Strong normalization y medidas decrecientes: demostraciones sintácticas de terminación en λ-cálculo tipado

Cristian Sottile

CONICET & Universidad de Buenos Aires

XXII Jornadas de Ciencias de la Computación

DCC, FCEIA, Universidad Nacional de Rosario

25 de octubre de 2024

Outline

Preliminares

- (Brevísima) ~~Introducción a~~ Proyección de λ-cálculo tipado
- La propiedad de *Strong normalization*
- La técnica de **reducibilidad**
- Medidas decrecientes
- El koan $\#26$

Novedades

- Propuesta
- Observación de Turing: grados de redexes y *weak normalization*
- Cálculo auxiliar $\lambda^{\mathtt{m}}$
- Medida $\mathcal{W}$: contando argumentos
- Medida $\mathcal{T}^{\mathtt{m}}$: contando (ciertos) términos alcanzables
- Medida $\mathcal{W}_{\cap}$: extensión a tipos intersección (idempotentes)

λ-cálculo

λ-cálculo

Estructura inductiva de los programas

$$t ::= x \mid \lambda x.t \mid tt$$

Reglas de cómputo

$$(\lambda x.t)s \to_\beta t[s/x]$$

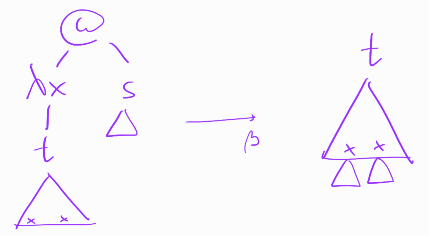
λ-cálculo

Estructura inductiva de los programas

$$t \ ::= \ x \mid \lambda x.t \mid tt$$

Reglas de cómputo

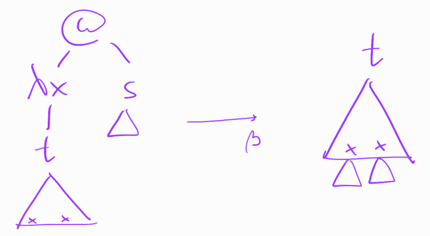
$$(\lambda x.t)s \ \to_\beta \ t[s/x]$$



λ-cálculo

Estructura inductiva de los programas

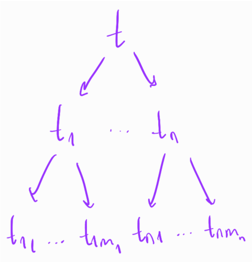
$$t \ ::= \ x \mid \lambda x.t \mid tt$$



Reglas de cómputo

$$(\lambda x.t)s \ \rightarrow_\beta \ t[s/x]$$

Cadenas de reducción



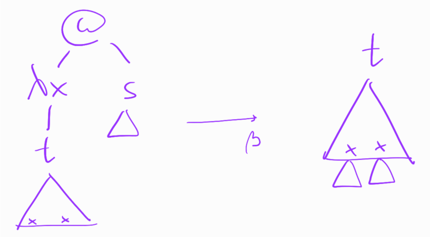
λ-cálculo

Estructura inductiva de los programas

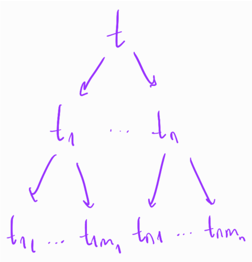
$$t \;::=\; x \mid \lambda x.t \mid tt$$

Reglas de cómputo

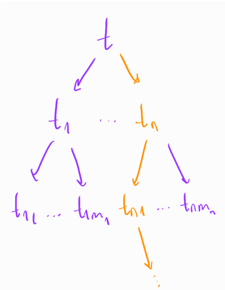
$$(\lambda x.t)s \;\to_\beta\; t[s/x]$$



Cadenas de reducción



Pueden ser infinitas



λ-cálculo tipado

λ-cálculo tipado

Motivación Lenguaje más seguro

λ-cálculo tipado

Motivación Lenguaje más seguro

Admitimos solo términos que tengan “sentido”

λ-cálculo tipado

Motivación Lenguaje más seguro

Admitimos solo términos que tengan "sentido" *e.g.* $f\,x$

λ-cálculo tipado

Motivación Lenguaje más seguro

Admitimos solo términos que tengan "sentido" *e.g.* $f\ x$

Tipos

$$A \ ::= \ \tau \mid A \to A$$

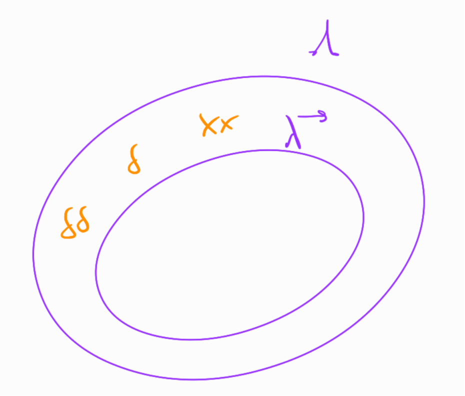
λ-cálculo tipado

Motivación Lenguaje más seguro

Admitimos solo términos que tengan "sentido" *e.g.* $f\, x$

Tipos

$$A \ ::= \ \tau \mid A \to A$$



La propiedad de terminación: *Strong normalization*

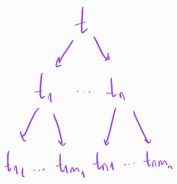
Definición

$\nexists t \to_\beta t_1 \to_\beta t_2 \to_\beta \cdots$

La propiedad de terminación: *Strong normalization*

Definición

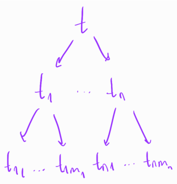
$$\nexists t \to_\beta t_1 \to_\beta t_2 \to_\beta \cdots$$

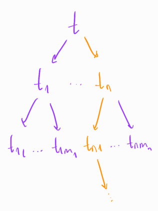


La propiedad de terminación: *Strong normalization*

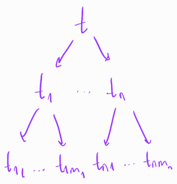
Definición

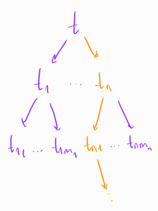
$$\nexists t \to_\beta t_1 \to_\beta t_2 \to_\beta \cdots$$





La propiedad de terminación: *Strong normalization*





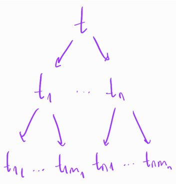
Definición

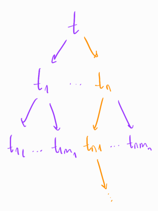
$$\nexists t \to_\beta t_1 \to_\beta t_2 \to_\beta \cdots$$

Motivación

- Obtener un resultado del cómputo
- Equivale a la simplificación de pruebas (vía Curry-Howard)
- Desarrollo de técnicas (*e.g.* tipos intersección, logical relations)
- Es interesante

La propiedad de terminación: *Strong normalization*





Definición

$$\nexists t \to_\beta t_1 \to_\beta t_2 \to_\beta \cdots$$

Motivación

- Obtener un resultado del cómputo
- Equivale a la simplificación de pruebas (vía Curry-Howard)
- Desarrollo de técnicas (*e.g.* tipos intersección, logical relations)
- Es interesante

Reducibilidad [Tait'67, Girard'72]: la técnica más usada

- Concisa
- Extensible a sistemas más complejos (*e.g.* System F, CoC)

Construyendo reducibilidad por prueba y error

Primer intento

Inducción en t

Construyendo reducibilidad por prueba y error

Primer intento

Inducción en t

Construyendo reducibilidad por prueba y error

Primer intento

Inducción en t

Caso aplicación ts

Construyendo reducibilidad por prueba y error

Primer intento

Inducción en t

Caso aplicación ts

- HI: t y s SN

Construyendo reducibilidad por prueba y error

Primer intento

Inducción en t

Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$

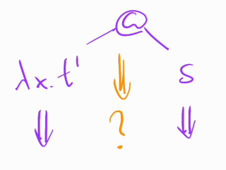
Construyendo reducibilidad por prueba y error

Primer intento

Inducción en t

Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$



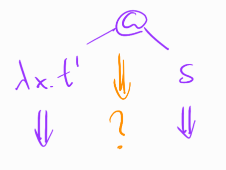
Construyendo reducibilidad por prueba y error

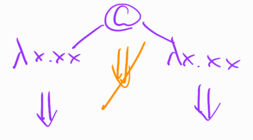
Primer intento

Inducción en t

Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$





Construyendo reducibilidad por prueba y error

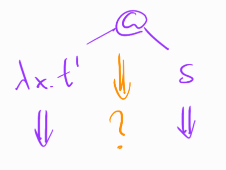
Primer intento

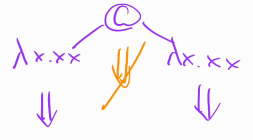
Inducción en t

Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$

Solución





Construyendo reducibilidad por prueba y error

Primer intento

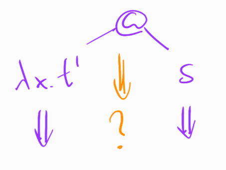
Inducción en t

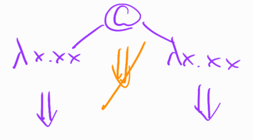
Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$

Solución

- observar qué deben cumplir los términos para ser SN





Construyendo reducibilidad por prueba y error

Primer intento

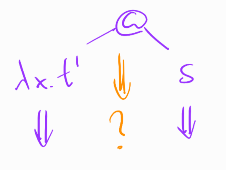
Inducción en t

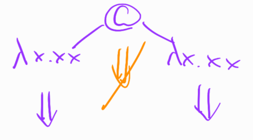
Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?





Construyendo reducibilidad por prueba y error

Primer intento

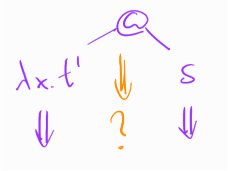
Inducción en t

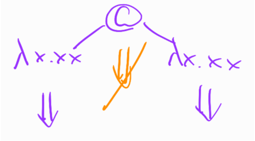
Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?





Dado un término t, debe cumplir:

Construyendo reducibilidad por prueba y error

Primer intento

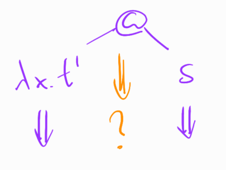
Inducción en t

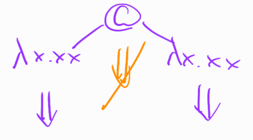
Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?





Dado un término t, debe cumplir:

1. ser SN "solo"

Construyendo reducibilidad por prueba y error

Primer intento

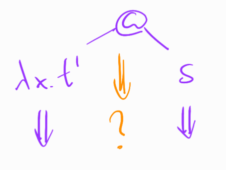
Inducción en t

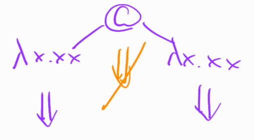
Caso aplicación ts

- HI: t y s SN
- **no alcanza** para ver SN si $t = \lambda x.t'$

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?





Dado un término t, debe cumplir:

1. ser SN "solo"
2. ser SN al "combinarse" (aplicarse)

Construyendo reducibilidad por prueba y error

Primer intento

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?

Candidatos de reducibilidad

Dado un término t, debe cumplir:

1. ser SN "solo"
2. ser SN al "combinarse" (aplicarse)

Construyendo reducibilidad por prueba y error

Primer intento

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?

