

Una medida decreciente directa para tipos intersección

Pablo
Barenbaum
UNQ/CONICET & ICC/UBA

Simona Ronchi
della Rocca
Università di Torino

Cristian Sottile
UNQ & CONICET & UBA

VI Jornadas de Investigadores en Formación
CyT, UNQ

14 de noviembre de 2025



Formalismo para representar a todas las funciones computables

Variables x

Funciones λ

Aplicaciones @

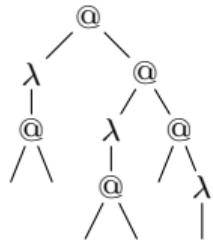
Formalismo para representar a todas las funciones computables

Variables x

Funciones λ

Aplicaciones @

Intuición



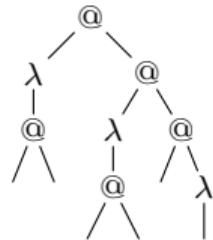
Formalismo para representar a todas las funciones computables

Variables x

Funciones λ

Aplicaciones @

Intuición



Cómputo

Transformaciones sucesivas en el árbol

(al juntar @ y λ)

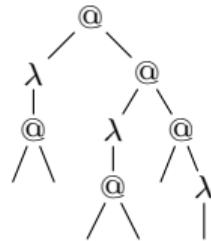
Formalismo para representar a todas las funciones computables

Variables x

Funciones λ

Aplicaciones @

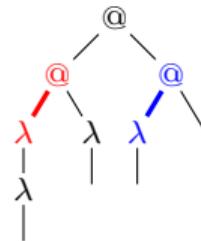
Intuición



Cómputo

Transformaciones sucesivas en el árbol

(al juntar @ y λ)



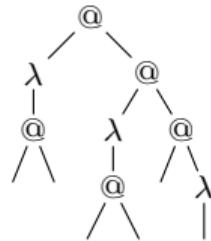
Formalismo para representar a todas las funciones computables

Variables x

Funciones λ

Aplicaciones @

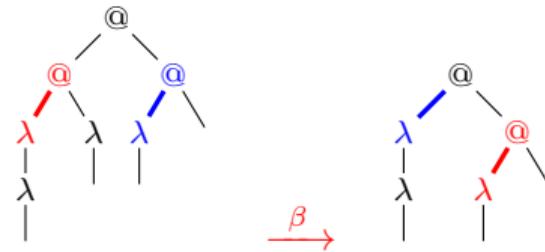
Intuición



Cómputo

Transformaciones sucesivas en el árbol

(al juntar @ y λ)



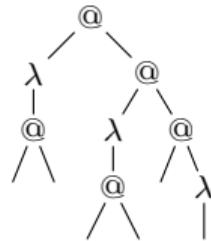
Formalismo para representar a todas las funciones computables

Variables x

Funciones λ

Aplicaciones @

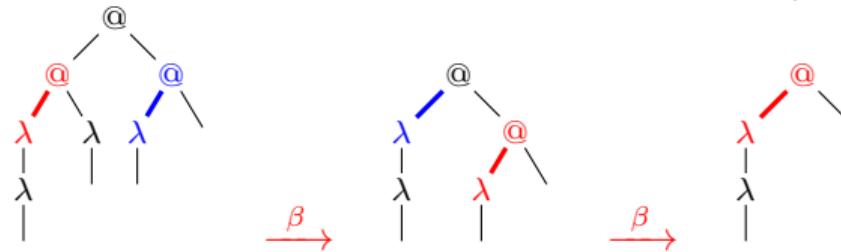
Intuición



Cómputo

Transformaciones sucesivas en el árbol

(al juntar @ y λ)



λ -cálculo

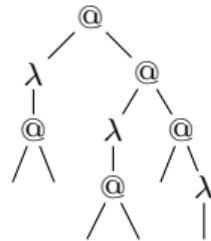
Formalismo para representar a todas las funciones computables

Variables x

Funciones λ

Aplicaciones @

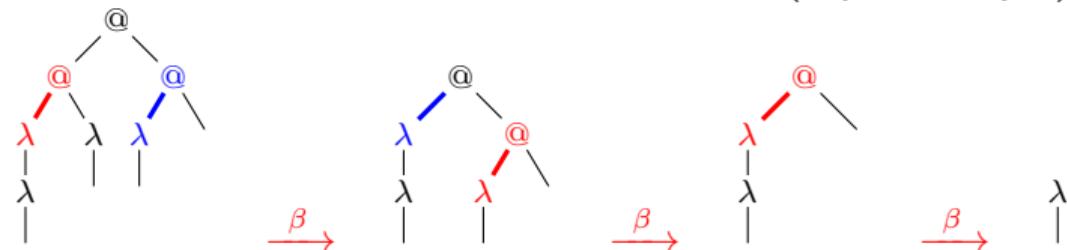
Intuición



Cómputo

Transformaciones sucesivas en el árbol

(al juntar @ y λ)

