Cálculo Lambda Tipado (2/3)

Departamento de Computación, FCEyN, UBA

7 Febrero 2018

Mas extensiones

- Registros
- Declaraciones locales
- Referencias

  $$\text{Programación Imperativa} = \text{Progr. Funcional} + \text{Efectos}$$

- Recursión

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 $\lambda^{\cdots r}$

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

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

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

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

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

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

- $\mathit{let}\ x : \mathit{Nat} = \underline{2}\ \mathit{in}\ \mathit{succ}(x)$
- $\mathit{pred}\ (\mathit{let}\ x : \mathit{Nat} = \underline{2}\ \mathit{in}\ x)$
- $\mathit{let}\ x : \mathit{Nat} = \underline{2}\ \mathit{in}\ \mathit{let}\ x : \mathit{Nat} = \underline{3}\ \mathit{in}\ x$

Tipado de $\lambda^{\ldots let}$

$$\frac{\Gamma \rhd M : \sigma_1 \quad \Gamma, x : \sigma_1 \rhd N : \sigma_2}{\Gamma \rhd \mathit{let}\ x : \sigma_1 = M\ \mathit{in}\ N : \sigma_2}\ (\text{T-Let})$$

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

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

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

Referencias - Motivación

- En una expresión como $let\ x : Nat = \underline{2}\ in\ M$
  - x es una variable declarada con valor 2.
  - El valor de x permanece inalterado a lo largo de la evaluación de M.
  - En este sentido x es inmutable: no existe una operación de asignación.
- En programación imperativa pasa todo lo contrario.
  - Todas las variables son mutables.
- Vamos a extender Cálculo Lambda Tipado con variables mutables.

Operaciones básicas

Asignación

$x := M$ almacena en la referencia x el valor de M.

Alocación (Reserva de memoria)

ref M genera una referencia fresca cuyo contenido es el valor de M.

Derreferenciación (Lectura)

$!x$ sigue la referencia x y retorna su contenido.

Comandos = Expresiones con efectos

- El término $let\ x = ref\ \underline{2}\ in\ x := succ(!x)$, ¿A qué evalúa?
- La asignación es una expresión que interesa por su efecto y no su valor.
  - Carece de sentido preguntarse por el valor de una asignación.
  - ¡Sí tiene sentido preguntarse por el efecto!

Comando

Expresión que se evalúa para causar un efecto; lo expresamos con un nuevo valor *unit*.

- Un lenguaje funcional puro es uno en el que las expresiones son puras en el sentido de carecer de efectos.

Tipos y términos de λ^{bnu}

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

$$M \quad ::= \quad \ldots \mid unit$$

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:

$$\frac{}{\Gamma \rhd \mathit{unit} : \mathit{Unit}}\ (\text{T-UNIT})$$

NB:

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

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

Utilidad

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

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

  - 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

Expresiones de tipos

Las expresiones de tipos se extienden del siguiente modo

$$\sigma \quad ::= \quad \mathit{Bool} \mid \mathit{Nat} \mid \sigma \to \tau \mid \mathit{Unit} \mid \mathit{Ref}\ \sigma$$

Descripción informal:

- *Ref* σ es el tipo de las referencias a valores de tipo σ.
- Ej. *Ref* (*Bool* $\to$ *Nat*) es el tipo de las referencias a funciones de *Bool* en *Nat*.

Términos

$$
\begin{array}{rcl}
M & ::= & x \\
 & | & \lambda x : \sigma . M \\
 & | & M\ N \\
 & | & unit \\
 & | & ref\ \ M \\
 & | & !M \\
 & | & M := N \\
 & | & \ldots
\end{array}
$$

El sistema de tipado excluirá términos "mal formados".