Dado un término t, debe cumplir:

1. ser SN "solo"
2. ser SN al "combinarse" (aplicarse)

Candidatos de reducibilidad

- Los tipos indican cómo puede combinarse un término

Construyendo reducibilidad por prueba y error

Primer intento

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?

Dado un término t, debe cumplir:

1. ser SN "solo"
2. ser SN al "combinarse" (aplicarse)

Candidatos de reducibilidad

- Los tipos indican cómo puede combinarse un término
- Por inducción en el tipo, definimos los conjuntos de términos que cumplen: los **reducibles**

$$
\begin{aligned}
\mathit{RED}_\tau &= SN \\
\mathit{RED}_{A\to B} &= \{\ t \mid \forall s \in \mathit{RED}_A.\ ts \in \mathit{RED}_B\ \}
\end{aligned}
$$

Construyendo reducibilidad por prueba y error

Primer intento

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?

Dado un término t, debe cumplir:

1. ser SN "solo"
2. ser SN al "combinarse" (aplicarse)

Candidatos de reducibilidad

- Los tipos indican cómo puede combinarse un término
- Por inducción en el tipo, definimos los conjuntos de términos que cumplen: los **reducibles**

$$
\begin{aligned}
\mathit{RED}_\tau &= SN \\
\mathit{RED}_{A \to B} &= \{\ t \mid \forall s \in \mathit{RED}_A .\ ts \in \mathit{RED}_B\ \}
\end{aligned}
$$

- Propiedades de *RED*:
  - cerrado por reducción
  - """cerrado por antireducción"""

Construyendo reducibilidad por prueba y error

Primer intento

Solución

- observar qué deben cumplir los términos para ser SN
- ¿qué necesito de la HI?

Dado un término t, debe cumplir:

1. ser SN "solo"
2. ser SN al "combinarse" (aplicarse)

Candidatos de reducibilidad

- Los tipos indican cómo puede combinarse un término
- Por inducción en el tipo, definimos los conjuntos de términos que cumplen: los **reducibles**

$$\mathit{RED}_\tau = SN$$
$$\mathit{RED}_{A\to B} = \{\ t \mid \forall s \in \mathit{RED}_A.\ ts \in \mathit{RED}_B\ \}$$

- Propiedades de *RED*:
  - cerrado por reducción
  - """cerrado por antireducción"""
- Vemos que todos los términos son reducibles: $t : A \implies \mathit{RED}_A(t)$

Reducibilidad [Tait'67, Girard'72]

Segundo Intento

Reducibilidad [Tait'67, Girard'72]

Segundo Intento

Quería ver que todos los términos son SN

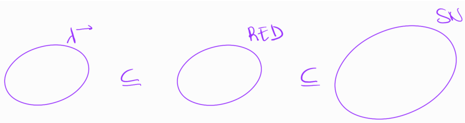
Quiero ver que todos los términos son *RED*

Reducibilidad [Tait'67, Girard'72]

Segundo Intento

Quería ver que todos los términos son SN

Quiero ver que todos los términos son *RED*

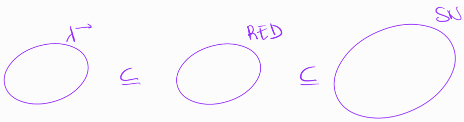


Reducibilidad [Tait'67, Girard'72]

Segundo Intento

Quería ver que todos los términos son SN

Quiero ver que todos los términos son *RED*



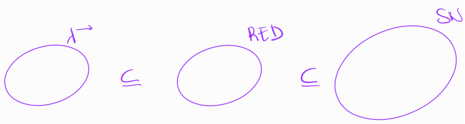
Inducción en t

Reducibilidad [Tait'67, Girard'72]

Segundo Intento

Quería ver que todos los términos son SN

Quiero ver que todos los términos son *RED*



Inducción en t

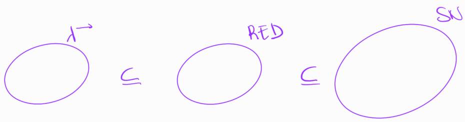
- **Caso aplicación ts** HI: t y s *RED* (SN solos y combinados)

Reducibilidad [Tait'67, Girard'72]

Segundo Intento

Quería ver que todos los términos son SN

Quiero ver que todos los términos son *RED*



Inducción en t

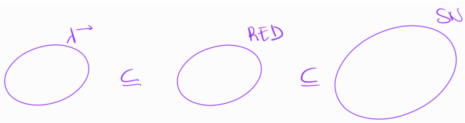
- **Caso aplicación ts** HI: t y s *RED* (SN solos y combinados)
- **Caso abstracción $\lambda x.t$** tengo que probar que es *RED* al aplicarse

Reducibilidad [Tait'67, Girard'72]

Segundo Intento

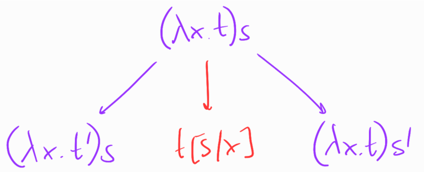
Quería ver que todos los términos son SN

Quiero ver que todos los términos son *RED*



Inducción en t

- **Caso aplicación ts** HI: t y s *RED* (SN solos y combinados)
- **Caso abstracción $\lambda x.t$** tengo que probar que es *RED* al aplicarse
  - HI t *RED*
  - **necesito $t[s/x]$ *RED*** para cualquier s

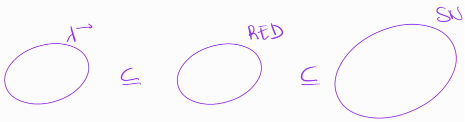


Reducibilidad [Tait'67, Girard'72]

Segundo Intento

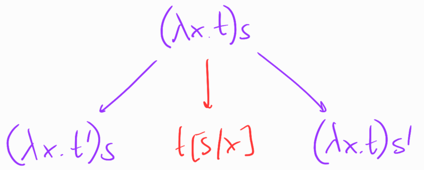
Quería ver que todos los términos son SN

Quiero ver que todos los términos son *RED*



Inducción en t

- **Caso aplicación ts** HI: t y s *RED* (SN solos y combinados)
- **Caso abstracción $\lambda x.t$** tengo que probar que es *RED* al aplicarse
    - HI t *RED*
    - **necesito $t[s/x]$ *RED*** para cualquier s



Solución

- fortalezco HI
- pruebo lema más general: todo **cierre reducible** θ de t es *RED*

$$t : A \implies RED_A(\theta t)$$

Reducibilidad [Tait'67, Girard'72]

Quería ver que todos los términos son SN

Quería ver que todos los términos son *RED*

Quiero ver que todos los cierres reducibles de términos son *RED*

Reducibilidad [Tait'67, Girard'72]

Quería ver que todos los términos son SN

Quería ver que todos los términos son *RED*

Quiero ver que todos los cierres reducibles de términos son *RED*

Inducción en t

- **Caso aplicación ts** HI: θt y θs *RED*
- **Caso abstracción $\lambda x.t$** tengo que probar que es *RED* al aplicarse

Reducibilidad [Tait'67, Girard'72]

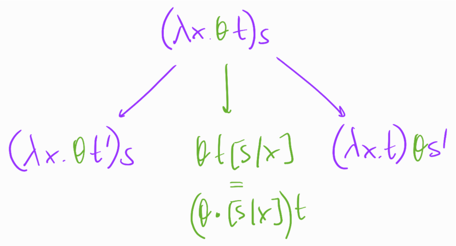
Quería ver que todos los términos son SN

Quería ver que todos los términos son *RED*

Quiero ver que todos los cierres reducibles de términos son *RED*

Inducción en t

- **Caso aplicación ts** HI: θt y θs *RED*
- **Caso abstracción $\lambda x.t$** tengo que probar que es *RED* al aplicarse
  - **HI θt *RED***
  - por inducción en $|\theta t| + |s|$, todos los reductos en un paso son *RED*



Reducibilidad [Tait'67, Girard'72]

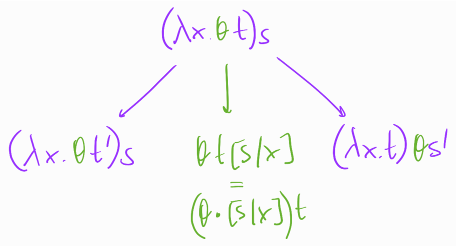
Quería ver que todos los términos son SN

Quería ver que todos los términos son *RED*

Quiero ver que todos los cierres reducibles de términos son *RED*

Inducción en t

- **Caso aplicación ts** HI: θt y θs *RED*
- **Caso abstracción $\lambda x.t$** tengo que probar que es *RED* al aplicarse
  - **HI θt *RED***
  - por inducción en $|\theta t| + |s|$, todos los reductos en un paso son *RED*



Teorema $t : A \implies t \in SN$

Demostración El cierre identidad es reducible, $\therefore\ t : A \implies t \in SN$

Profundizando en la técnica de reducibilidad

Gallier (en *Proving properties of typed λ-terms using realizability, covers, and sheaves*)

This paper provides some answers to the above questions. But before explaining our results, we would like to explain our motivations and our point of view a little more. Reducibility proofs are seductive and thrilling, but also elusive. Following these proofs step-by-step, we see that they "work" (when they are not wrong!), but I claim that most of us would still admit that they are not sure *why* these proofs work! The situation is somewhat comparable to driving a Ferrari (I suppose): the feeling of power is tremendous, but what exactly is under the hood? What kind of carburator, what kind of valve mechanism, gives such power and flexibility?

Profundizando en la técnica de reducibilidad

Gallier (en *Proving properties of typed λ-terms using realizability, covers, and sheaves*)

This paper provides some answers to the above questions. But before explaining our results, we would like to explain our motivations and our point of view a little more. Reducibility proofs are seductive and thrilling, but also elusive. Following these proofs step-by-step, we see that they "work" (when they are not wrong!), but I claim that most of us would still admit that they are not sure *why* these proofs work! The situation is somewhat comparable to driving a Ferrari (I suppose): the feeling of power is tremendous, but what exactly is under the hood? What kind of carburator, what kind of valve mechanism, gives such power and flexibility?

van de Pol (en *Two different strong normalization proofs?*)

In the literature, these two methods are often put in contrast ([Gan80, § 6.3] and [Gir87, annex 2.C.1]). The proof using functionals seems to be more transparent and economizes on proof theoretical complexity. On the other hand, seeing the two proofs one gets the feeling that "somehow, the same thing is going on". Indeed De Vrijer [dV87, § 0.1] remarks that a proof using strong computability can be seen as abstracting from concrete information in the functionals that is not strictly needed in a termination proof, but which provides for an estimate of reduction lengths.