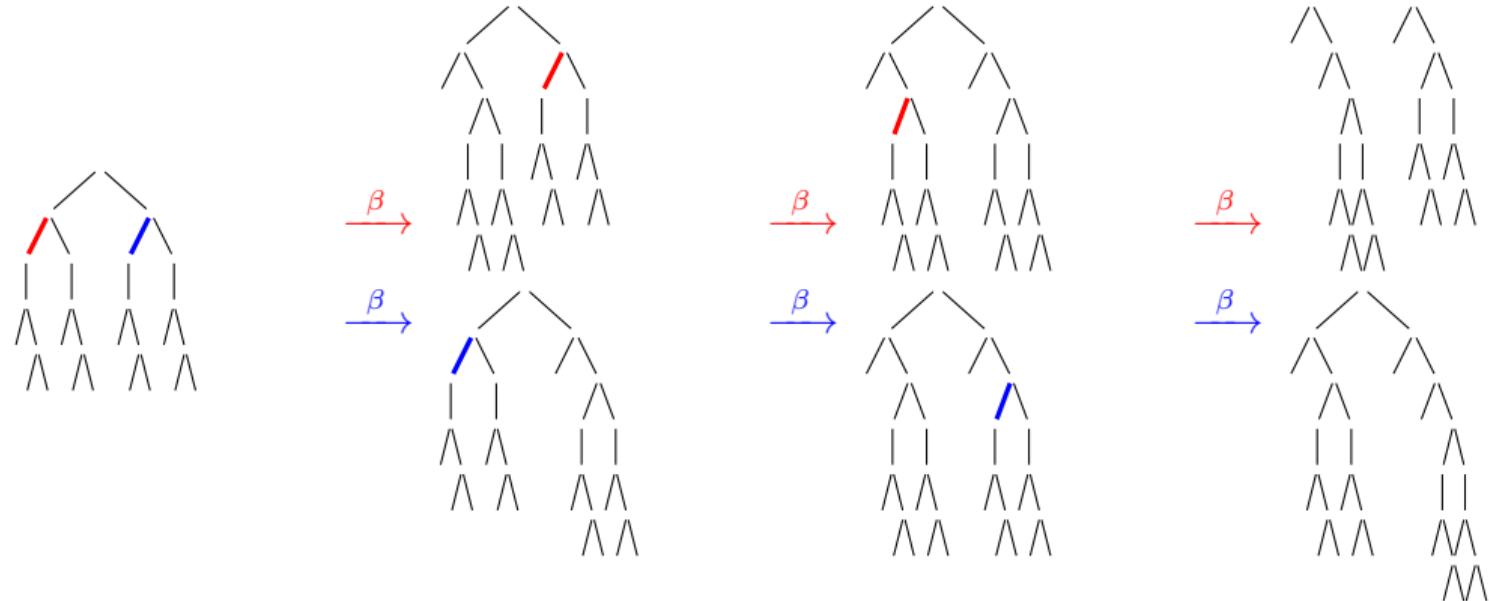


Terminación

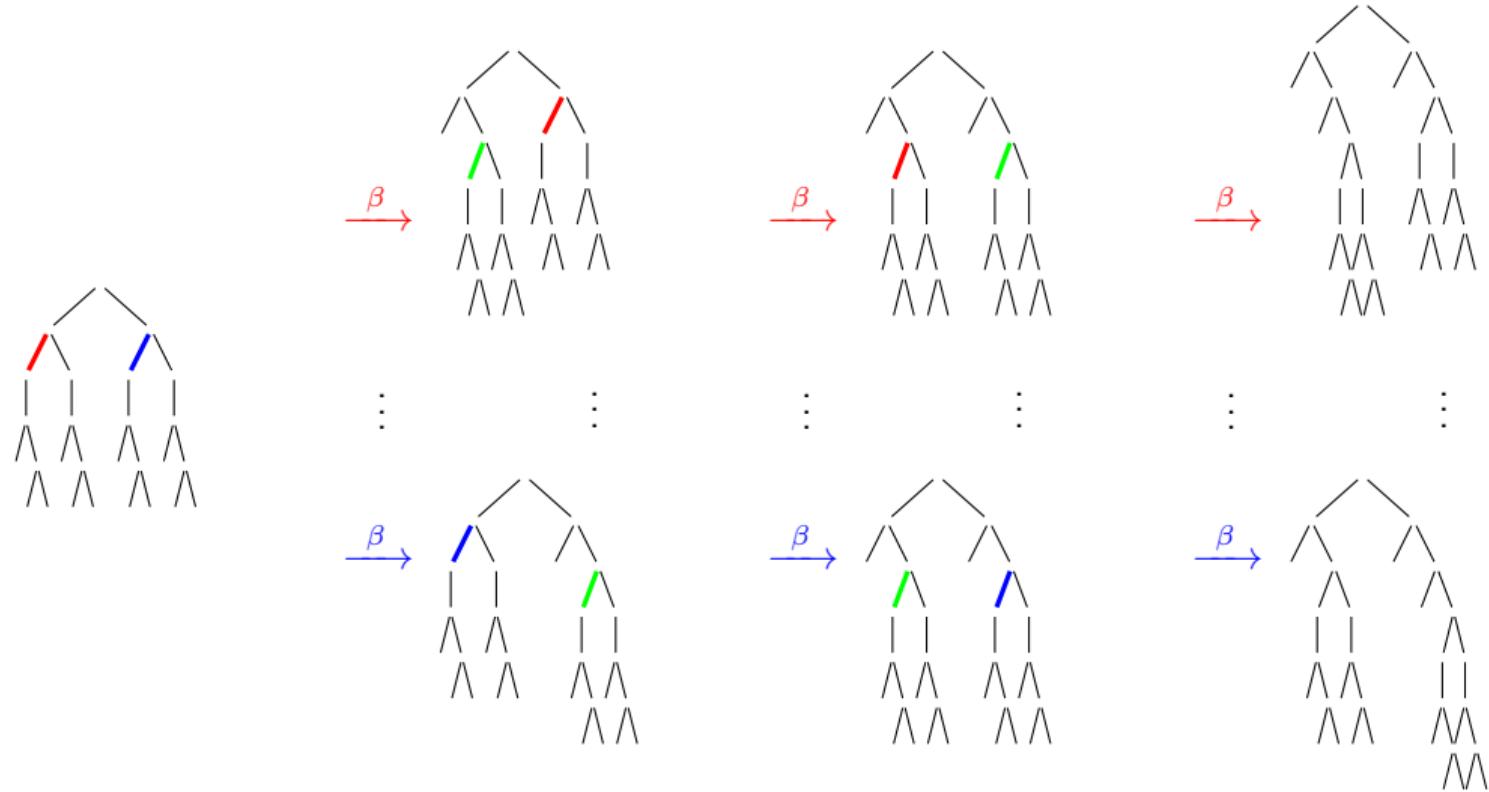
Árbol sin posibles transformaciones

(no hay @ y λ juntos)

