$. Then,

Qualitative Versus Quantitative Subject Reduction

Let $\mathbf{t} \rightarrow_{\boxed{\texttt{UsefulVSC}}} \mathbf{t}'$. Then,

Qualitative Subject Reduction

If $\mathbf{t}$ is $\boxed{\mathcal{U}}$-typable then $\mathbf{t}'$ is $\boxed{\mathcal{U}}$-typable

Qualitative Versus Quantitative Subject Reduction

Let $\mathbf{t} \rightarrow_{\texttt{UsefulVSC}} \mathbf{t}'$. Then,

Qualitative Subject Reduction 

If $\mathbf{t}$ is $\mathcal{U}$-typable then $\mathbf{t}'$ is $\mathcal{U}$-typable

Quantitative Subject Reduction 

If $\mathbf{t}$ is **tightly** $\mathcal{U}$-typable with counters $(\mathbf{m}, \mathbf{e})$, and the reduction step is **multiplicative** then $\mathbf{t}'$ is **tightly** $\mathcal{U}$-typable with $\mathbb{C}$-counters $(\mathbf{m} - \mathbf{1}, \mathbf{e})$.

Qualitative Versus Quantitative Subject Reduction

Let $\mathbf{t} \rightarrow_{\texttt{UsefulVSC}} \mathbf{t}'$. Then,

Qualitative Subject Reduction 

If $\mathbf{t}$ is $\mathcal{U}$-typable then $\mathbf{t}'$ is $\mathcal{U}$-typable

Quantitative Subject Reduction 

If $\mathbf{t}$ is **tightly** $\mathcal{U}$-typable with counters $(\mathbf{m}, \mathbf{e})$, and the reduction step is **multiplicative** then $\mathbf{t}'$ is **tightly** $\mathcal{U}$-typable with $\mathbb{C}$-counters $(\mathbf{m} - \mathbf{1}, \mathbf{e})$.

If $\mathbf{t}$ is **tightly** $\mathcal{U}$-typable with counters $(\mathbf{m}, \mathbf{e})$, and the reduction step is **exponential** then $\mathbf{t}'$ is **tightly** $\mathcal{U}$-typable with $\mathbb{C}$-counters $(\mathbf{m}, \mathbf{e} - \mathbf{1})$.

Qualitative Versus Quantitative Characterization of Termination

Qualitative Versus Quantitative Characterization of Termination

Theorem (Qualitative Soundness and Completeness)

Let $\mathbf{t}$ *be a term. Then* $\mathbf{t}$ *is* $\mathcal{U}$-**typable** *if and only if* $\mathbf{t}$ `UsefulVSC`-**terminates**.

Qualitative Versus Quantitative Characterization of Termination

Theorem (Qualitative Soundness and Completeness)

Let $\mathbf{t}$ *be a term. Then* $\mathbf{t}$ *is* $\mathcal{U}$ -**typable** *if and only if* $\mathbf{t}$ `UsefulVSC` -**terminates**.

Theorem (Quantitative Soundness and Completeness)

Let $\mathbf{t}$ *be a term. Then* $\mathbf{t}$ *is tightly* $\mathcal{U}$ -**typable** *with counters* $(\mathbf{m}, \mathbf{e})$ *if and only if* $\mathbf{t}$ `UsefulVSC` -**terminates** *after* $\mathbf{m}$ **multiplicative** *steps and* $\mathbf{e}$ **exponential** *steps.*

Qualitative Versus Quantitative Characterization of Termination

Theorem (Qualitative Soundness and Completene

Let $\mathbf{t}$ *be a term. Then* $\mathbf{t}$ *is* $\mathcal{U}$ **-typable** *if an* … **inates**.

Theorem (Q

Let $\mathbf{t}$ *be a term.* … $, \mathbf{e})$ *if and only if* $\mathbf{t}$ `UsefulVSC` **-te** … *eps and* $\mathbf{e}$ **exponential** *steps.*

Similar results can be obtained for `VSC` and `LinearVSC` by means of system $\mathcal{T}$

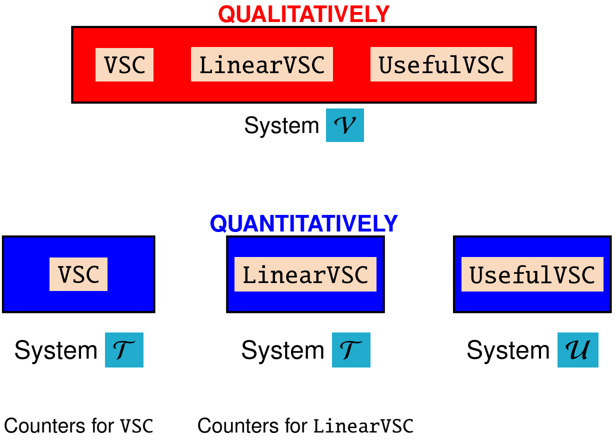
The Full Picture

QUALITATIVELY

`VSC` `LinearVSC` `UsefulVSC`

System $\mathcal{V}$

The Full Picture



Agenda

Concluding Remarks For Lesson 5

- Simple observational equivalence proof for `CbN`, `CbNeed`, `Neededness`.