Medidas decrecientes [Gandy'80, de Vrijer'87]

Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

Asignación tal que

$\# : \Lambda \to \mathit{WFO} \qquad M \to_\beta N \implies \#(M) > \#(N)$

Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

Asignación $\# : \Lambda \to \mathit{WFO}$ tal que $M \to_\beta N \implies \#(M) > \#(N)$

Corolario

$\nexists\, M_1 \to_\beta M_2 \to_\beta \cdots$

Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

Asignación tal que

$$\# : \Lambda \to \mathit{WFO} \qquad M \to_\beta N \implies \#(M) > \#(N)$$

Medidas de Gandy y de Vrijer

Corolario

$$\nexists\, M_1 \to_\beta M_2 \to_\beta \cdots$$

Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

Asignación tal que

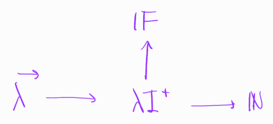
$\# : \Lambda \to \mathit{WFO} \qquad M \to_\beta N \implies \#(M) > \#(N)$

Corolario

$\nexists\, M_1 \to_\beta M_2 \to_\beta \cdots$

Medidas de Gandy y de Vrijer

Basadas en interpretaciones de $\boldsymbol{\lambda}^{\to}$ a **increasing functionals**



Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

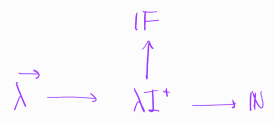
Asignación $\# : \Lambda \to \mathit{WFO}$ tal que $M \to_\beta N \implies \#(M) > \#(N)$

Corolario

$\nexists\ M_1 \to_\beta M_2 \to_\beta \cdots$

Medidas de Gandy y de Vrijer

Basadas en interpretaciones de $\boldsymbol{\lambda}^{\rightarrow}$ a **increasing functionals**



1. Definen el conjunto de los *IF increasing functionals* (funciones de alto orden sobre naturales crecientes punto a punto)

Medidas decrecientes [Gandy'80, de Vrijer'87]

Corolario

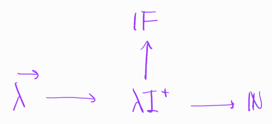
$\nexists\, M_1 \to_\beta M_2 \to_\beta \cdots$

Definición

Asignación tal que

$\# : \Lambda \to \mathit{WFO} \qquad M \to_\beta N \implies \#(M) > \#(N)$

Medidas de Gandy y de Vrijer



Basadas en interpretaciones de $\lambda^{\to}$ a **increasing functionals**

1. Definen el conjunto de los *IF increasing functionals* (funciones de alto orden sobre naturales crecientes punto a punto)
2. Definen operaciones sobre los *IF*

Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

Asignación tal que

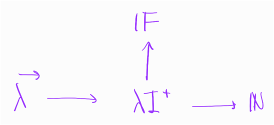
$\# : \Lambda \to \mathit{WFO}$ $\quad M \to_\beta N \implies \#(M) > \#(N)$

Corolario

$\nexists\, M_1 \to_\beta M_2 \to_\beta \cdots$

Medidas de Gandy y de Vrijer

Basadas en interpretaciones de $\lambda^\to$ a **increasing functionals**



1. Definen el conjunto de los *IF increasing functionals*
   (funciones de alto orden sobre naturales crecientes punto a punto)
2. Definen operaciones sobre los *IF*
3. Definen una proyección de *IF* a $\mathbb{N}$

Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

Asignación tal que

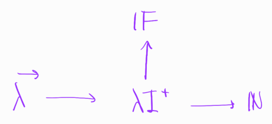
$\# : \Lambda \to \mathit{WFO}$ $M \to_\beta N \implies \#(M) > \#(N)$

Corolario

$\nexists\ M_1 \to_\beta M_2 \to_\beta \cdots$

Medidas de Gandy y de Vrijer

Basadas en interpretaciones de $\lambda^{\to}$ a **increasing functionals**



1. Definen el conjunto de los *IF increasing functionals* (funciones de alto orden sobre naturales crecientes punto a punto)
2. Definen operaciones sobre los *IF*
3. Definen una proyección de *IF* a $\mathbb{N}$
4. Definen la asignación de términos en *IF*

Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

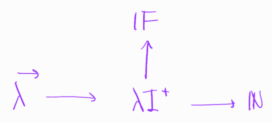
Asignación $\# : \Lambda \to \mathit{WFO}$ tal que $M \to_\beta N \implies \#(M) > \#(N)$

Corolario

$\nexists\ M_1 \to_\beta M_2 \to_\beta \cdots$

Medidas de Gandy y de Vrijer

Basadas en interpretaciones de $\boldsymbol{\lambda^\to}$ a **increasing functionals**



1. Definen el conjunto de los *IF increasing functionals* (funciones de alto orden sobre naturales crecientes punto a punto)
2. Definen operaciones sobre los *IF*
3. Definen una proyección de *IF* a $\mathbb{N}$
4. Definen la asignación de términos en *IF*
5. Prueban que la proyección de la asignación decrece con cada reducción

Medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

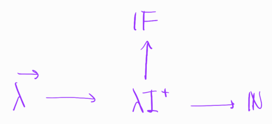
Asignación $\# : \Lambda \to \mathit{WFO}$ tal que $M \to_\beta N \implies \#(M) > \#(N)$

Corolario

$\nexists\, M_1 \to_\beta M_2 \to_\beta \cdots$

Medidas de Gandy y de Vrijer

Basadas en interpretaciones de $\lambda^{\to}$ a **increasing functionals**



1. Definen el conjunto de los *IF increasing functionals* (funciones de alto orden sobre naturales crecientes punto a punto)
2. Definen operaciones sobre los *IF*
3. Definen una proyección de *IF* a $\mathbb{N}$
4. Definen la asignación de términos en *IF*
5. Prueban que la proyección de la asignación decrece con cada reducción

Reducción de SN a WN [Nederpelt'73, Klop'80]

Reducción mediante λI + Prueba de WN

Why?

Why decreasing measures?

- insight
- intuition
- metrics

Why?

Why decreasing measures?

- insight
- intuition
- metrics

The koan #26

- Posed by Gödel
- Submitted by Barendregt
- Find an “easy” mapping from $\lambda^{\rightarrow}$ to ordinals

Why?

Why decreasing measures?

- insight
- intuition
- metrics

The koan #26

- Posed by Gödel
- Submitted by Barendregt
- Find an "easy" mapping from $\lambda^{\rightarrow}$ to ordinals

Why "syntactic"

- sort of convention
- soft classification of SN proofs
- maybe **abstract** vs **concrete** would be better?
- **external** vs **internal**?
- we stick to the convention

semantic	**syntactic**
reducibility (RC)	decreasing measures (DM)
	reduction of SN to WN (NK)

syntactic = "internal" analysis over the **structure of terms** or the **rewriting relation**

Our work

[Barenbaum & Sottile FSCD'23]

- **An auxiliar calculus $\lambda^{\mathtt{m}}$** to manipulate (non-)erasure through memories
- **A simple measure $\mathcal{W}$** based on counting memories
- **A complex measure $\mathcal{T}^{\mathtt{m}}$** generalizing Turing's WN one

Our work

[Barenbaum & Sottile FSCD'23]

- **An auxiliar calculus $\lambda^{\mathtt{m}}$** to manipulate (non-)erasure through memories
- **A simple measure $\mathcal{W}$** based on counting memories
- **A complex measure $\mathcal{T}^{\mathtt{m}}$** generalizing Turing's WN one

[Barenbaum, Ronchi della Rocca, Sottile]

- A presentation of **idempotent intersection types a la Church**
- An adaptation of $\mathcal{W}$ to idempotent intersection types, $\mathcal{W}_{\cap}$

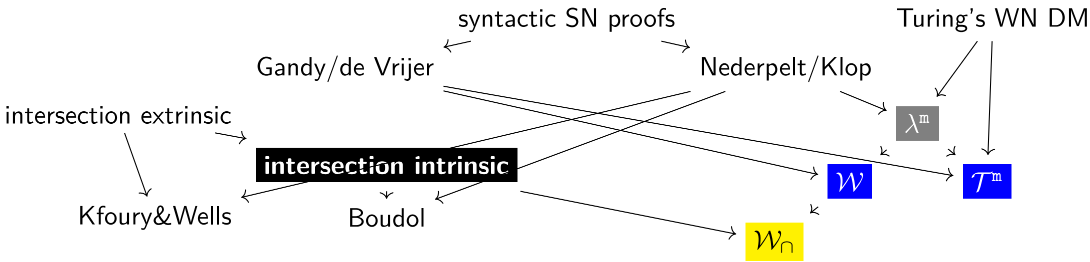
Our work

[Barenbaum & Sottile FSCD'23]

- **An auxiliar calculus $\lambda^{\mathtt{m}}$** to manipulate (non-)erasure through memories
- **A simple measure $\mathcal{W}$** based on counting memories
- **A complex measure $\mathcal{T}^{\mathtt{m}}$** generalizing Turing's WN one

[Barenbaum, Ronchi della Rocca, Sottile]

- A presentation of **idempotent intersection types a la Church**
- An adaptation of $\mathcal{W}$ to idempotent intersection types, $\mathcal{W}_{\cap}$



The auxiliar non-erasing $\lambda^{\mathtt{m}}$-calculus

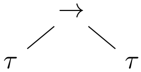
Turing's measure: preliminary definitions

Height of a type

Length of longest path as tree

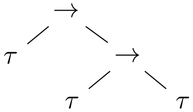
Examples

$\tau \to \tau$



1

$\tau \to \tau \to \tau$



2

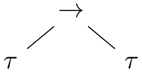
Turing's measure: preliminary definitions

Height of a type

Length of longest path as tree

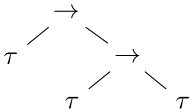
Examples

$\tau \to \tau$



1

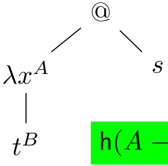
$\tau \to \tau \to \tau$



2

Degree of a redex

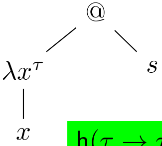
Height of its lambda



$\mathsf{h}(A \to B)$

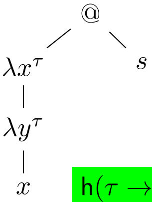
Examples

$(\lambda x^\tau.x)s$



$\mathsf{h}(\tau \to \tau) = 1$

$(\lambda x^\tau.\lambda y^\tau.x)s$



$\mathsf{h}(\tau \to \tau \to \tau) = 2$

Turing's measure: Weak Normalization

Map terms $\mapsto$ multiset of the redex degrees

$\mathcal{T}(M) = [\ d \mid R$ is a redex of degree d in $M\ ]$

Turing's measure: Weak Normalization

Map terms $\mapsto$ multiset of the redex degrees

$\mathcal{T}(M) = [\ d \mid R$ is a redex of degree d in $M\ ]$

Example

$$\mathcal{T}(\underbrace{(\lambda x^\tau.\lambda y^\tau.x)\,\underbrace{(\lambda x^\tau.x)s}_{1}}_{2}) = [2,1]$$

Two crucial observations [Turing, 1940s]

1. a redex cannot create redexes of greater or equal degree
2. a redex can copy redexes of any degree

Turing's measure: Weak Normalization

Map terms $\mapsto$ multiset of the redex degrees

$\mathcal{T}(M) = [\ d \mid R$ is a redex of degree d in $M\ ]$

Example

$$\mathcal{T}(\underbrace{(\lambda x^{\tau}.\lambda y^{\tau}.x)\,\underbrace{(\lambda x^{\tau}.x)s}_{1}}_{2}) = [2, 1]$$

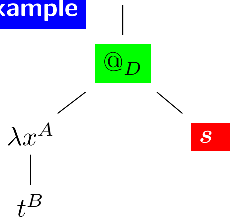
Two crucial observations [Turing, 1940s]

1. a redex cannot create redexes of greater or equal degree
2. a redex can copy redexes of any degree

WN: choosing the redex to contract

- has the greatest degree
- rightmost occurrence of that degree

Example



Contracting rightmost greatest $@_D$

- **cannot** create redexes $\geq D$
- **cannot** copy redexes $\geq D$

Hence

- one less D redex

The auxiliar non-erasing $\lambda^{\mathtt{m}}-$calculus

Definition

$$t \; ::= \; x \mid \lambda x.t \mid t\,t \mid t\{t\} \qquad (\lambda x.t)s \to_m t[s/x]\{s\}$$

The auxiliar non-erasing $\lambda^{\mathtt{m}}-$calculus

Definition

$$t \ ::= \ x \mid \lambda x.t \mid t\,t \mid t\{t\} \qquad (\lambda x.t)s \to_m t[s/x]\{s\}$$

Properties

- WCR
- WN
- SR

The auxiliar non-erasing $\lambda^{\mathtt{m}}-$calculus

Definition

$t ::= x \mid \lambda x.t \mid t\,t \mid t\{t\}$ $\quad (\lambda x.t)s \to_m t[s/x]\{s\}$

Properties

- WCR
- WN
- SR

Why not to erase?