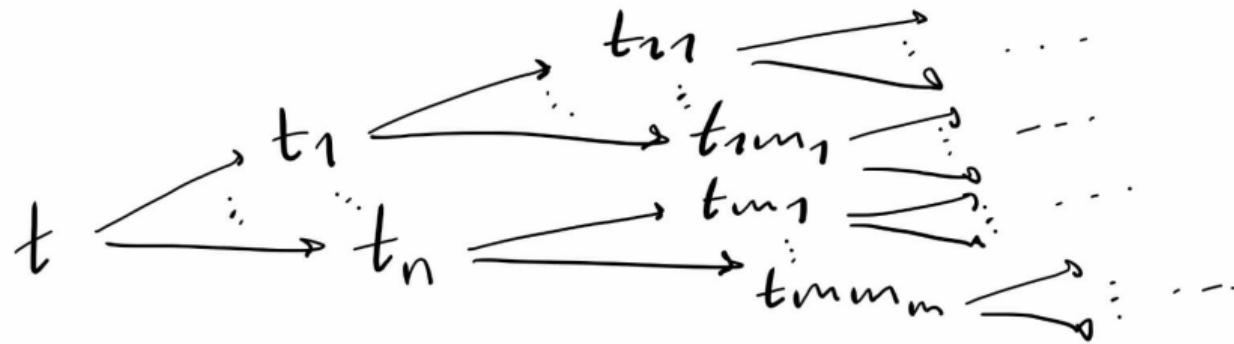
El árbol de ejecución



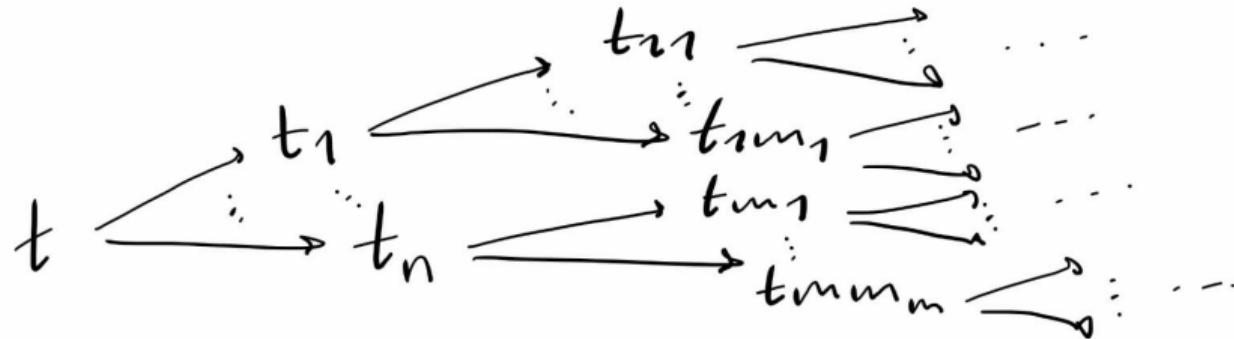
El árbol de ejecución



El árbol de ejecución

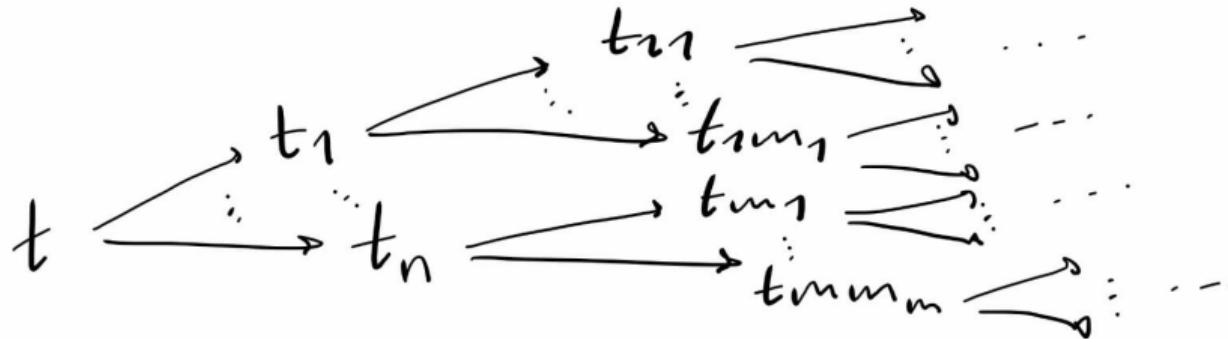


La propiedad de terminación: definición



Todas las ramas del árbol de ejecución son finitas

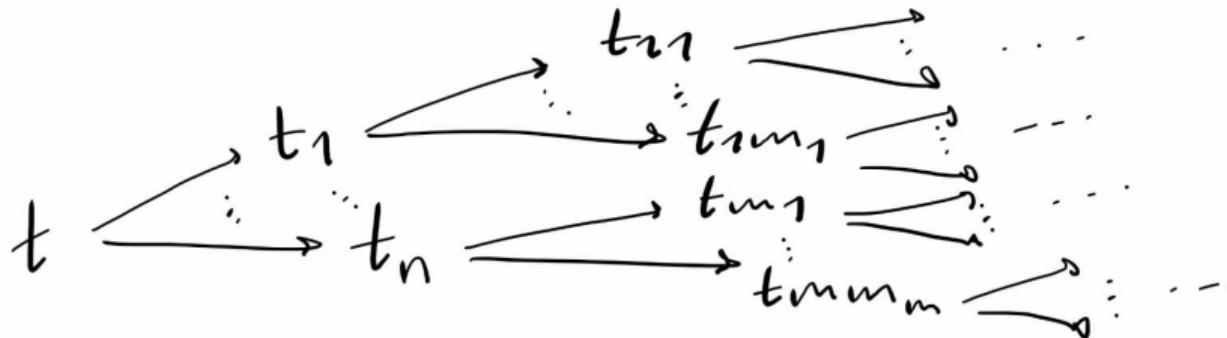