Concluding Remarks For Lesson 5

- Simple observational equivalence proof for `CbN`, `CbNeed`, `Neededness`.
- Simple observational equivalence proof for `VSC`, `LinearVSC`, `UsefulVSC`.

Concluding Remarks For Lesson 5

- Simple observational equivalence proof for `CbN`, `CbNeed`, `Neededness`.
- Simple observational equivalence proof for `VSC`, `LinearVSC`, `UsefulVSC`.
- Quantitative systems with **SPLIT MEASURES** for `VSC`, `LinearVSC`, `UsefulVSC` counting exactly multiplicative and exponential steps.

Concluding Remarks For Lesson 5

- Simple observational equivalence proof for `CbN`, `CbNeed`, `Neededness`.
- Simple observational equivalence proof for `VSC`, `LinearVSC`, `UsefulVSC`.
- Quantitative systems with **SPLIT MEASURES** for `VSC`, `LinearVSC`, `UsefulVSC` counting exactly multiplicative and exponential steps.
- Case `UsefulVSC` : **complex** operational semantics, **simple** typing system.

Concluding Remarks For Lesson 5

- Simple observational equivalence proof for `CbN`, `CbNeed`, `Neededness`.
- Simple observational equivalence proof for `VSC`, `LinearVSC`, `UsefulVSC`.
- Quantitative systems with **SPLIT MEASURES** for `VSC`, `LinearVSC`, `UsefulVSC` counting exactly multiplicative and exponential steps.
- Case `UsefulVSC` : **complex** operational semantics, **simple** typing system.
- Cases `VSC`, `LinearVSC`: **simple** operational semantics, **subtle** typing system.

Concluding Remarks For Lesson 5

- Simple observational equivalence proof for `CbN`, `CbNeed`, `Neededness`.
- Simple observational equivalence proof for `VSC`, `LinearVSC`, `UsefulVSC`.
- Quantitative systems with **SPLIT MEASURES** for `VSC`, `LinearVSC`, `UsefulVSC` counting exactly multiplicative and exponential steps.
- Case `UsefulVSC` : **complex** operational semantics, **simple** typing system.
- Cases `VSC`, `LinearVSC`: **simple** operational semantics, **subtle** typing system.
- Can we capture the three operational semantics in a single system by means of different counters?

Concluding Remarks For Lesson 5

- Simple observational equivalence proof for `CbN`, `CbNeed`, `Neededness`.
- Simple observational equivalence proof for `VSC`, `LinearVSC`, `UsefulVSC`.
- Quantitative systems with **SPLIT MEASURES** for `VSC`, `LinearVSC`, `UsefulVSC` counting exactly multiplicative and exponential steps.
- Case `UsefulVSC` : **complex** operational semantics, **simple** typing system.
- Cases `VSC`, `LinearVSC`: **simple** operational semantics, **subtle** typing system.
- Can we capture the three operational semantics in a single system by means of different counters?
- Challenging cases:
  - **Effectful** models of computation (algebraic, continuations, . . .)
  - **Useful Call-by-Name** evaluation (and other interesting time cost models)
  - **Strong evaluation** (for proof assistants)

Takeaway Messages

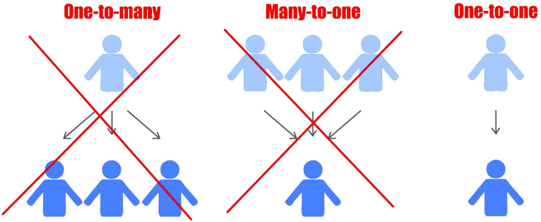
Lambda calculus is a microscope for understanding programming languages

Message 2

Linear logic is a resource-aware logic that tracks assumption usage

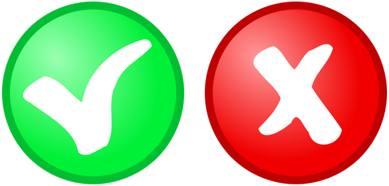
Message 3

(Intuitionistic) linear logic proof nets captured by the fine-grained pn term calculus



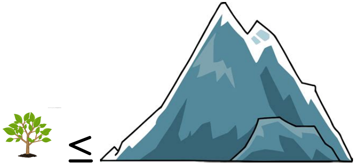
Message 4

Idempotent Intersection Types to Reason About Qualitative Properties of Programs



Message 5

Non-Idempotent Intersection Types to Reason About Quantitative Properties of Programs



Non-Idempotent Types with Upper Bounds

Non-Idempotent Types with Exact Measures

Non-Idempotent Types with Split Measures

A Subsuming Framework Inspired from Linear Logic to Capture Different Models of Computation

Intersection Types to Reason about Observational Equivalence

Call-by-Name

Call-by-Value

?

Call-by-Need

Linear Call-by-Value

Neededness

CbN/CbV
Useful Evaluation


THANK
YOU
FOR
YOUR
ATTENTION


Time for
Questions