- Nederpelt-Klop's:
  INC WCR WN $\Rightarrow$ *DEC*

The auxiliar non-erasing $\lambda^{\mathtt{m}}-$calculus

Definition

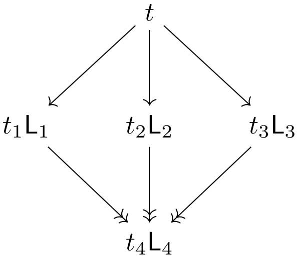
$$t \ ::= \ x \mid \lambda x.t \mid t\,t \mid t\{t\} \qquad (\lambda x.t)s \to_m t[s/x]\{s\}$$

Properties

- WCR
- WN
- SR

Why not to erase?

- Nederpelt-Klop's:
  INC WCR WN $\Rightarrow$ *DEC*
- **Retain information**



The auxiliar non-erasing $\lambda^{\mathtt{m}}-$calculus

Definition

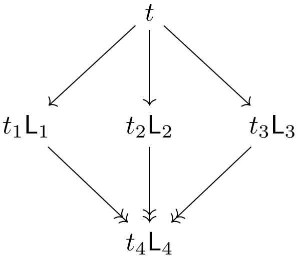
$t \ ::= \ x \mid \lambda x.t \mid t\,t \mid t\{t\}$ $\qquad (\lambda x.t)s \rightarrow_m t[s/x]\{s\}$

Properties

- WCR
- WN
- SR

Why not to erase?

- Nederpelt-Klop's: *INC WCR WN* $\Rightarrow$ *DEC*
- **Retain information**



Operations

- **weight** of a term:
  $\mathsf{w}(t)$ = amount of memories
  e.g. $\mathsf{w}(x\{y\{z\}\}\{w\}) = 3$
- **simplification** of a term:
  $\mathsf{S}_D(t)$ = "bottom-up" contraction of all D redexes
  $\mathsf{S}_*(t) = \mathsf{S}_1(\ldots \mathsf{S}_{\mathsf{maxdeg}}(t)\ldots)$

The auxiliar non-erasing $\lambda^{\mathtt{m}}$−calculus

Definition

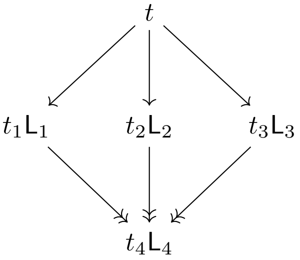
$$t \ ::= \ x \mid \lambda x.t \mid t\,t \mid t\{t\} \qquad (\lambda x.t)s \to_m t[s/x]\{s\}$$

Properties

- WCR
- WN
- SR

Why not to erase?

- Nederpelt-Klop's: *INC WCR WN* $\Rightarrow$ *DEC*
- **Retain information**



Operations

- **weight** of a term:
  w(t) = amount of memories
  e.g. w$(x\{y\{z\}\}\{w\}) = 3$
- **simplification** of a term:
  S$_D(t)$ = "bottom-up" contraction of all D redexes
  S$_*(t)$ = S$_1(\dots$ S$_{\mathsf{maxdeg}}(t)\dots)$

Properties

- Reduction arrives at simplification $\quad t \to_m^* \mathsf{S}_*(t)$
- Simplification is normal form $\quad \mathsf{S}_*(t) = \mathtt{nf}(t)$

$\mathcal{W}$: counting memories

Measure $\mathcal{W}$

Recall $(\lambda x.t)s \to_m t[s/x]\{s\}$ $\mathsf{w}(t)$ = amount of memories

Measure $\mathcal{W}$

Recall $(\lambda x.t)s \to_m t[s/x]\{s\}$ $\qquad$ $\mathsf{w}(t)$ = amount of memories

Idea $\qquad t \to s \quad \Longrightarrow \quad \texttt{nf}(t)$ has more memories than $\texttt{nf}(s)$

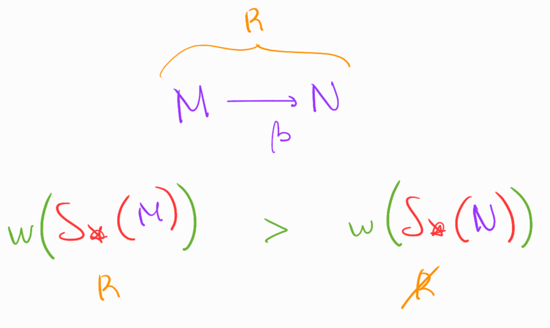
Measure $\mathcal{W}$

Recall $(\lambda x.t)s \to_m t[s/x]\{s\}$

$\mathrm{w}(t)$ = amount of memories

Idea

$t \to s \implies$ $\mathtt{nf}(t)$ has more memories than $\mathtt{nf}(s)$



Definition

$$\mathcal{W}(M) = \mathrm{w}(\mathsf{S}_*(M))$$

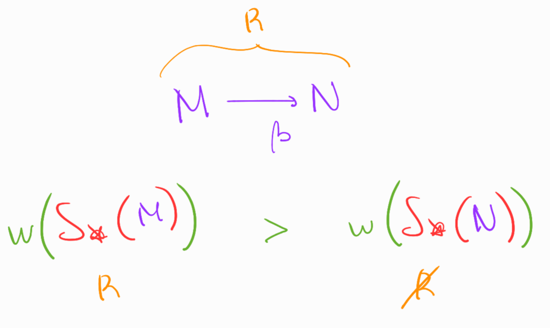
Measure $\mathcal{W}$

Recall $(\lambda x.t)s \to_m t[s/x]\{s\}$

$\mathsf{w}(t)$ = amount of memories

Idea

$t \to s \implies$ $\mathtt{nf}(t)$ has more memories than $\mathtt{nf}(s)$



Definition

$$\mathcal{W}(M) = \mathsf{w}(\mathsf{S}_*(M))$$

Theorem

$$M \to_\beta N \implies \mathcal{W}(M) > \mathcal{W}(N)$$

$\mathcal{T}^{\mathtt{m}}$: generalizing Turing's WN measure

Turing's measure: adaptation to SN

Proposal generalize the measure so that it decreases by contracting *any* redex

Turing's measure: adaptation to SN

Proposal generalize the measure so that it decreases by contracting *any* redex

Problems

- $(>)$ A redex copies redexes of greater degree
- $(=)$ A redex copies redexes of same degree

Turing's measure: adaptation to SN

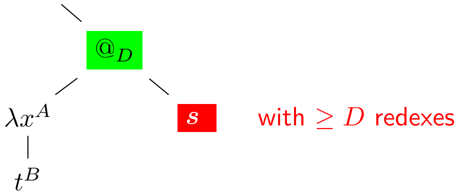
Proposal generalize the measure so that it decreases by contracting *any* redex

Problems

($>$) A redex copies redexes of greater degree

($=$) A redex copies redexes of same degree

For instance



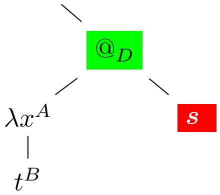
Turing's measure: adaptation to SN

Proposal generalize the measure so that it decreases by contracting *any* redex

Problems

- $(>)$ A redex copies redexes of greater degree
- $(=)$ A redex copies redexes of same degree

For instance



with $\geq D$ redexes

Idea

$i)$ generalize $\mathcal{T}$ to a **family of measures $\mathcal{T}'_D$ indexed by a degree** $D \in \mathbb{N}$

$\mathcal{T}'_2(M) = [2,1]$ and $\mathcal{T}'_1(M) = [1]$

Turing's measure: adaptation to SN

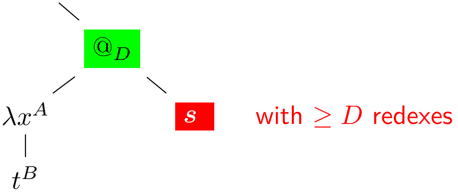
Proposal generalize the measure so that it decreases by contracting *any* redex

Problems

($>$) A redex copies redexes of greater degree

($=$) A redex copies redexes of same degree

For instance



Idea

$i)$ generalize $\mathcal{T}$ to a **family of measures $\mathcal{T}'_D$ indexed by a degree** $D \in \mathbb{N}$

$$\mathcal{T}'_2(M) = [2, 1] \qquad \text{and} \qquad \mathcal{T}'_1(M) = [1]$$

$ii)$ **associate extra information among with redex degrees**

e.g. consider smaller redexes' info (through the same measure)

$$\mathcal{T}'_2(M) = [\ (2, \mathcal{T}'_1(M)),\ (1, [])\ ] \qquad \mathcal{T}'_1(M) = [\ (1, [])\ ]$$

Measure $\mathcal{T}^{\mathtt{m}}$

More information...

$$\mathcal{T}'_2(M) = [\ (2, ?),\ (1, ?)\ ]$$

Measure $\mathcal{T}^{\mathtt{m}}$

More information...

$$\mathcal{T}_2'(M) = [\ (2, \boxed{?}),\ (1, \boxed{?})\ ]$$

Idea

Development of degree D

reduction involving only redexes D

Measure $\mathcal{T}^{\mathtt{m}}$

More information...

$$\mathcal{T}_2'(M) = [\ (2, ?),\ (1, ?)\ ]$$

Idea

Development of degree D
reduction involving only redexes D

All developments of degree D
paths of the complete D-reduction graph from t

Measure $\mathcal{T}^{\mathtt{m}}$

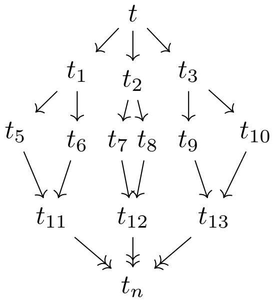
More information...

$$\mathcal{T}_2'(M) = [\ (2, \boxed{?}),\ (1, \boxed{?})\ ]$$

Idea

Development of degree D
reduction involving only redexes D

All developments of degree D
paths of the complete D-reduction graph from t



Measure $\mathcal{T}^{\mathtt{m}}$

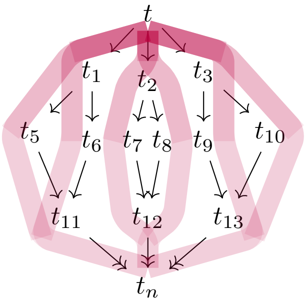
More information...

$$\mathcal{T}_2'(M) = [\ (2, ?),\ (1, ?)\ ]$$

Idea

Development of degree D
reduction involving only redexes D

All developments of degree D
paths of the complete D-reduction graph from t



Measure $\mathcal{T}^{\mathtt{m}}$

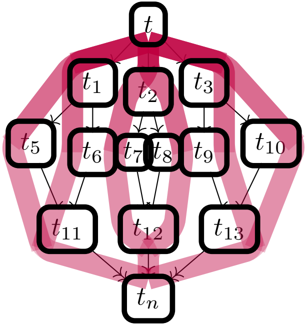
More information...