La propiedad de terminación: definición



Todas las ramas del árbol de ejecución son finitas

- ▶ Es conocida para sistemas clásicos

La propiedad de terminación: definición

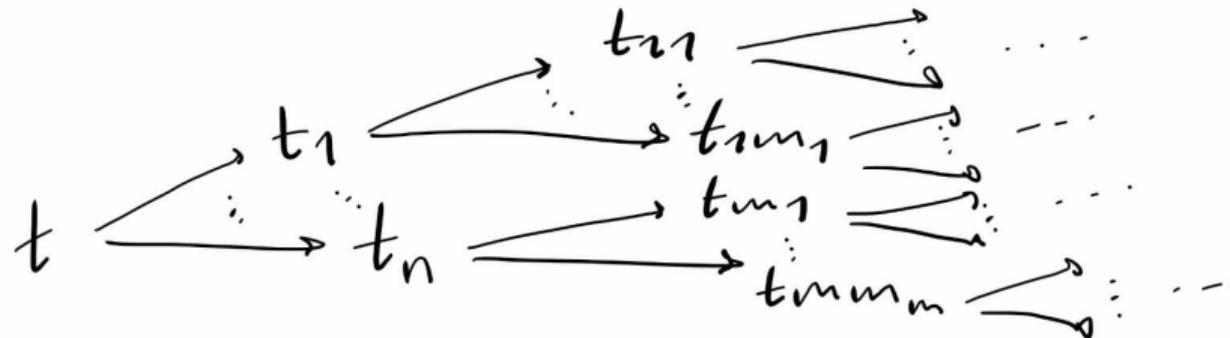


Todas las ramas del árbol de ejecución son finitas

- ▶ Es conocida para sistemas clásicos
- ▶ Es **importante** para λ -cálculos tipados

