Garantías de terminación en λ-cálculos polimórficos

Cristian Sottile

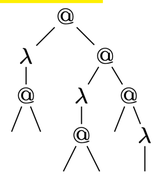
I. λ-cálculo

Formalismo para representar a todas las funciones computables

Constructores

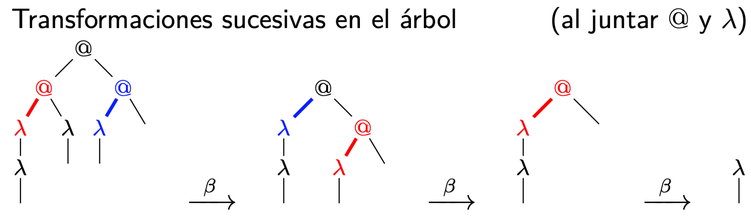
Variables x Funciones λ Aplicaciones @

Intuición



Cómputo

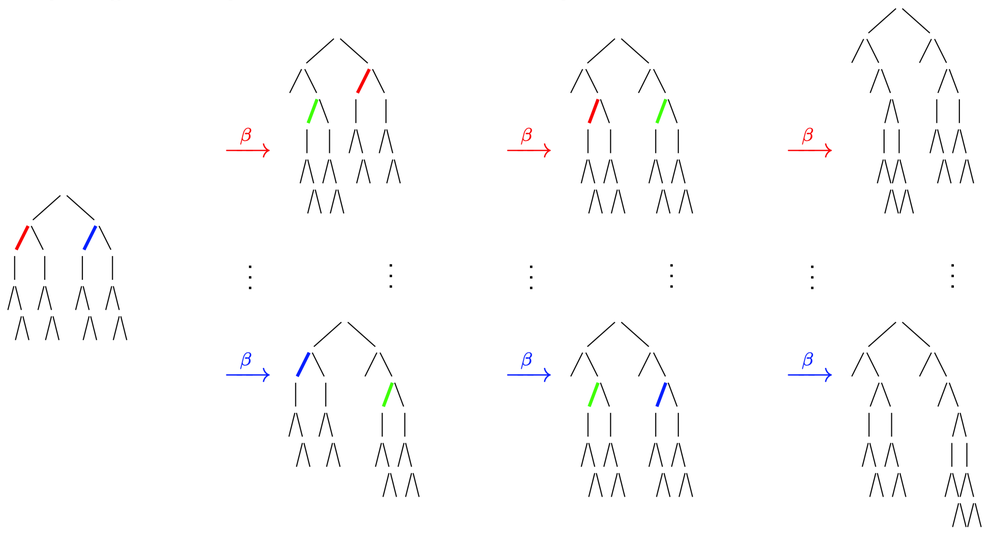
Transformaciones sucesivas en el árbol (al juntar @ y λ)



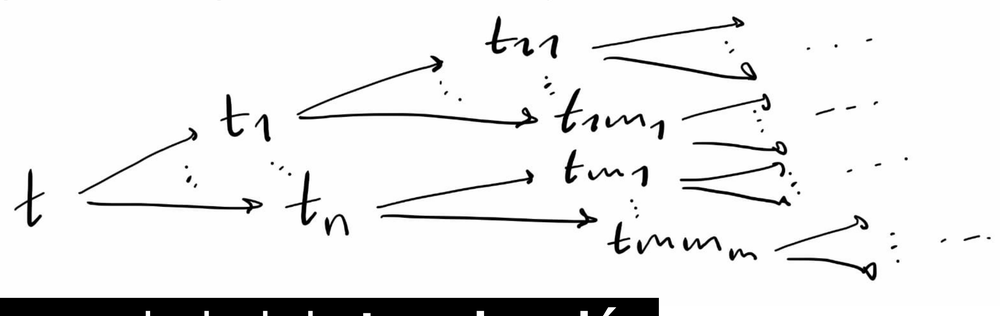
Terminación

Árbol sin posibles transformaciones (no hay @ y λ juntos)