- $!\underline{2}$
- $\underline{2} := \underline{3}$

Reglas de tipado - Preliminares

$$\frac{\Gamma \rhd M_1 : \sigma}{\Gamma \rhd \mathit{ref}\ M_1 : \mathit{Ref}\ \sigma}\ (\text{T-Ref})$$

$$\frac{\Gamma \rhd M_1 : \mathit{Ref}\ \sigma}{\Gamma \rhd !M_1 : \sigma}\ (\text{T-DeRef})$$

$$\frac{\Gamma \rhd M_1 : \mathit{Ref}\ \sigma_1 \quad \Gamma \rhd M_2 : \sigma_1}{\Gamma \rhd M_1 := M_2 : \mathit{Unit}}\ (\text{T-Assign})$$

Motivación

Al intentar formalizar la semántica operacional surgen las preguntas:

- ¿Cuáles son los valores de tipo $Ref\ \sigma$?
- ¿Cómo modelar la evaluación del término $ref\ M$?

Las respuestas dependen de otra pregunta.

¿Qué es una referencia?

Rta. Es una abstracción de una porción de memoria que se encuentra en uso.

Memoria o "store"

- Usamos direcciones (simbólicas) o "locations" $l, l_i \in \mathcal{L}$ para representar referencias.

  Memoria (o "store"): función parcial de direcciones a valores.

- Usamos letras μ, μ' para referirnos a stores.
- Notación:
  - $\mu[l \mapsto V]$ es el store resultante de pisar $\mu(l)$ con V.
  - $\mu \oplus (l \mapsto V)$ es el store extendido resultante de ampliar μ con una nueva asociación $l \mapsto V$ (asumimos $l \notin Dom(\mu)$).

Los juicios de evaluación toman la forma:

$$M \mid \mu \to M' \mid \mu'$$

Valores

Intuición:

$$\frac{l \notin Dom(\mu)}{ref\ V \mid \mu \to l \mid \mu \oplus (l \mapsto V)}\ (\text{E-RefV})$$

Los valores posibles ahora incluyen las direcciones.

$$V \quad ::= \quad unit \mid \lambda x : \sigma.M \mid l$$

Dado que los valores son un subconjunto de los términos,

- debemos ampliar los términos con direcciones;
- éstas son producto de la formalización y no se pretende que sean utilizadas por el programador.

Términos extendidos

$$
\begin{array}{rcl}
M & ::= & x \\
 & | & \lambda x : \sigma . M \\
 & | & M\ N \\
 & | & unit \\
 & | & ref\ \ M \\
 & | & !M \\
 & | & M := N \\
 & | & l \\
 & | & \ldots
\end{array}
$$

Juicios de tipado

$$\Gamma \rhd l : \mathbf{?}$$

- Depende de los valores que se almacenen en la dirección l.
- Situación parecida a las variables libres.
- Precisamos un "contexto de tipado" para direcciones:
  - Σ función parcial de direcciones en tipos.

Nuevo juicio de tipado

$$\Gamma | \Sigma \rhd M : \sigma$$

Reglas de tipado - Definitivas

$$\frac{\Gamma|\Sigma \rhd M_1 : \sigma}{\Gamma|\Sigma \rhd \mathit{ref}\ M_1 : \mathit{Ref}\ \sigma}\ (\text{T-Ref})$$

$$\frac{\Gamma|\Sigma \rhd M_1 : \mathit{Ref}\ \sigma}{\Gamma|\Sigma \rhd !M_1 : \sigma}\ (\text{T-DeRef})$$

$$\frac{\Gamma|\Sigma \rhd M_1 : \mathit{Ref}\ \sigma_1 \quad \Gamma|\Sigma \rhd M_2 : \sigma_1}{\Gamma|\Sigma \rhd M_1 := M_2 : \mathit{Unit}}\ (\text{T-Assign})$$

$$\frac{\Sigma(l) = \sigma}{\Gamma|\Sigma \rhd l : \mathit{Ref}\ \sigma}\ (\text{T-Loc})$$

Juicios de evaluación en un paso

$$M \mid \mu \to M' \mid \mu'$$

- Recordar el conjunto de valores (expresiones resultantes de evaluar por completo a términos cerrados y bien tipados).

$$V \quad ::= \quad \mathit{true} \mid \mathit{false} \mid 0 \mid \underline{n} \mid \mathit{unit} \mid \lambda x : \sigma . M \mid \mathit{l}$$