nos interesa demostrarla cuando definimos un nuevo cálculo

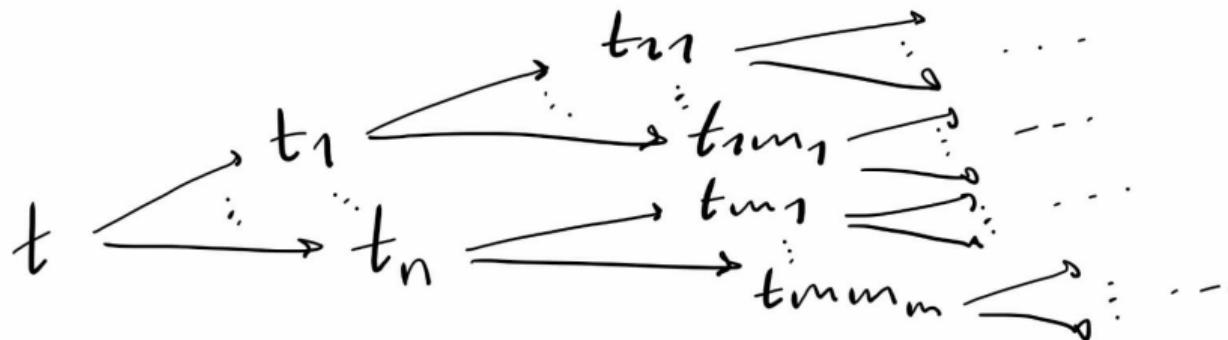
La propiedad de terminación: definición



Todas las ramas del árbol de ejecución son finitas

- ▶ Es conocida para sistemas clásicos
- ▶ Es **importante** para λ -cálculos tipados
 - nos interesa demostrarla cuando definimos un nuevo cálculo
- ▶ Es de las más **difíciles** de probar
 - nos interesa seguir encontrando formas de demostrarla

La propiedad de terminación: definición



Todas las ramas del árbol de ejecución son finitas

- ▶ Es conocida para sistemas clásicos
- ▶ Es **importante** para λ -cálculos tipados
 - nos interesa demostrarla cuando definimos un nuevo cálculo
- ▶ Es de las más **difíciles** de probar
 - nos interesa seguir encontrando formas de demostrarla
- ▶ Dos técnicas

semántica

reducibilidad

sintáctica

medidas decrecientes

El método (sintáctico) de medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

función de programas en un orden bien fundado

El método (sintáctico) de medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

función de programas en un orden bien fundado
tal que decrece con cada paso de computación

El método (sintáctico) de medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

función de programas en un orden bien fundado

tal que decrece con cada paso de computación

$$\# : \Lambda \rightarrow \text{Ord} \quad t \rightarrow s \implies \#(t) > \#(s)$$

El método (sintáctico) de medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

función de programas en un orden bien fundado

tal que decrece con cada paso de computación

$$\# : \Lambda \rightarrow \text{Ord} \quad t \rightarrow s \implies \#(t) > \#(s)$$

Ejemplo

$$30 > 23 > 16 > 10 > 4 > 0 \not> \dots$$

El método (sintáctico) de medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

función de **programas** en un **orden bien fundado**

tal que **decrece** con cada paso de computación

$$\# : \Lambda \rightarrow \text{Ord} \quad t \rightarrow s \implies \#(t) > \#(s)$$

Ejemplo

$$\begin{array}{ccccccccccccc} 30 & > & 23 & > & 16 & > & 10 & > & 4 & > & 0 & \not> & \dots \\ t_0 & \rightarrow & t_1 & \rightarrow & t_2 & \rightarrow & t_3 & \rightarrow & t_4 & \rightarrow & t_5 & \not\rightarrow & \dots \end{array}$$

El método (sintáctico) de medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

función de **programas** en un **orden bien fundado**

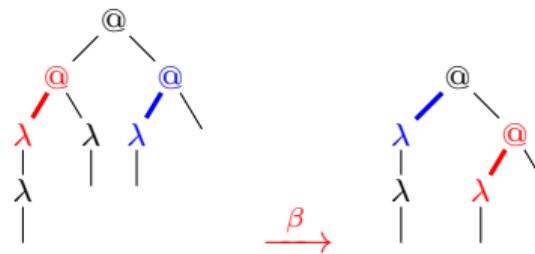
tal que **decrece** con cada paso de computación

$$\# : \Lambda \rightarrow \text{Ord} \quad t \rightarrow s \implies \#(t) > \#(s)$$

Ejemplo

$$\begin{array}{ccccccccccccc} 30 & > & 23 & > & 16 & > & 10 & > & 4 & > & 0 & \not> & \dots \\ t_0 & \rightarrow & t_1 & \rightarrow & t_2 & \rightarrow & t_3 & \rightarrow & t_4 & \rightarrow & t_5 & \not\rightarrow & \dots \end{array}$$

Intuición



El método (sintáctico) de medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

función de **programas** en un **orden bien fundado**

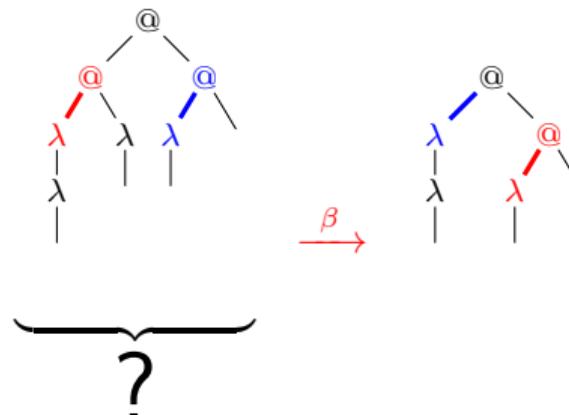
tal que **decrece** con cada paso de computación

$$\# : \Lambda \rightarrow \text{Ord} \quad t \rightarrow s \implies \#(t) > \#(s)$$