Un programa puede tener varias posibles transformaciones



Así que también podemos ver a la ejecución como un árbol



II. La propiedad de **terminación**

todas las ramas del árbol de ejecución son finitas

- Es importante para λ-cálculos tipados
- Es de las más difíciles de demostrar
- Hay dos técnicas generales para probarla
  - el método de **reducibilidad** (semántico)
    interpretación de tipos con estructura + parametrización (candidatos) de tipos sin estructura
    (conjuntos en la semántica capturan propiedades que garantizan terminación de sus programas)

$$\begin{aligned} \rho &: \mathrm{Var} \to \mathrm{Type} \\ [\![X]\!]_\rho &= \rho(X) \\ [\![A \to B]\!]_\rho &= \{t : A \to B \mid \text{ for all } s \in [\![A]\!]_\rho,\ t(s) \in [\![B]\!]_\rho\} \\ [\![\forall X.A]\!]_\rho &= \{t : \forall X.A \mid \text{ for all } B, \mathcal{B},\ t(B) \in [\![A]\!]_{\rho\cdot[\mathcal{B}/X]}\} \end{aligned}$$

  - el método de **medidas decrecientes** (sintáctico)
    función de programas en orden bien fundado, tal que decrece con cada paso de computación
    (rama de ejecución infinita implicaría cadena decreciente infinita en el orden bien fundado, absurdo)

$$\# : \Lambda \to \mathrm{Ord} \qquad t \to s \implies \#(t) > \#(s)$$

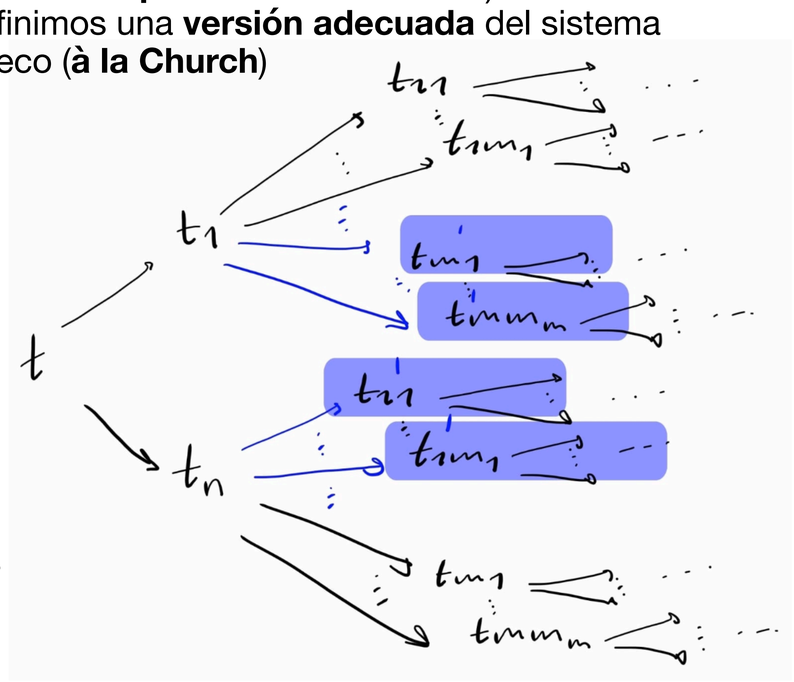
III. Una medida decreciente para $\Lambda_\cap$ (tipos intersección idempotente)

P. Barenbaum, S. Ronchi Della Rocca, C. Sottile. *Strong normalization through idempotent intersection types: a new syntactical approach*. MFPS 2025. ENTICS vol. 5

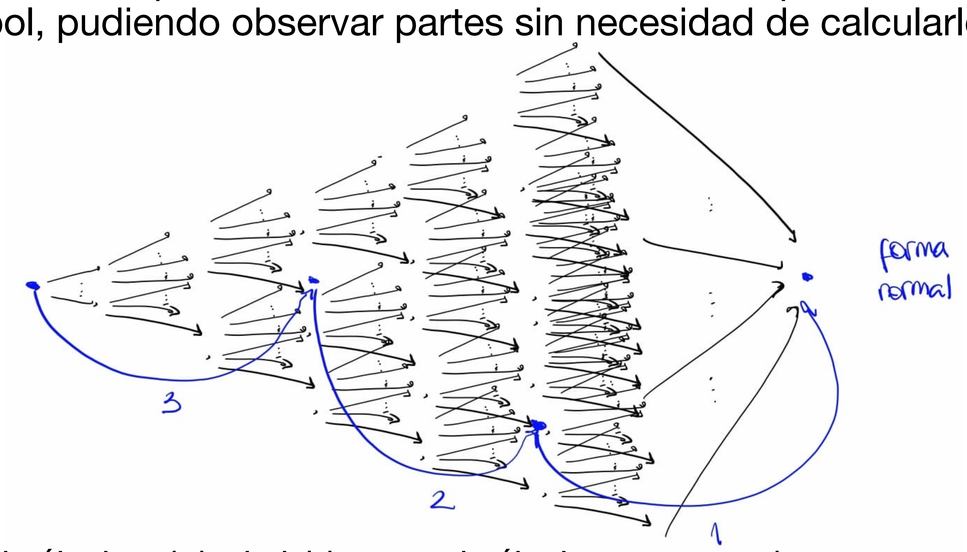
- Terminación es una propiedad conocida para $\Lambda_\cap$
- Hay pruebas basadas en las dos técnicas
- Encontramos una **nueva medida decreciente** con mejoras sobre las actuales:
  - su dominio es más simple (**números naturales**)
  - es completa (**funciona para todas las ramas**)
- Para aplicarla, definimos una **versión adecuada** del sistema con tipado intrínseco (**à la Church**)