Juicios de evaluación en un paso (1/4)

$$\frac{M_1 \mid \mu \to M_1' \mid \mu'}{M_1\ M_2 \mid \mu \to M_1'\ M_2 \mid \mu'}\ (\text{E-App1})$$

$$\frac{M_2 \mid \mu \to M_2' \mid \mu'}{V_1\ M_2 \mid \mu \to V_1\ M_2' \mid \mu'}\ (\text{E-App2})$$

$$\frac{}{(\lambda x : \sigma.M)\ V \mid \mu \to M\{x \leftarrow V\} \mid \mu}\ (\text{E-AppAbs})$$

Nota: Estas reglas no modifican el store.

Juicios de evaluación en un paso (2/4)

$$\frac{M_1 \mid \mu \to M_1' \mid \mu'}{!M_1 \mid \mu \to !M_1' \mid \mu'} \text{ (E-Deref)}$$

$$\frac{\mu(l) = V}{!l \mid \mu \to V \mid \mu} \text{ (E-DerefLoc)}$$

Juicios de evaluación en un paso (3/4)

$$\frac{M_1 \mid \mu \to M_1' \mid \mu'}{M_1 := M_2 \mid \mu \to M_1' := M_2 \mid \mu'} \text{(E-Assign1)}$$

$$\frac{M_2 \mid \mu \to M_2' \mid \mu'}{V := M_2 \mid \mu \to V := M_2' \mid \mu'} \text{(E-Assign2)}$$

$$\frac{}{l := V \mid \mu \to \mathit{unit} \mid \mu[l \mapsto V]} \text{(E-Assign)}$$

Juicios de evaluación en un paso (4/4)

$$\frac{M_1 \mid \mu \to M_1' \mid \mu'}{\mathit{ref}\ M_1 \mid \mu \to \mathit{ref}\ M_1' \mid \mu'}\ (\text{E-}\textsc{Ref})$$

$$\frac{l \notin \mathit{Dom}(\mu)}{\mathit{ref}\ V \mid \mu \to l \mid \mu \oplus (l \mapsto V)}\ (\text{E-}\textsc{RefV})$$

Ejemplo

$$
\begin{array}{ll}
 & \mathit{let}\ x = \mathit{ref}\ \underline{2}\ \mathit{in}\ (\lambda\_ : \mathit{Unit}.!x)\,(x := \mathit{succ}(!x)) \mid \mu \\
\to & \mathit{let}\ x = l_1\ \mathit{in}\ (\lambda\_ : \mathit{Unit}.!x)\,(x := \mathit{succ}(!x)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda\_ : \mathit{Unit}.!l_1)\,(l_1 := \mathit{succ}(!l_1)) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda\_ : \mathit{Unit}.!l_1)\,(l_1 := \mathit{succ}(\underline{2})) \mid \mu \oplus (l_1 \mapsto \underline{2}) \\
\to & (\lambda\_ : \mathit{Unit}.!l_1)\,\mathit{unit} \mid (\mu \oplus (l_1 \mapsto \underline{2}))[l_1 \mapsto \underline{3}] \\
\to & !l_1 \mid \mu \oplus (l_1 \mapsto \underline{3}) \\
\to & \underline{3} \mid \mu \oplus (l_1 \mapsto \underline{3})
\end{array}
$$

Ejemplo

Sea

$$M \quad = \quad \lambda r : Ref\,(Unit \to Unit).\; let\ f = !r\ in\ (r := \lambda x : Unit.f\ x); (!r)\ unit$$

$$
\begin{array}{ll}
 & M\,(ref\ (\lambda x : Unit.x)) \mid \mu \\
\to & M\,l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.x) \\
\to & let\ f = !l_1\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & let\ f = \lambda x : Unit.x\ in\ (l_1 := \lambda x : Unit.f\ x); (!l_1)\ unit \mid \ldots \\
\to & (l_1 := \lambda x : Unit.(\lambda x : Unit.x)\,x); (!l_1)\ unit \mid \ldots \\
\to & unit; (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\,x) \\
\to & (!l_1)\ unit \mid \mu \oplus (l_1 \mapsto \lambda x : Unit.(\lambda x : Unit.x)\,x) \\
\to & (\lambda x : Unit.(\lambda x : Unit.x)\,x)\ unit \mid \ldots \\
\to & (\lambda x : Unit.x)\ unit \mid \ldots \\
\to & unit \mid \ldots
\end{array}
$$

