Cálculo Lambda Tipado (2)

Resultados básicos

Unicidad de tipos

Si $\Gamma \rhd M : \sigma$ y $\Gamma \rhd M : \tau$ son derivables, entonces $\sigma = \tau$

Resultados básicos

Unicidad de tipos

Si $\Gamma \triangleright M : \sigma$ y $\Gamma \triangleright M : \tau$ son derivables, entonces $\sigma = \tau$

Weakening+Strengthening

Resultados básicos

Unicidad de tipos

Si $\Gamma \triangleright M : \sigma$ y $\Gamma \triangleright M : \tau$ son derivables, entonces $\sigma = \tau$

Weakening+Strengthening

Si $\Gamma \triangleright M : \sigma$ es derivable y $\Gamma \cap \Gamma'$ contiene a todas las variables libres de M, entonces $\Gamma' \triangleright M : \sigma$

Propiedades

Lema (Determinismo del juicio de evaluación en un paso)

Si $M \to M'$ y $M \to M''$, entonces $M' = M''$

Propiedades

Una forma normal es un término que no puede evaluarse más (i.e. M tal que no existe N, $M \to N$)

Propiedades

Una forma normal es un término que no puede evaluarse más (i.e. M tal que no existe N, $M \to N$)

Lema (Unicidad de formas normales)

Si $M \twoheadrightarrow U$ y $M \twoheadrightarrow V$ con U, V formas normales, entonces $U = V$

Propiedades

Una forma normal es un término que no puede evaluarse más (i.e. M tal que no existe N, $M \to N$)

Lema (Unicidad de formas normales)

Si $M \twoheadrightarrow U$ y $M \twoheadrightarrow V$ con U, V formas normales, entonces $U = V$

Lema (Terminación)

Para todo M existe una forma normal N tal que $M \twoheadrightarrow N$

Propiedades

Una forma normal es un término que no puede evaluarse más (i.e. M tal que no existe N, $M \to N$)

Lema (Unicidad de formas normales)

Si $M \twoheadrightarrow U$ y $M \twoheadrightarrow V$ con U, V formas normales, entonces $U = V$

Lema (Terminación)

Para todo M existe una forma normal N tal que $M \twoheadrightarrow N$

Lema

Todo valor está en forma normal

Propiedades

Una forma normal es un término que no puede evaluarse más (i.e. M tal que no existe N, $M \to N$)

Lema (Unicidad de formas normales)

Si $M \twoheadrightarrow U$ y $M \twoheadrightarrow V$ con U, V formas normales, entonces $U = V$

Lema (Terminación)

Para todo M existe una forma normal N tal que $M \twoheadrightarrow N$

Lema

Todo valor está en forma normal

- No vale el recíproco en λ^b (pero sí vale en el cálculo de las expresiones booleanas cerradas):
  - $\mathit{if}\ x\ \mathit{then}\ \mathit{true}\ \mathit{else}\ \mathit{false}$
  - x
  - $\mathit{true}\ \mathit{false}$

Objetivo de un sistema de tipos

Garantizar la ausencia de errores

Estado de error

- Un forma normal que no es un valor
- Representa un estado (= término) que no es un valor pero en el que la evaluación está trabada
- Representa estado en el cual el sistema de run-time en una implementación real generaría una excepción

Ejemplos

- *if x then M else N*
  - Obs: no es cerrado
  - Un término M es cerrado si $FV(M) = \emptyset$
- *true M*
  - Obs: no es tipable

Corrección

Corrección $=$ Progreso $+$ Preservación

Progreso

Si M es cerrado y bien tipado entonces

1. M es un valor
2. o bien existe M' tal que $M \to M'$

La evaluación no puede trabarse para términos cerrados, bien tipados que no son valores

Preservación

Si $\Gamma \triangleright M : \sigma$ y $M \to N$, entonces $\Gamma \triangleright N : \sigma$

La evaluación preserva tipos

Tipos y términos de λ^{bn}

$$\sigma \quad ::= \quad Bool \mid Nat \mid \sigma \to \rho$$

$$M \quad ::= \quad \ldots \mid 0 \mid succ(M) \mid pred(M) \mid iszero(M)$$

Descripción informal:

- $succ(M)$: evaluar M hasta arrojar un número e incrementarlo
- $pred(M)$: evaluar M hasta arrojar un número y decrementarlo
- $iszero(M)$: evaluar M hasta arrojar un número, luego retornar $true/false$ según sea cero o no

Tipado de λ^{bn}

Agregamos a los axiomas y regla de tipado de λ^b los siguientes:

$$\frac{}{\Gamma \rhd 0 : \mathit{Nat}}\ (\textsc{T-Zero})$$

$$\frac{\Gamma \rhd M : \mathit{Nat}}{\Gamma \rhd \mathit{succ}(M) : \mathit{Nat}}\ (\textsc{T-Succ}) \qquad \frac{\Gamma \rhd M : \mathit{Nat}}{\Gamma \rhd \mathit{pred}(M) : \mathit{Nat}}\ (\textsc{T-Pred})$$

$$\frac{\Gamma \rhd M : \mathit{Nat}}{\Gamma \rhd \mathit{iszero}(M) : \mathit{Bool}}\ (\textsc{T-IsZero})$$