$$\mathcal{T}_2'(M) = [\ (2, \boxed{?}),\ (1, \boxed{?})\ ]$$

Idea

Development of degree D
reduction involving only redexes D

All developments of degree D
paths of the complete D-reduction graph from t



Measure $\mathcal{T}^{\mathtt{m}}$

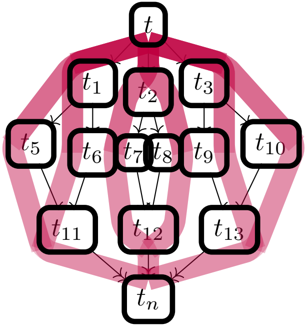
More information...

$$\mathcal{T}_2'(M) = [\ (2, ?),\ (1, ?)\ ]$$

Idea

Development of degree D
reduction involving only redexes D

All developments of degree D
paths of the complete D-reduction graph from t



Extra information

Multiset $[\ \mathcal{T}^m_{D-1}(t') \mid t' \text{ is } D\text{-reachable from } t\ ]$

$$[\ \mathcal{T}^m_{D-1}(t), \mathcal{T}^m_{D-1}(t_1),\ \ldots, \mathcal{T}^m_{D-1}(t_n)\ ]$$

Measure $\mathcal{T}^{\mathtt{m}}$

More information...

$$\mathcal{T}_2'(M) = [\ (2, ?),\ (1, ?)\ ]$$

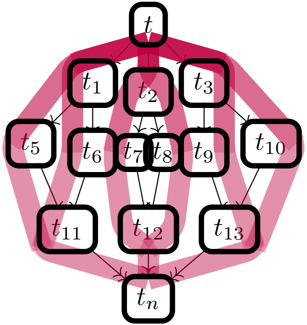
Idea

Development of degree D

reduction involving only redexes D

All developments of degree D

paths of the complete D-reduction graph from t



Extra information

Multiset $[\ \mathcal{T}^m_{D-1}(t') \mid t'$ is D-reachable from $t\ ]$

$$[\ \mathcal{T}^m_{D-1}(t), \mathcal{T}^m_{D-1}(t_1),\ \ldots, \mathcal{T}^m_{D-1}(t_n)\ ]$$

Definition

$$\mathcal{T}^m_D(t) = [\ (i, \mathcal{V}^m_i(t)) \mid R \text{ is a redex of degree } i \leq D \text{ in } t\ ]$$

$$\mathcal{V}^m_D(t) = [\ \mathcal{T}^m_{D-1}(t') \mid \rho : t \xrightarrow{D}{}^*_m t'\ ]$$

Measure $\mathcal{T}^{\mathtt{m}}$

More information...

$$\mathcal{T}'_2(M) = [\ (2, ?),\ (1, ?)\ ]$$

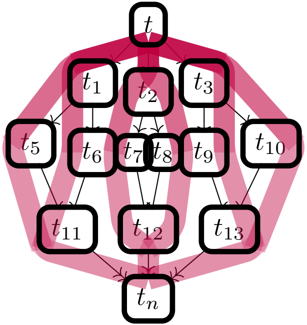
Idea

Development of degree D

reduction involving only redexes D

All developments of degree D

paths of the complete D-reduction graph from t



Extra information

Multiset $[\ \mathcal{T}^m_{D-1}(t') \mid t'$ is D-reachable from $t\ ]$

$$[\ \mathcal{T}^m_{D-1}(t), \mathcal{T}^m_{D-1}(t_1),\ \ldots, \mathcal{T}^m_{D-1}(t_n)\ ]$$

Definition

$$\mathcal{T}^m_D(t) = [\ (i, \mathcal{V}^m_i(t)) \mid R \text{ is a redex of degree } i \leq D \text{ in } t\ ]$$

$$\mathcal{V}^m_D(t) = [\ \mathcal{T}^m_{D-1}(t') \mid \rho : t \xrightarrow{D}{}^{*}_{m} t'\ ]$$

Theorem

$$M \to_\beta N \implies \mathcal{T}^m(M) > \mathcal{T}^m(N)$$

$\mathcal{W}_\cap$: extending $\mathcal{W}$ to Idempotent Intersection Types

Motivation

Existing decreasing measures

Motivation

Existing decreasing measures

[Kfoury & Wells'95]

- **Domain of DM:** multiset of numbers
- **Methodology:** WN $\implies$ SN + DM proving WN (indirect)
- **Auxiliary calculus:** a la Curry

Motivation

Existing decreasing measures

[Kfoury & Wells'95]

- **Domain of DM:** multiset of numbers
- **Methodology:** WN $\implies$ SN + DM proving WN (indirect)
- **Auxiliary calculus:** a la Curry

[Boudol'03]

- **Domain of DM:** pair of numbers
- **Methodology:** WN $\implies$ SN + DM proving WN (indirect)
- **Auxiliary calculus:** a la Church, ad hoc

Motivation

Existing decreasing measures

[Kfoury & Wells'95]

- **Domain of DM:** multiset of numbers
- **Methodology:** WN $\implies$ SN + DM proving WN (indirect)
- **Auxiliary calculus:** a la Curry

[Boudol'03]

- **Domain of DM:** pair of numbers
- **Methodology:** WN $\implies$ SN + DM proving WN (indirect)
- **Auxiliary calculus:** a la Church, ad hoc

Our proposal **Barenbaum, Ronchi della Rocca & Sottile (WIP)**

- **Domain of DM:** **number**
- **Methodology:** DM proving SN **(direct)**
- **Auxiliary calculus:** a la Church, **correspondent** of a la Curry calculus

Idempotent Intersection Types (a la Curry)

Key idea

- Variables can have multiple types *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x : \tau$
- Hence a term can have truly different (non-unifiable) types

Very powerful at charaterizing properties

Idempotent Intersection Types (a la Curry)

Key idea

- Variables can have multiple types *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x : \tau$
- Hence a term can have truly different (non-unifiable) types

Very powerful at charaterizing properties

***The* typing rule**

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}}\ e - multi$$

Idempotent Intersection Types (a la Curry)

Key idea

- Variables can have multiple types *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x : \tau$
- Hence a term can have truly different (non-unifiable) types

Very powerful at charaterizing properties

***The* typing rule**

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}} \; e - multi$$

Example

Let

$$A = \tau \to \tau$$

Idempotent Intersection Types (a la Curry)

Key idea

- Variables can have multiple types *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x : \tau$
- Hence a term can have truly different (non-unifiable) types

Very powerful at charaterizing properties

***The* typing rule**

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}} \; e-multi$$

Example

Let

$$A = \tau \to \tau \qquad x : \{\boldsymbol{A \to A}, \boldsymbol{A}\} \vdash \boldsymbol{xx} : A$$

Idempotent Intersection Types (a la Curry)

Key idea

- Variables can have multiple types *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x : \tau$
- Hence a term can have truly different (non-unifiable) types

Very powerful at charaterizing properties

***The* typing rule**

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}} \; e - multi$$

Example

Let

$$A = \tau \to \tau \qquad x : \{\boldsymbol{A} \to \boldsymbol{A}, \boldsymbol{A}\} \vdash \boldsymbol{xx} : A \qquad \vdash \lambda x.x : \{\boldsymbol{A} \to \boldsymbol{A}, \boldsymbol{A}\}$$

Idempotent Intersection Types (a la Curry)

Key idea

- Variables can have multiple types — *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x : \tau$
- Hence a term can have truly different (non-unifiable) types

Very powerful at charaterizing properties

***The* typing rule**

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}} \; e - multi$$

Example

Let

$$A = \tau \to \tau \qquad x : \{\boldsymbol{A} \to \boldsymbol{A}, \boldsymbol{A}\} \vdash \boldsymbol{xx} : A \qquad \vdash \lambda x.x : \{\boldsymbol{A} \to \boldsymbol{A}, \boldsymbol{A}\}$$

Then

$$\vdash (\lambda x.\boldsymbol{xx})(\lambda x.x) : A$$

Idempotent Intersection Types (a la Curry)

Key idea

- Variables can have multiple types *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x : \tau$
- Hence a term can have truly different (non-unifiable) types

Very powerful at charaterizing properties

***The* typing rule**

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}} \; e - multi$$

Example

Let

$$A = \tau \to \tau \qquad x : \{\boldsymbol{A \to A}, \boldsymbol{A}\} \vdash \boldsymbol{xx} : A \qquad \vdash \lambda x.x : \{\boldsymbol{A \to A}, \boldsymbol{A}\}$$

Then

$$\vdash (\lambda x.\boldsymbol{xx})(\lambda x.x) : A \qquad (\lambda x.\boldsymbol{xx})(\lambda x.x) \to_\beta \boldsymbol{(\lambda x.x)(\lambda x.x)}$$

Idempotent Intersection Types a la Church

Key idea

- Variables can have multiple types **defined a priori** *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x^{\tau} : \tau$
- Hence a term **modulo erasure** can have truly different (non-unifiable) types

Idempotent Intersection Types a la Church

Key idea

- Variables can have multiple types **defined a priori** *e.g.* $x : \{\tau, \tau \to \tau\} \vdash x^{\tau} : \tau$
- Hence a term **modulo erasure** can have truly different (non-unifiable) types

Motivation

- $\lambda^{\mathtt{m}}$ is a la Church (easier syntactic analysis)
- abscence of standard correspondent Church version of Curry system

Idempotent Intersection Types a la Church **Key changes**

Type unicity

- $\Lambda^{\mathtt{e}}_{\cap}$ assigns **multiple** types to each term
- $\Lambda^{\mathtt{i}}_{\cap}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \dots, A_n\}}\ e \quad \Longrightarrow \quad \frac{(\Gamma \vdash s_i : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1, \dots, s_n\} : \{A_1, \dots, A_n\}}\ i$$

Idempotent Intersection Types a la Church **Key changes**

Type unicity

- $\Lambda_\cap^{\mathsf{e}}$ assigns **multiple** types to each term
- $\Lambda_\cap^{\mathsf{i}}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i\in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1,\ldots,A_n\}}\ e \quad \Longrightarrow \quad \frac{(\Gamma \vdash s_i : A_i)_{i\in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1,\ldots,s_n\} : \{A_1,\ldots,A_n\}}\ i$$

Reduction refinement

- $\Lambda_\cap^{\mathsf{e}}$ **agnostic** substitution
- $\Lambda_\cap^{\mathsf{i}}$ **depending** (on types) substitution

Recall $\Lambda_\cap^{\mathsf{e}}$ $\quad \vdash \lambda x.x : \{A \to A, A\}$

Now

Then

So

Idempotent Intersection Types a la Church Key changes

Type unicity

- $\Lambda_\cap^{\mathsf{e}}$ assigns **multiple** types to each term
- $\Lambda_\cap^{\mathsf{i}}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}}\ e \quad \Longrightarrow \quad \frac{(\Gamma \vdash s_i : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1, \ldots, s_n\} : \{A_1, \ldots, A_n\}}\ i$$

Reduction refinement

- $\Lambda_\cap^{\mathsf{e}}$ **agnostic** substitution
- $\Lambda_\cap^{\mathsf{i}}$ **depending** (on types) substitution

Recall $\Lambda_\cap^{\mathsf{e}}$ $\quad \vdash \lambda x.x : \{\boldsymbol{A} \to \boldsymbol{A}, \boldsymbol{A}\}$ $\quad (\lambda x.\boldsymbol{x}\boldsymbol{x})(\lambda x.x) \to_\beta (\boldsymbol{\lambda x.x})(\boldsymbol{\lambda x.x})$