Definimos una extensión con memorias que replica las ramas de cada nodo en sus hermanos, resultando en un árbol más ancho (y profundo), que llamamos "denso"

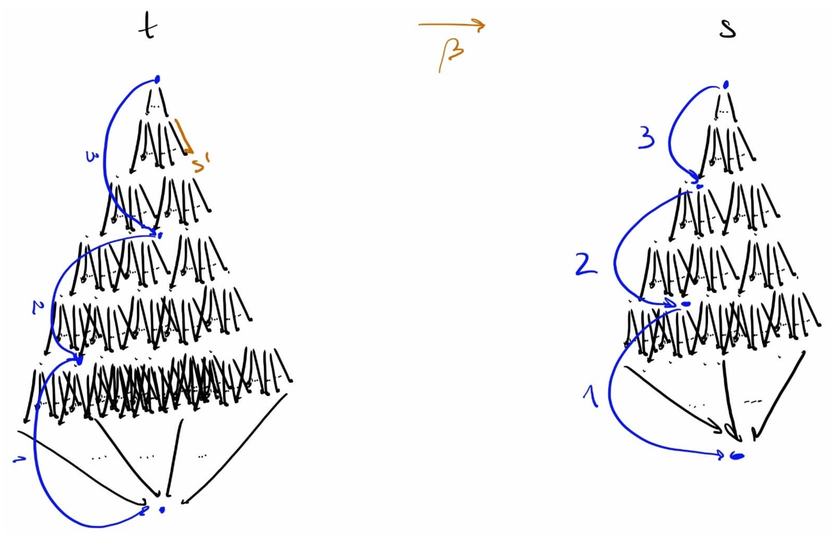
La idea de las memorias no es modificar estado, sino **evitar olvidar** partes del programa



Definimos operaciones recursivas estructurales para "movernos" en el árbol, pudiendo observar partes sin necesidad de calcularlo completo



- el cálculo original olvida pero el cálculo con memorias no
- el árbol original sin memorias está parcialmente embebido en el denso
- hay nodos en el denso que no pueden recrearse luego de un paso de computación en el cálculo original
- nuestra medida es la función que encuentra esos nodos y suma la cantidad



IV. Reducibilidad módulo isomorfismos

trabajo en progreso con Alejandro Díaz-Caro

Tipos isomorfos
- transformación de programas de un tipo a otro
- sin pérdida de información

$$A \Rightarrow (B \wedge C) \equiv (A \Rightarrow B) \wedge (A \Rightarrow C)$$
$$\forall X.(A \Rightarrow B) \equiv A \Rightarrow \forall X.B$$
$$\lambda x^A.\ \langle r, s\rangle \rightleftarrows \langle \lambda x^A.\ r, \lambda x^A.\ s\rangle$$
$$\Lambda X.\lambda x^A.r \rightleftarrows \lambda x^A.\Lambda X.r$$
$$(\lambda x^A.\ r)[B] \rightleftarrows \lambda x^A.\ r[B]$$

λ-cálculo polimórfico módulo isomorfismos
- equivalencia por isomorfismo en los tipos
- equivalencia por "correspondencia" en los programas

los correspondientes son aquellos de sus tipos isomorfos a los que se pueden transformar

Estamos trabajando en su prueba de terminación

El desafío de reducibilidad ante isomorfismos
- la equivalencia en la interpretación
- la equivalencia en la parametrización
- la explosión de casos en la reconstrucción del árbol