Valores y evaluación en un paso de λ^{bn} (1/2)

Valores

$$V \quad ::= \quad \ldots \mid \underline{n} \quad \text{donde } \underline{n} \text{ abrevia } succ^{n}(0).$$

Juicio de evaluación en un paso (1/2)

$$\frac{M_1 \to M_1'}{succ(M_1) \to succ(M_1')}\ (\textsc{E-Succ})$$

$$\frac{}{pred(0) \to 0}\ (\textsc{E-PredZero}) \qquad \frac{}{pred(\underline{n+1}) \to \underline{n}}\ (\textsc{E-PredSucc})$$

$$\frac{M_1 \to M_1'}{pred(M_1) \to pred(M_1')}\ (\textsc{E-Pred})$$

Valores y evaluación en un paso de $\lambda^{bn}(2/2)$

Juicio de evaluación en un paso (2/2)

$$\frac{}{\mathit{iszero}(0) \rightarrow \mathit{true}}\ (\text{E-IsZeroZero})$$

$$\frac{}{\mathit{iszero}(\underline{n+1}) \rightarrow \mathit{false}}\ (\text{E-IsZeroSucc})$$

$$\frac{M_1 \rightarrow M_1'}{\mathit{iszero}(M_1) \rightarrow \mathit{iszero}(M_1')}\ (\text{E-IsZero})$$

Además de los juicios de evaluación en un paso de $\mathrm{C}\text{-}\lambda^b$.

Tipos y términos de $\lambda^{\cdots r}$

Sea $\mathcal{L}$ un conjunto de etiquetas

$$\sigma \quad ::= \quad \ldots \mid \{l_i : \sigma_i \ ^{i \in 1..n}\}$$

Tipos y términos de $\lambda^{\cdots r}$

Sea $\mathcal{L}$ un conjunto de etiquetas

$$\sigma \quad ::= \quad \ldots \mid \{l_i : \sigma_i \ ^{i \in 1..n}\}$$

- $\{nombre : String, edad : Nat\}$
- $\{persona : \{nombre : String, edad : Nat\}, cuil : Nat\}$

$$\{nombre : String, edad : Nat\} \neq \{edad : Nat, nombre : String\}$$

Tipos y términos de $\lambda^{\cdots r}$

$$M \quad ::= \quad \ldots \mid \{l_i = M_i \ ^{i \in 1..n}\} \mid M.l$$

Descripción informal:

- El registro $\{l_i = M_i \ ^{i \in 1..n}\}$ evalúa a $\{l_i = V_i \ ^{i \in 1..n}\}$ donde V_i es el valor al que evalúa M_i, $i \in 1..n$
- $M.l$: evaluar M hasta que arroje $\{l_i = V_i \ ^{i \in 1..n}\}$, luego proyectar el campo correspondiente

Ejemplos

- $\lambda x : Nat.\lambda y : Bool.\{edad = x, esMujer = y\}$
- $\lambda p : \{edad : Nat, esMujer : Bool\}.p.edad$
- $(\lambda p : \{edad : Nat, esMujer : Bool\}.p.edad)$
  $\{edad = 20, esMujer = false\}$

Tipado de $\lambda^{\cdots r}$

$$\frac{\Gamma \rhd M_i : \sigma_i \quad \text{para cada } i \in 1..n}{\Gamma \rhd \{l_i = M_i \;^{i \in 1..n}\} : \{l_i : \sigma_i \;^{i \in 1..n}\}}\ (\text{T-Rcd})$$

$$\frac{\Gamma \rhd M : \{l_i : \sigma_i \;^{i \in 1..n}\} \quad j \in 1..n}{\Gamma \rhd M.l_j : \sigma_j}\ (\text{T-Proj})$$

Semántica operacional de $\lambda^{\cdots r}$

Valores

$$V \quad ::= \quad \ldots \mid \{l_i = V_i \ ^{i \in 1..n}\}$$

Semántica operacional de $\lambda^{\cdots r}$

$$\frac{M_j \to M_j'}{\begin{array}{c}\{l_i = V_i \ ^{i \in 1..j-1}, l_j = M_j, l_i = M_i \ ^{i \in j+1..n}\} \\ \to \\ \{l_i = V_i \ ^{i \in 1..j-1}, l_j = M_j', l_i = M_i \ ^{i \in j+1..n}\}\end{array}} \text{ (E-RCD)}$$

$$\frac{j \in 1..n}{\{l_i = V_i \ ^{i \in 1..n}\}.l_j \to V_j} \text{ (E-PROJRCD)}$$

$$\frac{M \to M'}{M.l \to M'.l} \text{ (E-PROJ)}$$

Tipos y términos de λ^{bnu}

$$\begin{array}{rcl} \sigma & ::= & \mathit{Bool} \mid \mathit{Nat} \mid \mathit{Unit} \mid \sigma \to \rho \\ M & ::= & \ldots \mid \mathit{unit} \end{array}$$

Descripción informal:

- *Unit* es un tipo unitario y el único valor posible de una expresión de ese tipo es *unit*.
- Cumple rol similar a *void* en C o Java

Tipado de λ^{bnu}

Agregamos el axioma de tipado:

$$\frac{}{\Gamma \triangleright \mathit{unit} : \mathit{Unit}} \ (\text{T-U}{\small\text{NIT}})$$

NB:

- No hay reglas de evaluación nuevas
- Extendemos el conjunto de valores V con *unit*

$$V \quad ::= \quad \dots \mid \mathit{unit}$$

Utilidad

- Su utilidad principal es en lenguajes con efectos laterales
- En estos lenguajes es útil poder evaluar varias expresiones en secuencia

$$M_1; M_2 \stackrel{\text{def}}{=} (\lambda x : Unit.M_2)\, M_1 \qquad x \notin FV(M_2)$$

  - La evaluación de $M_1; M_2$ consiste en primero evaluar M_1 y luego M_2
  - Con la definición dada, este comportamiento se logra con las reglas de evaluación definidas previamente

Tipos y términos de $\lambda^{\ldots let}$

$$M \quad ::= \quad \ldots \mid let\ x : \sigma = M\ in\ N$$

Descripción informal:

- $let\ x : \sigma = M\ in\ N$: evaluar M a un valor V, ligar x a V y evaluar N
- Mejora la legibilidad
- La extensión con *let* no implica agregar nuevos tipos

Ejemplo

- $let\ x : Nat = \underline{2}\ in\ succ(x)$
- $pred\ (let\ x : Nat = \underline{2}\ in\ x)$
- $let\ f : Nat \rightarrow Nat = \lambda x : Nat.succ(n)\ in\ f(f0))$
- $let\ x : Nat = \underline{2}\ in\ let\ x : Nat = \underline{3}\ in\ x$

Semántica operacional de $\lambda^{\ldots let}$

$$\frac{M_1 \to M_1'}{let\ x : \sigma = M_1\ in\ M_2 \to let\ x : \sigma = M_1'\ in\ M_2}\ (\text{E-Let})$$

$$\frac{}{let\ x : \sigma = V_1\ in\ M_2 \to M_2\{x \leftarrow V_1\}}\ (\text{E-LetV})$$

Recursión

Ecuación recursiva

$$f = \ldots f \ldots f \ldots$$

Términos y tipado

$$M \quad ::= \quad \ldots \mid \mathsf{fix}\ \ M$$

- No se precisan nuevos tipos pero sí una regla de tipado.

$$\frac{\Gamma \triangleright M : \sigma_1 \to \sigma_1}{\Gamma \triangleright \mathsf{fix}\ \ M : \sigma_1} \,(\text{T-Fix})$$

Semántica operacional small-step

No hay valores nuevos pero sí reglas de evaluación en un paso nuevas.

$$\frac{M_1 \to M_1'}{\mathsf{fix}\ \ M_1 \to \mathsf{fix}\ \ M_1'}\ (\text{E-}\textsc{Fix})$$

$$\frac{}{\mathsf{fix}\ (\lambda x : \sigma.M) \to M\{x \leftarrow \mathsf{fix}\ (\lambda x : \sigma.M)\}}\ (\text{E-}\textsc{FixBeta})$$

Ejemplos

Sea M el término

$$\begin{array}{l}\lambda f : Nat \to Nat. \\ \ \lambda x : Nat. \\ \ \ if \ \ \text{iszero}(x) \ then \ \underline{1} \ else \ x * f(\text{pred}(x))\end{array}$$

en

$$let \ \ fact = \text{fix} \ \ M \ \ in \ \ fact \, \underline{3}$$

Ejemplos

Ahora podemos definir funciones parciales:

$\text{fix}\,(\lambda x : Nat.succ\; x)$

Ejemplos

Sea M el término

$$\begin{array}{l}\lambda s : Nat \rightarrow Nat \rightarrow Nat. \\ \ \lambda x : Nat. \\ \ \ \lambda y : Nat. \\ \ \ \ \mathit{if}\ \ \mathrm{iszero}(x)\ \mathit{then}\ y\ \mathit{else}\ \ \mathrm{succ}(s\ \mathrm{pred}(x)\ y)\end{array}$$

en

$$\mathit{let}\ \mathit{suma} = \mathrm{fix}\ M\ \ \mathit{in}\ \mathit{suma}\, \underline{2}\, \underline{3}$$

Letrec

Una construcción alternativa para definir funciones recursivas es

$$letrec\ f : \sigma \to \sigma = \lambda x : \sigma.M\ in\ N$$

Por ejemplo,

$$\begin{array}{l} letrec \\ \ fact : Nat \to Nat = \\ \lambda x : Nat.if\ \ iszero(x)\ then\ \underline{1}\ else\ x * fact(pred(x)) \\ \ \ in\ fact\,\underline{3} \end{array}$$

letrec puede escribirse en términos de fix del siguiente modo:

$$let\ f = \mathrm{fix}\,(\lambda f : \sigma \to \sigma.\lambda x : \sigma.M)\ in\ N$$