Ejemplo

Sea

$$\begin{aligned} M \quad = \quad & \lambda r : \mathit{Ref}\,(\mathit{Unit} \to \mathit{Unit}). \\ & \mathit{let}\ f = !r \\ & \mathit{in}\ (r := \lambda x : \mathit{Unit}.f\ x); (!r)\ \mathit{unit} \end{aligned}$$

Reemplazamos f por $!r$ y nos queda

$$\begin{aligned} M' \quad = \quad & \lambda r : \mathit{Ref}\ (\mathit{Unit} \to \mathit{Unit}). \\ & (r := \lambda x : \mathit{Unit}.(!r)\ x); (!r)\ \mathit{unit} \end{aligned}$$

Vamos a evaluar este nuevo M' aplicado al mismo término que en el slide anterior y ver qué pasa...

Ejemplo

$$M' \quad = \quad \lambda r : \mathit{Ref}\ (\mathit{Unit} \to \mathit{Unit}).\ (r := \lambda x : \mathit{Unit}.(!r)\, x); (!r)\, \mathit{unit}$$

$$\begin{array}{ll}
 & M'\,(\mathit{ref}\ (\lambda x : \mathit{Unit}.x)) \mid \mu \\
\to & M'\, l_1 \mid \mu \oplus (l_1 \mapsto \lambda x : \mathit{Unit}.x) \\
\to & (l_1 := \lambda x : \mathit{Unit}.(!l_1)\, x); (!l_1)\, \mathit{unit} \mid \ldots \\
\to & \mathit{unit}; (!l_1)\, \mathit{unit} \mid \mu \oplus (l_1 \mapsto \lambda x : \mathit{Unit}.(!l_1)\, x) \\
\to & \boxed{(!l_1)\, \mathit{unit}} \mid \ldots \\
\to & (\lambda x : \mathit{Unit}.(!l_1)\, x)\, \mathit{unit} \mid \ldots \\
\to & \boxed{(!l_1)\, \mathit{unit}} \mid \ldots \\
\to & \ldots
\end{array}$$

Nota: no todo término cerrado y bien tipado termina en λ^{bnr} (λ-cálculo con booleanos, naturales y referencias).

La clase pasada - Corrección de sistema de tipos

$$\text{Corrección} = \text{Progreso} + \text{Preservación}$$

Progreso

Si M es cerrado y bien tipado entonces

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

Preservación

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

Debemos reformular estos resultados en el marco de referencias.

Preservación - Formulación ingenua

La formulación ingenua siguiente es errónea:

$$\Gamma|\Sigma \rhd M : \sigma \text{ y } M \,|\, \mu \to M' \,|\, \mu' \text{ implica } \Gamma|\Sigma \rhd M' : \sigma$$

- El problema: puede que la semántica no respete los tipos asumidos por el sistema de tipos para las direcciones (i.e. Σ).
- Vamos a ver un ejemplo concreto.

Preservación - Formulación ingenua

$$\Gamma|\Sigma \rhd M : \sigma \text{ y } M \,|\, \mu \to M' \,|\, \mu' \text{ implica } \Gamma|\Sigma \rhd M' : \sigma$$

Supongamos que

- $M = !I$
- $\Gamma = \emptyset$
- $\Sigma(I) = \mathit{Nat}$
- $\mu(I) = \mathit{true}$

Observar que

- $\Gamma|\Sigma \rhd M : \mathit{Nat}$ y
- $M \,|\, \mu \to \mathit{true} \,|\, \mu$
- pero $\Gamma|\Sigma \rhd \mathit{true} : \mathit{Nat}$ no vale.