Now

Then

So

Idempotent Intersection Types a la Church **Key changes**

Type unicity

- $\Lambda^{\mathsf{e}}_{\cap}$ assigns **multiple** types to each term
- $\Lambda^{\mathsf{i}}_{\cap}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}}\ e \qquad \Longrightarrow \qquad \frac{(\Gamma \vdash s_i : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1, \ldots, s_n\} : \{A_1, \ldots, A_n\}}\ i$$

Reduction refinement

- $\Lambda^{\mathsf{e}}_{\cap}$ **agnostic** substitution
- $\Lambda^{\mathsf{i}}_{\cap}$ **depending** (on types) substitution

Recall $\Lambda^{\mathsf{e}}_{\cap}$ $\quad \vdash \lambda x.x : \{\boldsymbol{A} \to \boldsymbol{A}, \boldsymbol{A}\}$ $\quad (\lambda x.\boldsymbol{xx})(\lambda x.x) \to_\beta (\boldsymbol{\lambda x.x})(\boldsymbol{\lambda x.x})$

Now $\quad x : \{\boldsymbol{A} \to \boldsymbol{A}, \boldsymbol{A}\} \vdash x^{\boldsymbol{A \to A}} x^{\boldsymbol{A}} : A$

Then

So

Idempotent Intersection Types a la Church **Key changes**

Type unicity

- $\Lambda^{\mathsf{e}}_{\cap}$ assigns **multiple** types to each term
- $\Lambda^{\mathsf{i}}_{\cap}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}}\ e \quad \Longrightarrow \quad \frac{(\Gamma \vdash s_i : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1, \ldots, s_n\} : \{A_1, \ldots, A_n\}}\ i$$

Reduction refinement

- $\Lambda^{\mathsf{e}}_{\cap}$ **agnostic** substitution
- $\Lambda^{\mathsf{i}}_{\cap}$ **depending** (on types) substitution

Recall $\Lambda^{\mathsf{e}}_{\cap}$ $\quad \vdash \lambda x.x : \{\boldsymbol{A \to A}, \boldsymbol{A}\} \quad (\lambda x.\boldsymbol{xx})(\lambda x.x) \to_\beta (\boldsymbol{\lambda x.x})(\boldsymbol{\lambda x.x})$

Now $\quad x : \{\boldsymbol{A \to A}, \boldsymbol{A}\} \vdash x^{\boldsymbol{A \to A}} x^{\boldsymbol{A}} : A \quad \vdash \lambda x^A.x : \boldsymbol{A \to A} \quad \vdash \lambda x^\tau.x : \boldsymbol{A}$

Then

So

Idempotent Intersection Types a la Church **Key changes**

Type unicity

- $\Lambda^{\mathsf{e}}_{\cap}$ assigns **multiple** types to each term
- $\Lambda^{\mathsf{i}}_{\cap}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}}\; e \quad \Longrightarrow \quad \frac{(\Gamma \vdash s_i : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1, \ldots, s_n\} : \{A_1, \ldots, A_n\}}\; i$$

Reduction refinement

- $\Lambda^{\mathsf{e}}_{\cap}$ **agnostic** substitution
- $\Lambda^{\mathsf{i}}_{\cap}$ **depending** (on types) substitution

Recall $\Lambda^{\mathsf{e}}_{\cap}$ $\quad \vdash \lambda x.x : \{\boldsymbol{A \to A}, \boldsymbol{A}\} \qquad (\lambda x.\boldsymbol{xx})(\lambda x.x) \to_\beta (\boldsymbol{\lambda x.x})(\boldsymbol{\lambda x.x})$

Now $\quad x : \{\boldsymbol{A \to A}, \boldsymbol{A}\} \vdash x^{\boldsymbol{A \to A}} x^{\boldsymbol{A}} : A \qquad \vdash \lambda x^A.x : \boldsymbol{A \to A} \qquad \vdash \lambda x^\tau.x : \boldsymbol{A}$

Then $\quad (\lambda x^{\{\boldsymbol{A \to A}, \boldsymbol{A}\}}.x^{\boldsymbol{A \to A}} x^{\boldsymbol{A}})\{\lambda x^A.x, \lambda x^\tau.x\}$

So

Idempotent Intersection Types a la Church **Key changes**

Type unicity

- $\Lambda^{\mathsf{e}}_{\cap}$ assigns **multiple** types to each term
- $\Lambda^{\mathsf{i}}_{\cap}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i\in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}}\ e \quad \Longrightarrow \quad \frac{(\Gamma \vdash s_i : A_i)_{i\in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1, \ldots, s_n\} : \{A_1, \ldots, A_n\}}\ i$$

Reduction refinement

- $\Lambda^{\mathsf{e}}_{\cap}$ **agnostic** substitution
- $\Lambda^{\mathsf{i}}_{\cap}$ **depending** (on types) substitution

Recall $\Lambda^{\mathsf{e}}_{\cap}$ $\quad \vdash \lambda x.x : \{\boldsymbol{A \to A}, \boldsymbol{A}\} \qquad (\lambda x.\boldsymbol{xx})(\lambda x.x) \to_\beta (\boldsymbol{\lambda x.x})(\boldsymbol{\lambda x.x})$

Now $\quad x : \{\boldsymbol{A \to A}, \boldsymbol{A}\} \vdash x^{\boldsymbol{A\to A}} x^{\boldsymbol{A}} : A \qquad \vdash \lambda x^A.x : \boldsymbol{A \to A} \qquad \vdash \lambda x^\tau.x : \boldsymbol{A}$

Then $\quad (\lambda x^{\{\boldsymbol{A\to A}, \boldsymbol{A}\}}.x^{\boldsymbol{A\to A}} x^{\boldsymbol{A}})\{\lambda x^A.x, \lambda x^\tau.x\} \to_\beta (\lambda x^A.x)(\lambda x^\tau.x)$

So

Idempotent Intersection Types a la Church Key changes

Type unicity

- $\Lambda^{\mathsf{e}}_{\cap}$ assigns **multiple** types to each term
- $\Lambda^{\mathsf{i}}_{\cap}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}}\ e \qquad \Longrightarrow \qquad \frac{(\Gamma \vdash s_i : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1, \ldots, s_n\} : \{A_1, \ldots, A_n\}}\ i$$

Reduction refinement

- $\Lambda^{\mathsf{e}}_{\cap}$ **agnostic** substitution
- $\Lambda^{\mathsf{i}}_{\cap}$ **depending** (on types) substitution

Recall $\Lambda^{\mathsf{e}}_{\cap}$ $\quad \vdash \lambda x.x : \{A \to A, A\} \qquad (\lambda x.xx)(\lambda x.x) \to_\beta (\lambda x.x)(\lambda x.x)$

Now $\quad x : \{A \to A, A\} \vdash x^{A \to A} x^{A} : A \qquad \vdash \lambda x^A.x : A \to A \qquad \vdash \lambda x^\tau.x : A$

Then $\quad (\lambda x^{\{A \to A, A\}}.x^{A \to A} x^{A})\{\lambda x^A.x, \lambda x^\tau.x\} \quad \to_\beta \quad (\lambda x^A.x)(\lambda x^\tau.x)$

So $\quad (\lambda x.t)s \to_\beta t[s/x]$

Idempotent Intersection Types a la Church **Key changes**

Type unicity

- $\Lambda^{\mathsf{e}}_{\cap}$ assigns **multiple** types to each term
- $\Lambda^{\mathsf{i}}_{\cap}$ assigns **one** type to each term

$$\frac{(\Gamma \vdash N : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash N : \{A_1, \ldots, A_n\}}\ e \quad \Longrightarrow \quad \frac{(\Gamma \vdash s_i : A_i)_{i \in 1..n} \quad A_i \neq A_j}{\Gamma \Vdash \{s_1, \ldots, s_n\} : \{A_1, \ldots, A_n\}}\ i$$

Reduction refinement

- $\Lambda^{\mathsf{e}}_{\cap}$ **agnostic** substitution
- $\Lambda^{\mathsf{i}}_{\cap}$ **depending** (on types) substitution

Recall $\Lambda^{\mathsf{e}}_{\cap}$ $\quad \vdash \lambda x.x : \{A \to A, A\} \qquad (\lambda x.xx)(\lambda x.x) \to_\beta (\lambda x.x)(\lambda x.x)$

Now $\quad x : \{A \to A, A\} \vdash x^{A \to A} x^{A} : A \qquad \vdash \lambda x^{A}.x : A \to A \qquad \vdash \lambda x^{\tau}.x : A$

Then $\quad (\lambda x^{\{A \to A, A\}}.x^{A \to A} x^{A})\{\lambda x^{A}.x, \lambda x^{\tau}.x\} \to_\beta (\lambda x^{A}.x)(\lambda x^{\tau}.x)$

So $\quad (\lambda x.t)s \to_\beta t[s/x] \quad \Longrightarrow \quad (\lambda x^{\vec{A}}.t)\vec{s} \to_\beta t[s_1/x^{A_1}]\ldots[s_n/x^{A_n}]$

Idempotent Intersection Types a la Church Correspondence

Problem Reducing the argument of an application

$\Lambda^{\mathtt{e}}_{\cap}$ no problem

$$ts \to_\beta ts'$$

$\Lambda^{\mathtt{i}}_{\cap}$

$$\begin{aligned} t\{s_1, s_2, \ldots, s_n\} &\to_\beta t\{s'_1, s_2, \ldots, s_n\} \\ &\to_\beta t\{s'_1, s'_2, \ldots, s_n\} \\ &\to_\beta \ldots \\ &\to_\beta t\{s'_1, s'_2, \ldots, s'_n\} \end{aligned}$$

Idempotent Intersection Types a la Church Correspondence

Problem Reducing the argument of an application

$\Lambda^{\mathtt{e}}_{\cap}$ no problem

$$ts \rightarrow_\beta ts'$$

$\Lambda^{\mathtt{i}}_{\cap}$

$$\begin{aligned} t\{s_1, s_2, \ldots, s_n\} &\rightarrow_\beta t\ \{s'_1, s_2, \ldots, s_n\} \\ &\rightarrow_\beta t\ \{s'_1, s'_2, \ldots, s_n\} \\ &\rightarrow_\beta \ldots \\ &\rightarrow_\beta t\{s'_1, s'_2, \ldots, s'_n\} \end{aligned}$$

Uniformity $\vec{s}$ uniform if all s_i are equal modulo erasure *e.g.* $\{\lambda x^\tau.x, \lambda x^A.x\}$

Refinement $\vec{s}$ refines (noted $\sqsubset$) $t \in \Lambda^{\mathtt{e}}_{\cap}$ if uniform and $t = s_i$ $\sqsubset \quad \lambda x.x$

Idempotent Intersection Types a la Church **Correspondence**

Problem Reducing the argument of an application

$\Lambda^{\mathsf{e}}_{\cap}$ no problem

$$ts \to_\beta ts'$$

$\Lambda^{\mathsf{i}}_{\cap}$

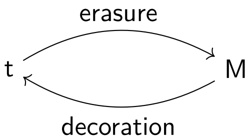
$$\begin{aligned} t\{s_1, s_2, \dots, s_n\} &\to_\beta t\{s'_1, s_2, \dots, s_n\} \\ &\to_\beta t\{s'_1, s'_2, \dots, s_n\} \\ &\to_\beta \dots \\ &\to_\beta t\{s'_1, s'_2, \dots, s'_n\} \end{aligned}$$