Ejemplo

$$\begin{array}{ccccccccccccc} 30 & > & 23 & > & 16 & > & 10 & > & 4 & > & 0 & \not> & \dots \\ t_0 & \rightarrow & t_1 & \rightarrow & t_2 & \rightarrow & t_3 & \rightarrow & t_4 & \rightarrow & t_5 & \not\rightarrow & \dots \end{array}$$

Intuición



El método (sintáctico) de medidas decrecientes [Gandy'80, de Vrijer'87]

Definición

función de **programas** en un **orden bien fundado**

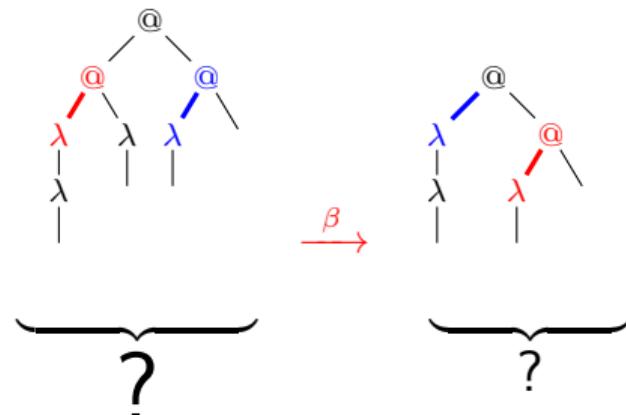
tal que **decrece** con cada paso de computación

$$\# : \Lambda \rightarrow \text{Ord} \quad t \rightarrow s \implies \#(t) > \#(s)$$

Ejemplo

$$\begin{array}{ccccccccccccc} 30 & > & 23 & > & 16 & > & 10 & > & 4 & > & 0 & \not> & \dots \\ t_0 & \rightarrow & t_1 & \rightarrow & t_2 & \rightarrow & t_3 & \rightarrow & t_4 & \rightarrow & t_5 & \not\rightarrow & \dots \end{array}$$

Intuición



Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

Tipos intersección admitiendo autoaplicación

$$x : ? \vdash_{\lambda} xx : A$$

$$x : \{A \rightarrow A, A\} \vdash_{\cap} xx : A$$

Medidas decrecientes en Λ_{\cap}

- ▶ Había dos
 - ▶ Kfoury & Wells'95
 - ▶ Boudol'03
- ▶ La nuestra introduce mejoras
 - ▶ dominio más simple: **números naturales**
 - ▶ es total: **funciona para todas las ramas**
 - ▶ yapa: definimos una versión de Λ_{\cap} con tipado explícito

Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

1. Agregamos otro constructor de programas

Variables x

Funciones λ

Aplicaciones @

Memorias ⟨ ⟩

Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

1. Agregamos otro constructor de programas

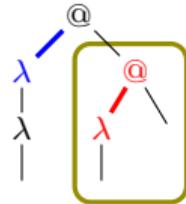
Variables x

Funciones λ

Aplicaciones @

Memorias ⟨ ⟩

2. Recordamos los argumentos consumidos



Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

1. Agregamos otro constructor de programas

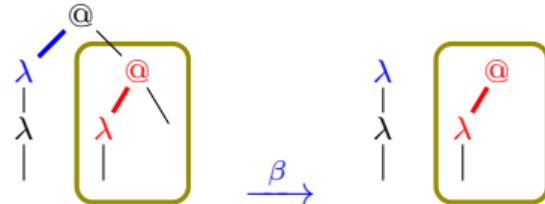
Variables x

Funciones λ

Aplicaciones @

Memorias ⟨ ⟩

2. Recordamos los argumentos consumidos



Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

1. Agregamos otro constructor de programas

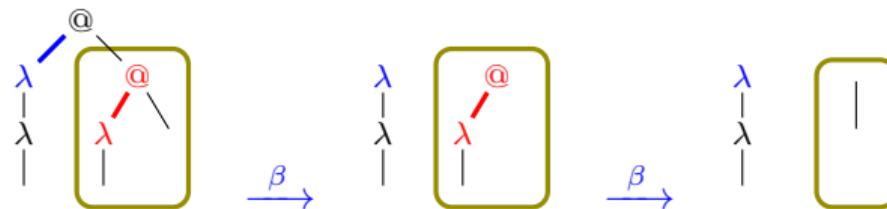
Variables x

Funciones λ

Aplicaciones @

Memorias ⟨ ⟩

2. Recordamos los argumentos consumidos



Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

1. Agregamos otro constructor de programas

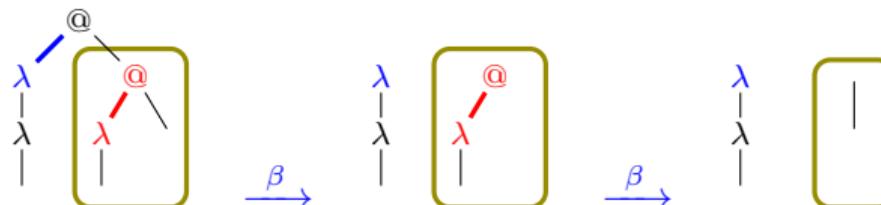
Variables x

Funciones λ

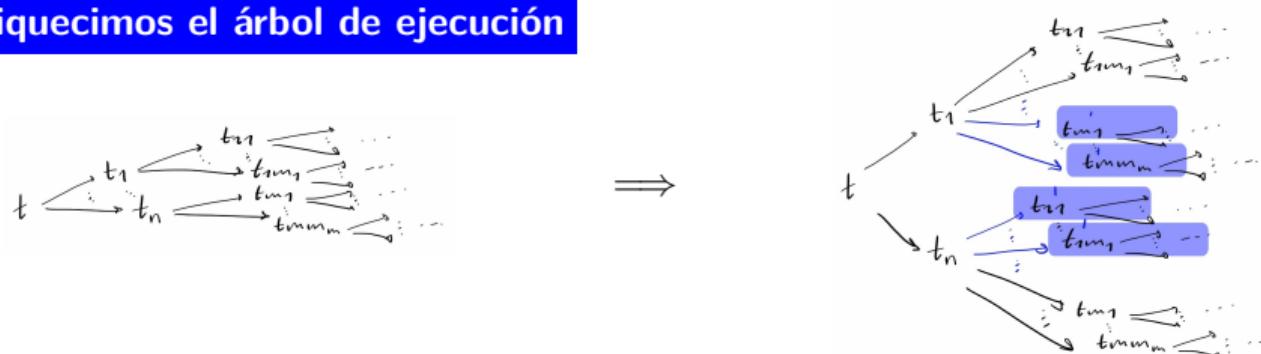
Aplicaciones @

Memorias ()

2. Recordamos los argumentos consumidos



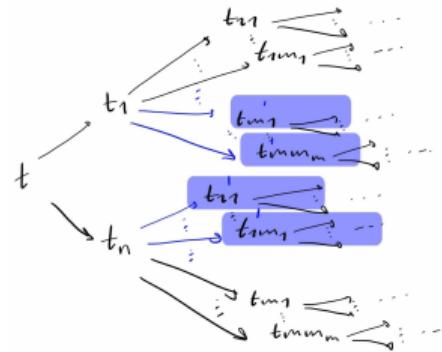
3. Enriquecemos el árbol de ejecución



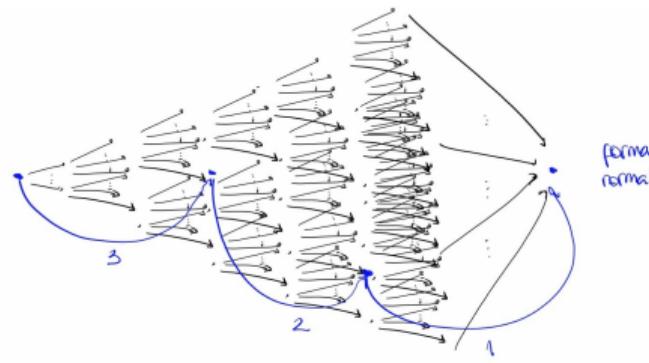
Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

4. Reconstruimos parcialmente el árbol



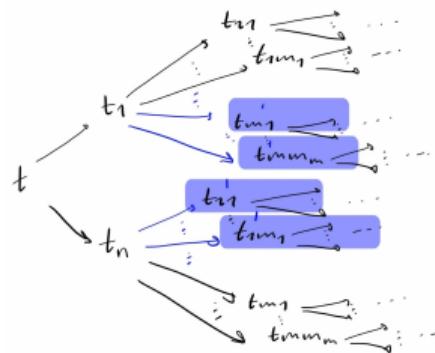
Operación $S_d(t)$



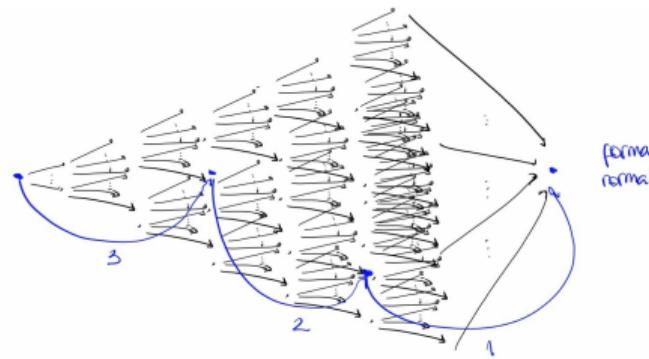
Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