Preservación - Formulación ingenua

Formulación ingenua

$$\Gamma|\Sigma \rhd M : \sigma \text{ y } M \,|\, \mu \to M' \,|\, \mu' \text{ implica } \Gamma|\Sigma \rhd M' : \sigma$$

Supongamos que

- $M = !I$
- $\Gamma = \emptyset$
- $\Sigma(I) = \boxed{Nat}$
- $\mu(I) = \boxed{true}$

Observar que

- $\Gamma|\Sigma \rhd M : Nat$ y
- $M \,|\, \mu \to true \,|\, \mu$
- pero $\Gamma|\Sigma \rhd true : Nat$ no vale.

Preservación - Reformulada

- Precisamos una noción de compatibilidad entre el store y el contexto de tipado para stores.
  - Debemos tipar los stores.
- Introducimos un nuevo juicio de tipado:

$$\Gamma | \Sigma \rhd \mu$$

- Este juicio se define del siguiente modo:

$$\Gamma | \Sigma \rhd \mu \text{ sii}$$

1. $Dom(\Sigma) = Dom(\mu)$ y
2. $\Gamma | \Sigma \rhd \mu(l) : \Sigma(l)$ para todo $l \in Dom(\mu)$.

Preservación - Reformulada

Reformulamos preservación del siguiente modo.

$$\text{Si } \Gamma|\Sigma \rhd M : \sigma \quad \text{y} \quad M \,|\, \mu \to N \,|\, \mu' \quad \text{y} \quad \Gamma|\Sigma \rhd \mu,$$

$$\text{entonces } \Gamma|\boxed{\Sigma} \rhd N : \sigma.$$

- Esto es casi correcto.
- No contempla la posibilidad de que el Σ encuadrado haya crecido en dominio respecto a Σ.
  - Por posibles reservas de memoria.

Preservación - Definitiva

Si

- $\Gamma|\Sigma \rhd M : \sigma$
- $M \,|\, \mu \to N \,|\, \mu'$
- $\Gamma|\Sigma \rhd \mu$

implica que existe $\Sigma' \supseteq \Sigma$ tal que

- $\Gamma|\Sigma' \rhd N : \sigma$
- $\Gamma|\Sigma' \rhd \mu'$

Progreso - Reformulado

Si M es cerrado y bien tipado (i.e. $\emptyset|\Sigma \rhd M : \sigma$ para algún Σ, σ) entonces:

1. M es un valor
2. o bien para cualquier store μ tal que $\emptyset|\Sigma \rhd \mu$, existe M' y μ' tal que $M \,|\, \mu \to M' \,|\mu'$.

Recursión

Ecuación recursiva

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

Dos explicaciones de la función denotada (cuando existe).

- Denotacional
  - Límite de una cadena de aproximaciones
- Operacional
  - El "desdoblador" y puntos fijos

Términos y tipado

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

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

$$\frac{\Gamma \rhd M : \sigma_1 \to \sigma_1}{\Gamma \rhd \mathsf{fix}\ M : \sigma_1}\ (\text{T-FIX})$$

Semántica operacional small-step

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

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

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

Ejemplos

Sea M el término

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

en

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

Ejemplos

$$\mathsf{fix}\,(\lambda x : \mathit{Nat}.x + 1)$$

Ejemplos

Sea M el término

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

en

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

Letrec

Una construcción alternativa para definir funciones recursivas es

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

Por ejemplo,

$$\begin{array}{l}\mathit{letrec}\\ \quad \mathit{fact} : \mathit{Nat} \to \mathit{Nat} =\\ \qquad \lambda x : \mathit{Nat}.\mathit{if}\ x = 0\ \mathit{then}\ \underline{1}\ \mathit{else}\ x * \mathit{fact}(\mathit{pred}(x))\\ \mathit{in}\ \mathit{fact}\ \underline{3}\end{array}$$

letrec puede escribirse en términos de fix del siguiente modo:

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

Fin de la clase

La clase que viene...

...inferencia de tipos.