Uniformity $\vec{s}$ uniform if all s_i are equal modulo erasure — *e.g.* $\{\lambda x^{\tau}.x, \lambda x^{A}.x\}$

Refinement $\vec{s}$ refines (noted $\sqsubset$) $t \in \Lambda^{\mathsf{e}}_{\cap}$ if uniform and $t = s_i$ — $\sqsubset \quad \lambda x.x$

Properties

Correspondence



Idempotent Intersection Types a la Church **Correspondence**

Problem Reducing the argument of an application

$\Lambda^{\mathtt{e}}_{\cap}$ no problem

$$ts \ \to_\beta \ ts'$$

$\Lambda^{\mathtt{i}}_{\cap}$

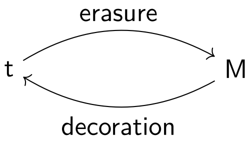
$$\begin{aligned} t\{s_1, s_2, \ldots, s_n\} &\to_\beta t\,\{s'_1, s_2, \ldots, s_n\} \\ &\to_\beta t\,\{s'_1, s'_2, \ldots, s_n\} \\ &\to_\beta \ldots \\ &\to_\beta t\{s'_1, s'_2, \ldots, s'_n\} \end{aligned}$$

Uniformity $\vec{s}$ uniform if all s_i are equal modulo erasure — *e.g.* $\{\lambda x^\tau.x, \lambda x^A.x\}$

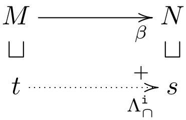
Refinement $\vec{s}$ refines (noted $\sqsubset$) $t \in \Lambda^{\mathtt{e}}_{\cap}$ if uniform and $t = s_i$ — $\sqsubset \quad \lambda x.x$

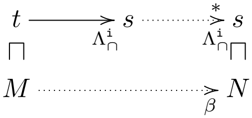
Properties

Correspondence



Simulation





Introducing memories in $\Lambda^{\mathtt{i}}_{\cap}$

Extension to $\lambda^{\mathtt{m}}_{\cap}$

- Addition of memories to the terms in $\Lambda^{\mathtt{i}}_{\cap}$
- Adaptation of definitions, properties and proofs of $\lambda^{\mathtt{m}}$ to multi-terms and multi-types

Introducing memories in $\Lambda^{\mathtt{i}}_{\cap}$

Extension to $\lambda^{\mathtt{m}}_{\cap}$

- Addition of memories to the terms in $\Lambda^{\mathtt{i}}_{\cap}$
- Adaptation of definitions, properties and proofs of $\lambda^{\mathtt{m}}$ to multi-terms and multi-types

Measure $\mathcal{W}_{\cap}$

Definition

$$\mathcal{W}(M) = \mathsf{w}(\mathsf{S}_*(M))$$

$$\begin{array}{ccccc}
M & \twoheadrightarrow & \mathsf{S}_*(M) & \longmapsto & \mathsf{w}(\mathsf{S}_*(M)) \\
\downarrow & & & & \\
N & \twoheadrightarrow & \mathsf{S}_*(N) & \longmapsto & \mathsf{w}(\mathsf{S}_*(N))
\end{array}$$

Introducing memories in $\Lambda^{\mathtt{i}}_{\cap}$

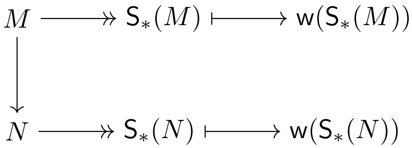
Extension to $\lambda^{\mathtt{m}}_{\cap}$

- Addition of memories to the terms in $\Lambda^{\mathtt{i}}_{\cap}$
- Adaptation of definitions, properties and proofs of $\lambda^{\mathtt{m}}$ to multi-terms and multi-types

Measure $\mathcal{W}_{\cap}$

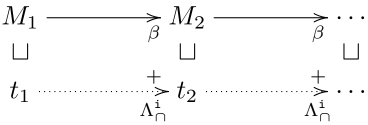
Definition

$$\mathcal{W}(M) = \mathsf{w}(\mathsf{S}_*(M))$$



Strong Normalization of $\Lambda^{\mathtt{e}}_{\cap}$

- SN of $\Lambda^{\mathtt{i}}_{\cap}$
- Correspondence
- Simulation



Conclusions and future work

Conclusions

- Overview of techniques for proving Strong Normalization
- Decreasing measures
- Auxiliar non-erasing λ^m calculus, which allowed us to:
  - define $\mathcal{W}$: DM based on counting accumulated memories in $\lambda^{\mathtt{m}}$
  - extend $\mathcal{W}$ to $\Lambda_\cap$, obtaining a simpler measure than existing ones
  - generalize Turing's WN measure to SN by adding smaller measures of D-reachable terms

Conclusions and future work

Conclusions

- Overview of techniques for proving Strong Normalization
- Decreasing measures
- Auxiliar non-erasing λ^m calculus, which allowed us to:
  - define $\mathcal{W}$: DM based on counting accumulated memories in $\lambda^{\mathtt{m}}$
  - extend $\mathcal{W}$ to $\Lambda_\cap$, obtaining a simpler measure than existing ones
  - generalize Turing's WN measure to SN by adding smaller measures of D-reachable terms

Future work

- Build a decreasing measure to System F
- Formalize them in a proof assistant
- Adapt $\mathcal{W}$ to idempotent intersection types characterizing head normal forms
- Further compare our measures with those by Gandy and de Vrijer

Why “syntactic”

- sort of convention
- soft classification of SN proofs
- but...

semantic	**syntactic**
reducibility (RC)	decreasing measures (DM) reduction of SN to WN (NK)

denotational	**operational**
RC, de Vrijer	Gandy, NK

denotational	**operational**	**syntactic**
RC, DM	NK	RC, DM, NK

- maybe **abstract** vs **concrete** would be better? **external** vs **internal**?
- we stick to the *soft* convention

  syntactic = “internal” analysis over the **structure of terms** or the **rewriting relation**

The auxiliar λ^m-calculus

Motivation

β is erasing $\qquad (\lambda x.y)t \ \rightarrow_\beta \ y$

A motivation not to erase

The auxiliar λ^m-calculus

Motivation

β is erasing $\qquad (\lambda x.y)t \;\to_\beta\; y$

A motivation not to erase

- Klop-Nederpelt lemma $\quad \mathit{INC} \wedge \mathit{WCR} \wedge \mathit{WN} \implies \mathit{SN} \wedge \mathit{CR}$

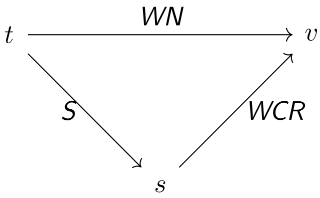
The auxiliar λ^m-calculus

Motivation

β is erasing $\qquad (\lambda x.y)t \ \rightarrow_\beta \ y$

A motivation not to erase

- Klop-Nederpelt lemma $\quad INC \wedge WCR \wedge WN \implies SN \wedge CR$
- We can obtain a decreasing measure from $INC \wedge WCR \wedge WN$
  - by WN there is a normal form v for any t
  - by WCR it is the same for every reduct s of t
  - by INC $inc(t) < inc(s) < inc(v)$
  - $dec(t) = inc(v) - inc(t)$



Intuitive definition of $\mathcal{W}$

Turing's measure "failing" example

Example: copying a redex of greater degree

$$\begin{aligned} I_1 &= \lambda x^{\tau}.x & \delta(I_1\, x) &= \mathsf{h}(\tau \to \tau) &&= 1 \\ I_2 &= \lambda x^{\tau \to \tau}.x & \delta(I_2\, I_1) &= \mathsf{h}((\tau \to \tau) \to (\tau \to \tau)) &&= 2 \\ K &= \lambda x^{\tau}.\lambda y^{\tau}.x & \delta(K\, \_) &= \mathsf{h}(\tau \to \tau \to \tau) &&= 2 \\ S_{K\,I} &= \lambda x^{\tau}.K\, x\, (I_1\, x) & \delta(S_{K\,I}\, \_) &= \mathsf{h}(\tau \to \tau) &&= 1 \end{aligned}$$

$$\mathcal{T}(\underbrace{S_{\underbrace{K}_{\mathsf{S2}}\, \underbrace{I}_{\mathsf{T1}}}\, (\underbrace{I_2\, I_1}_{\mathsf{U2}}\, x)}_{\mathsf{R1}}) = \{\underset{\mathsf{S}}{2}, \underset{\mathsf{U}}{2}, \underset{\mathsf{R}}{1}, \underset{\mathsf{T}}{1}\}$$

$$\mathcal{T}(K\, \underbrace{(\underbrace{I_2\, I_1}_{\mathsf{U'2}}\, x)}_{\mathsf{S2}}\, \underbrace{(I_1\, (\underbrace{I_2\, I_1}_{\mathsf{U'2}}\, x))}_{\mathsf{T1}}) = \{\underset{\mathsf{S}}{2}, \underset{\mathsf{U'}}{2}, \underset{\mathsf{U'}}{2}, \underset{\mathsf{T}}{1}\}$$

A first attempt: $\mathcal{T}'$ measure

Problems

$(>)$ A redex copies redexes of greater degree $\qquad \mathcal{T}(M) = [2,1] \longrightarrow \mathcal{T}(N) = [2,2]$

$(=)$ A redex copies redexes of same degree $\qquad \mathcal{T}(M) = [1,1] \longrightarrow \mathcal{T}(N) = [1,1]$

Idea

$i)$ generalize $\mathcal{T}$ to a family of measures $\mathcal{T}'_D$ indexed by a degree $D \in \mathbb{N}$, so *e.g.*

$$\mathcal{T}'_2(M) = [\underset{\mathsf{S}}{2}, \underset{\mathsf{R}}{1}] \qquad \text{and} \qquad \mathcal{T}'_1(M) = [\underset{\mathsf{R}}{1}]$$

$ii)$ instead of counting redex degrees in an isolated way,
consider also the information about remaining smaller redexes, so *e.g.*

$$\mathcal{T}'_2(M) = [\ (\underset{\mathsf{S}}{2}, \mathcal{T}'_1(M)),\ (\underset{\mathsf{R}}{1}, [])\ ] \qquad \mathcal{T}'_1(M) = [\ (\underset{\mathsf{R}}{1}, [])\ ]$$

Definition

- $\mathcal{T}'_D(M) = [(i, \mathcal{T}'_{i-1}(M)) \mid R$ is a redex of degree $i \leq D$ in $M]$
- $\mathcal{T}'(M) = \mathcal{T}'_D(M)$ where D is the maximum degree of M

A first attempt: $\mathcal{T}'$ measure

A working? example (>)

Definition

- $\mathcal{T}'_D(M) = [(d, \mathcal{T}'_{d-1}(M)) \mid R$ is a redex of degree $d \leq D$ in $M]$
- $\mathcal{T}'(M) = \mathcal{T}'_D(M)$ where D is the maximum degree of M