4. Reconstruimos parcialmente el árbol



Operación $S_d(t)$



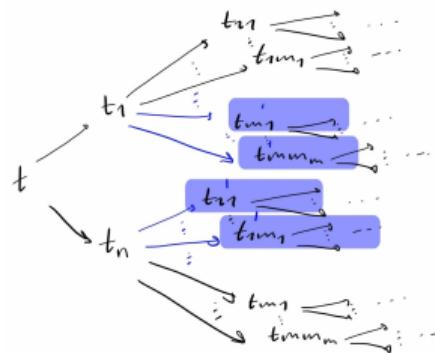
5. Obtenemos el resultado sin ejecutar

$$t \langle u_1 \rangle \dots \langle u_n \rangle$$

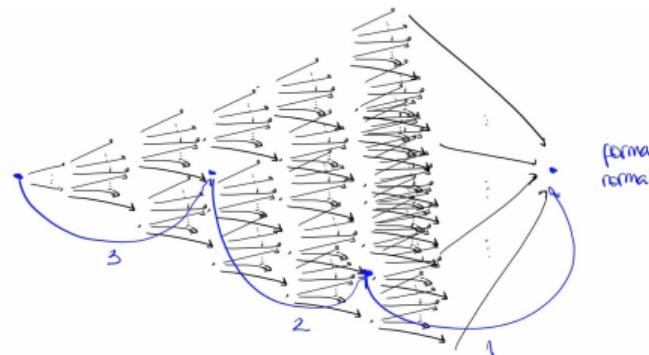
Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

4. Reconstruimos parcialmente el árbol



Operación $S_d(t)$



5. Obtenemos el resultado sin ejecutar

$$t \langle u_1 \rangle \dots \langle u_n \rangle$$

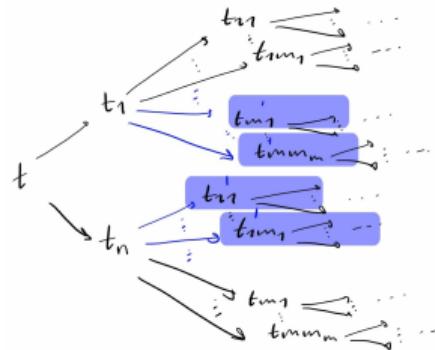
6. Contamos memorias en el resultado

$$\#(t \langle u_1 \rangle \dots \langle u_n \rangle) = n$$

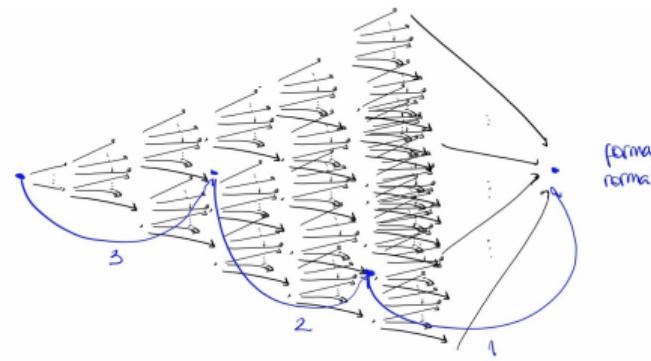
Una medida decreciente para Λ_{\cap} (tipos intersección idempotente)

[Barenbaum, Ronchi della Rocca, S.'25]

4. Reconstruimos parcialmente el árbol



Operación $S_d(t)$



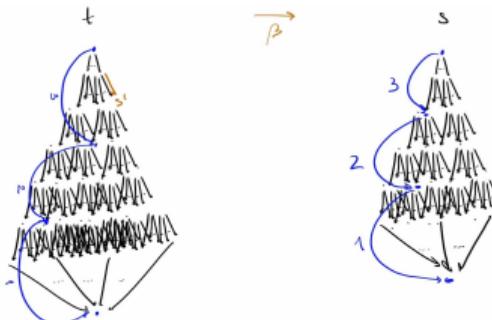
5. Obtenemos el resultado sin ejecutar

$$t \langle u_1 \rangle \dots \langle u_n \rangle$$

6. Contamos memorias en el resultado

$$\#(t \langle u_1 \rangle \dots \langle u_n \rangle) = n$$

7. Rastreamos la memoria que aporta $t \rightarrow s$



Conclusión y trabajo futuro

Conclusión

- ▶ Encontramos *algo* “dentro” de los programas en Λ_n que decrece con la ejecución

Conclusión y trabajo futuro

Conclusión

- ▶ Encontramos *algo* “dentro” de los programas en Λ_n que decrece con la ejecución
- ▶ Mejora sobre las medidas existentes: es un número y es total

Conclusión y trabajo futuro

Conclusión

- ▶ Encontramos *algo* “dentro” de los programas en Λ_n que decrece con la ejecución
- ▶ Mejora sobre las medidas existentes: es un número y es total

Trabajo futuro

- ▶ cota superior \implies cota exacta

Conclusión y trabajo futuro

Conclusión

- ▶ Encontramos *algo* “dentro” de los programas en Λ_n que decrece con la ejecución
- ▶ Mejora sobre las medidas existentes: es un número y es total

Trabajo futuro

- ▶ cota superior \implies cota exacta
- ▶ polimorfismo restringido \implies polimorfismo

Conclusión y trabajo futuro

Conclusión

- ▶ Encontramos *algo* “dentro” de los programas en Λ_n que decrece con la ejecución
- ▶ Mejora sobre las medidas existentes: es un número y es total

Trabajo futuro

- ▶ cota superior \implies cota exacta
- ▶ polimorfismo restringido \implies polimorfismo
- ▶ formalización (“implementación” en asistente de pruebas)