Nuestra propuesta
- **Reducibilidad à la Parigot**
  introducción de estructura en la familia de parametrización (candidatos de reducibilidad) + caracterización de interpretaciones según totalidad de eliminadores

$$\begin{aligned} \textstyle\prod_{i=1}^n F_i &= \{t \in \mathcal{T}_{A_1\times\ldots\times A_n} \mid \forall i \in 1..n.\ \pi_{A_i} t \in F_i\} && \text{with } F_i \subseteq \mathcal{T}_{A_i} \\ F \tilde{\to} G &= \{t \in \mathcal{T}_{A\to B} \mid \forall u \in F.\ ts \in G\} && \text{with } F \subseteq \mathcal{T}_A \text{ and } G \subseteq \mathcal{T}_B \\ \tilde{\forall}\mathbb{B}.U_\mathbb{B} &= \{t \in \mathcal{T}_{\forall X.A} \mid \forall \mathbb{B} \in \mathbb{K}.\ t\mathbb{B} \in U_\mathbb{B}\} && \text{with } (U_\mathbb{B})_{\mathbb{B}\in\mathbb{K}} \text{ family s.t.} \\ & && U_\mathbb{B} \subseteq \mathcal{T}_{A[\mathbb{B}/X]} \text{ for all } \mathbb{B} \in \mathbb{K} \\ S \tilde{\Rightarrow}_A G &= \{t \in \mathcal{T}_A \mid \forall \vec{u} \in S.\ t\vec{u} \in G\} && \text{with } S \subseteq (\mathcal{T}_* \cup \mathcal{F} \cup \{\overline{\pi_B}\})^{<\omega} \text{ and } G \subseteq \mathcal{T} \end{aligned}$$

- **Candidatos nuevos** a partir de la intersección de candidatos de tipos isomorfos (equivalencia en la parametrización)

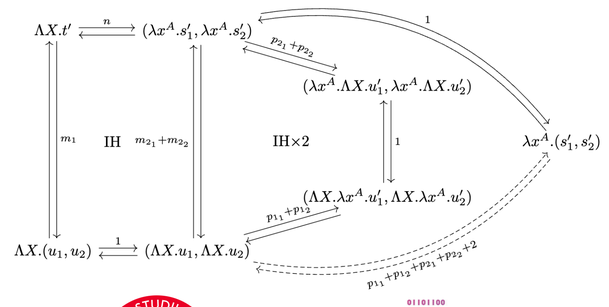
$$\frac{}{\mathcal{SN}_A \in \mathcal{R}_A} \qquad \frac{F \in \mathcal{R}_A \quad G \in \mathcal{R}_B}{F \tilde{\times} G \in \mathcal{R}_{A\times B}} \qquad \frac{U \in \mathcal{R}_A \quad V \in \mathcal{R}_B}{U \tilde{\to} V \in \mathcal{R}_{A\to B}}$$

$$\frac{X \subseteq \mathcal{R}_A}{\bigcap X \in \mathcal{R}_A} \qquad \frac{F \in \mathcal{R}_A \quad G \in \mathcal{R}_B \quad A \equiv B}{F \cap G \in \mathcal{R}_A} \qquad \frac{U_B \in \mathcal{R}_{A[B/X]}\ (\text{for all } B \in \mathcal{F})}{\tilde{\forall} B.U_B \in \mathcal{R}_{\forall X.A}}$$

- **Interpretación de tipos compuesta**: múltiples condiciones simultáneas, basadas en las múltiples formas posibles de los programas isomorfos (equivalencia en la interpretación)

$t \in [\![A]\!]_\rho$ *if and only if:*
- *if* $A \equiv X$, *then* $t \in \rho(X)$
- *if* $A \equiv B \times C$, *then* $t \in [\![B]\!]_\rho \tilde{\times} [\![C]\!]_\rho$
- *if* $A \equiv B \to C$, *then* $t \in [\![B]\!]_\rho \tilde{\to} [\![C]\!]_\rho$
- *if* $A \equiv \forall X.B$, *then* $t \in \tilde{\forall} C. \bigcap \{[\![B]\!]_{\rho\cdot[F/X]} | F \in \mathcal{R}_C\}$

- **Caracterización de** todos los **posibles pasos de computación** que pueden aparecer en un programa **al transformar subprogramas** en isomorfos (explosión de casos)



DEPARTAMENTO DE COMPUTACION — Facultad de Ciencias Exactas y Naturales - UBA — EXACTAS UBA — UBA Universidad de Buenos Aires — *Argentina virtus robur et studium* — CONICET — Universidad Nacional de Quilmes — Universitas Studii Taurinensis 1404 — Università di Torino — Loria Laboratoire lorrain de recherche en informatique et ses applications — Inria