Example

$$M \quad = \quad \underbrace{S_{\underset{\text{S2}}{\underline{K}}} \underset{\text{T1}}{\underline{I}} \, (\underset{\text{U2}}{\underline{I_2 \, I_1}} \, x)}_{\text{R1}} \quad \longrightarrow_\beta \quad \underbrace{K \, (\underset{\text{U}'2}{\underline{I_2 \, I_1}} \, x)}_{\text{S2}} \, \underbrace{(I_1 \, (\underset{\text{U}''2}{\underline{I_2 \, I_1}} \, x))}_{\text{T1}} \quad = \quad N$$

$$\mathcal{T}'_2(M) = [\ (\underset{\text{S}}{2}, \mathcal{T}'_1(M)),\ (\underset{\text{U}}{2}, \mathcal{T}'_1(M)),\ (\underset{\text{R}}{1}, []),\ (\underset{\text{T}}{1}, [])\ ] \qquad \mathcal{T}'_1(M) = [\ (\underset{\text{R}}{1}, []),\ (\underset{\text{T}}{1}, [])\ ]$$

$$\mathcal{T}'_2(N) = [\ (\underset{\text{S}}{2}, \mathcal{T}'_1(M)),\ (\underset{\text{U}'}{2}, \mathcal{T}'_1(M)),\ (\underset{\text{U}''}{2}, \mathcal{T}'_1(M)),\ (\underset{\text{T}}{1}, [])\ ] \qquad \mathcal{T}'_1(N) = [\ (\underset{\text{T}}{1}, [])\ ]$$

$$(2, [\ (1, []),\ \ (1, [])\ ]) \qquad > \qquad (2, [\ (1, [])\ ])$$

A first attempt: $\mathcal{T}'$ measure

A failing example $(=)$

Definition

- $\mathcal{T}'_D(M) = [(d, \mathcal{T}'_{d-1}(M)) \mid R$ is a redex of degree $d \leq D$ in $M]$
- $\mathcal{T}'(M) = \mathcal{T}'_D(M)$ where D is the maximum degree of M

Example Example

$$M = \underbrace{S_{\underset{S2}{\underline{K}}\ \underset{T1}{\underline{I}}}\ \underset{U1}{\underline{(I_1\ x)}}}_{R1} \longrightarrow_\beta \underbrace{K\ \underset{U'1}{\underline{(I_1\ x)}}}_{S2}\ \underbrace{((\underset{U''1}{\underline{I_1\ x}}))}_{T1} = N$$

$$\mathcal{T}'_2(M) = [\ (\underset{S}{2}, \mathcal{T}'_1(M)),\ (\underset{R}{1}, []),\ (\underset{T}{1}, []),\ (\underset{U}{1}, []),\ ] \qquad \mathcal{T}'_1(M) = [\ (\underset{R}{1}, []),\ (\underset{T}{1}, []),\ (\underset{U}{1}, []),\ ]$$

$$\mathcal{T}'_2(N) = [\ (\underset{S}{2}, \mathcal{T}'_1(M)),\ (\underset{T}{1}, []),\ (\underset{U'}{1}, []),\ (\underset{U''}{1}, [])\ ] \qquad \mathcal{T}'_1(N) = [\ (\underset{T}{1}, []),\ (\underset{U'}{1}, []),\ (\underset{U''}{1}, [])\ ]$$

$$(2, [\ (1, []),\ (1, []),\ (1, [])\ ]) = (2, [\ (1, []),\ (1, []),\ (1, [])\ ])$$

A second attempt: $\mathcal{T}^\beta$ measure

Definition (development of a set of redexes)

reduction sequence where each step corresponds to a **residual** of a redex **in the set**

- a **residual** is a copy of a redex left after contracting another
- notation: $\rho : m \xrightarrow{D}{}^{*}_{\beta} m'$

Idea

i) generalize $\mathcal{T}$ to a family of measures $\mathcal{T}^\beta_D$ indexed by a degree $D \in \mathbb{N}$

ii) instead of isolatedly counting redexes degrees, consider:

- from set of redexes of degree D
- target M' from every development $\rho : M \xrightarrow{D}{}^{*}_{\beta} M'$
- multiset of those $\mathcal{T}^\beta_{D-1}(M')$

Definition

$$\mathcal{T}^\beta_D(M) = [\ (i, \mathcal{V}^\beta_i(M)) \mid R \text{ is a redex of degree } i \leq D \text{ in } M\ ]$$

$$\mathcal{V}^\beta_D(M) = [\ \mathcal{T}^\beta_{D-1}(M') \mid \rho : M \xrightarrow{D}{}^{*}_{\beta} M'\ ]$$

Problem: our technique to prove it decreases does not work because of erasing

A second attempt: $\mathcal{T}^\beta$ measure

Definition (development of a set of redexes)

reduction sequence where each step corresponds to a **residual** of a redex **in the set**

- a **residual** is a copy of a redex left after contracting another
- notation: $\rho : m \xrightarrow{D}{}^{*}_{\beta} m'$

Idea

i) generalize $\mathcal{T}$ to a family of measures $\mathcal{T}^\beta_D$ indexed by a degree $D \in \mathbb{N}$

ii) instead of isolatedly counting redexes degrees, consider:

- from set of redexes of degree D
- target M' from every development $\rho : M \xrightarrow{D}{}^{*}_{\beta} M'$
- multiset of those $\mathcal{T}^\beta_{D-1}(M')$

Definition

$$\mathcal{T}^\beta_D(M) = [\ (i, \mathcal{V}^\beta_i(M)) \mid R \text{ is a redex of degree } i \leq D \text{ in } M\ ]$$

$$\mathcal{V}^\beta_D(M) = [\ \mathcal{T}^\beta_{D-1}(M') \mid \rho : M \xrightarrow{D}{}^{*}_{\beta} M'\ ]$$

Problem: our technique to prove it decreases does not work because of erasing

A second attempt: $\mathcal{T}^{\beta}$ measure

Definition

$$\mathcal{T}^{\beta}_{D}(M) = [\ (i, \mathcal{V}^{\beta}_{i}(M)) \mid R \text{ is a redex of degree } i \leq D \text{ in } M\ ]$$

$$\mathcal{V}^{\beta}_{D}(M) = [\ \mathcal{T}^{\beta}_{D-1}(M') \mid \rho : M \xrightarrow{D}{}^{*}_{\beta} M'\ ]$$

Reasoning about the auxiliar measure $\mathcal{V}^{\beta}_{D}$

Consider

$$M \underset{R}{\rightarrow_{\beta}} N \qquad \mathcal{T}^{\beta}_{D}(M) > \mathcal{T}^{\beta}_{D}(N) \qquad \mathcal{V}^{\beta}_{D}(M) > \mathcal{V}^{\beta}_{D}(N)$$

1. Copying a redex of same degree $(=)$
   - injective mapping from devs of $\mathcal{V}^{m}_{D}(N)$ to devs of $\mathcal{V}^{m}_{D}(M)$ $\quad R\rho : M \rightarrow_{\beta} N \rightarrow_{\beta}{}^{*} N'$

$$\mathcal{V}^{\beta}_{D}(M) > \mathcal{V}^{\beta}_{D}(N) \qquad \mathcal{T}^{\beta}_{D}(M) > \mathcal{T}^{\beta}_{D}(N)$$

2. Copying a redex of higher degree $(>)$
   - not clear the same can be done: a ρ may erase R

$$\mathcal{V}^{\beta}_{D}(M') = \mathcal{V}^{\beta}_{D}(N') \qquad \mathcal{T}^{\beta}_{D}(M') = \mathcal{T}^{\beta}_{D}(N')$$

A second attempt: $\mathcal{T}^\beta$ measure

Definition

$$\mathcal{T}^\beta_D(M) = [\ (i, \mathcal{V}^\beta_i(M)) \mid R \text{ is a redex of degree } i \leq D \text{ in } M\ ]$$

$$\mathcal{V}^\beta_D(M) = [\ \mathcal{T}^\beta_{D-1}(M') \mid \rho : M \xrightarrow{D}{}^{*}_{\beta} M'\ ]$$

Reasoning about the auxiliar measure $\mathcal{V}^\beta_D$

Consider

$$M \underset{R}{\to_\beta} N \qquad \mathcal{T}^\beta_D(M) > \mathcal{T}^\beta_D(N) \qquad \mathcal{V}^\beta_D(M) > \mathcal{V}^\beta_D(N)$$

1. Copying a redex of same degree $(=)$
   - injective mapping from devs of $\mathcal{V}^m_D(N)$ to devs of $\mathcal{V}^m_D(M)$ $\qquad R\rho : M \to_\beta N \to_\beta {}^* N'$

$$\mathcal{V}^\beta_D(M) > \mathcal{V}^\beta_D(N) \qquad \mathcal{T}^\beta_D(M) > \mathcal{T}^\beta_D(N)$$

2. Copying a redex of higher degree $(>)$
   - not clear the same can be done: a ρ may erase R

$$\mathcal{V}^\beta_D(M') = \mathcal{V}^\beta_D(N') \qquad \mathcal{T}^\beta_D(M') = \mathcal{T}^\beta_D(N')$$

$\mathcal{T}^m$ measure

Idea

$i)$ generalize $\mathcal{T}$ to a family of measures $\mathcal{T}^m_D$ indexed by a degree $D \in \mathbb{N}$

$ii)$ instead of isolatedly counting redexes degrees,
consider the multiset of the measures $\mathcal{T}^m_{D-1}$ of every target of a development of degree D

Definition

$$\mathcal{T}^m_D(t) = [\ (i, \mathcal{V}^m_i(t)) \mid R \text{ is a redex of degree } i \leq D \text{ in } t\ ]$$

$$\mathcal{V}^m_D(t) = [\ \mathcal{T}^m_{D-1}(t') \mid \rho : t \xrightarrow{D}{}^{*}_{m} t'\ ]$$

Lemmas

- **Forget/decrease**: forgetful reduction $\triangleright$ decreases $\mathcal{T}^m$
- **High/increase**: contracting a redex of degree $D > i$ increases (non-strictly) $\mathcal{T}^m_i$
  only $\leq i$, no D, in $\mathcal{T}^m_i$ — no erasing of any $\leq i$ — maybe copies of $\leq i$
- **Low/decrease**: contracting a redex of degree $i < D$ decreases (strictly) $\mathcal{T}^m_D$
  injective mappings from devs of $\mathcal{V}^m_D(N)$ to devs of $\mathcal{V}^m_D(M)$

Theorem

$$M \to_\beta N \qquad \Longrightarrow \qquad \mathcal{T}^m(M) > \mathcal{T}^m(N)$$

$\mathcal{T}^m$ measure

Idea

$i)$ generalize $\mathcal{T}$ to a family of measures $\mathcal{T}^m_D$ indexed by a degree $D \in \mathbb{N}$

$ii)$ instead of isolatedly counting redexes degrees,
consider the multiset of the measures $\mathcal{T}^m_{D-1}$ of every target of a development of degree D

Definition

$$\mathcal{T}^m_D(t) = [\ (i, \mathcal{V}^m_i(t)) \mid R \text{ is a redex of degree } i \leq D \text{ in } t\ ]$$

$$\mathcal{V}^m_D(t) = [\ \mathcal{T}^m_{D-1}(t') \mid \rho : t \xrightarrow{D}{}^{*}_{m} t'\ ]$$

Lemmas

- **Forget/decrease**: forgetful reduction $\triangleright$ decreases $\mathcal{T}^m$
- **High/increase**: contracting a redex of degree $D > i$ increases (non-strictly) $\mathcal{T}^m_i$
  only $\leq i$, no D, in $\mathcal{T}^m_i$ no erasing of any $\leq i$ maybe copies of $\leq i$
- **Low/decrease**: contracting a redex of degree $i < D$ decreases (strictly) $\mathcal{T}^m_D$
  injective mappings from devs of $\mathcal{V}^m_D(N)$ to devs of $\mathcal{V}^m_D(M)$

Theorem

$$M \ \to_\beta \ N \qquad \Longrightarrow \qquad \mathcal{T}^m(M) > \mathcal{T}^m(N)$$