

Cálculo Lambda Tipado

¿Qué es el Cálculo Lambda?

- ▶ Modelo de computación basado en **funciones**
 - ▶ da origen a la programación funcional
- ▶ Introducido por **Alonzo Church** en 1936
- ▶ Computacionalmente completo (i.e. Turing completo)
- ▶ Vamos a estudiar el **Cálculo Lambda Tipado** (A. Church, 1940)

Expresiones de tipos de λ^b

Las expresiones de tipos (o simplemente tipos) de λ^b son

$$\sigma, \tau ::= \text{Bool} \mid \sigma \rightarrow \tau$$

Expresiones de tipos de λ^b

Las **expresiones de tipos** (o simplemente **tipos**) de λ^b son

$$\sigma, \tau ::= Bool \mid \sigma \rightarrow \tau$$

Descripción informal:

- ▶ *Bool* es el tipo de los booleanos,
- ▶ $\sigma \rightarrow \tau$ es el tipo de las funciones de σ en τ (donde σ y τ denotan expresiones de tipo)

Términos de λ^b

Sea \mathcal{X} un conjunto infinito enumerable de variables y $x \in \mathcal{X}$. Los **términos** de λ^b están dados por

$$\begin{array}{lcl} M, N, P, Q & ::= & x \\ & | & \textit{true} \\ & | & \textit{false} \\ & | & \textit{if } M \textit{ then } P \textit{ else } Q \\ & | & \lambda x : \sigma. M \\ & | & M N \end{array}$$

Ejemplos

- ▶ $\lambda x : \text{Bool}.x$
- ▶ $\lambda x : \text{Bool}.\text{if } x \text{ then false else true}$
- ▶ $\lambda f : \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}.\lambda x : \text{Bool}.f\ x$
- ▶ $(\lambda f : \text{Bool} \rightarrow \text{Bool}.f\ \text{true})(\lambda y : \text{Bool}.y)$
- ▶ $\text{true}(\lambda x : \text{Bool}.x)$
- ▶ $x\ y$

Tipado (intuición)

- ▶ Queremos asignar un tipo σ a un término M .

Tipado (intuición)

- ▶ Queremos asignar un tipo σ a un término M .
- ▶ Vamos a definir una relación de tipado “:”.

Tipado (intuición)

- ▶ Queremos asignar un tipo σ a un término M .
- ▶ Vamos a definir una relación de tipado “:”.
 - ▶ $true : Bool$,
 - ▶ $false : Bool$,
 - ▶ $\lambda x : Bool. x : Bool \rightarrow Bool, \dots$

Tipado (intuición)

- ▶ Queremos asignar un tipo σ a un término M .
- ▶ Vamos a definir una relación de tipado “:”.
 - ▶ $true : Bool$,
 - ▶ $false : Bool$,
 - ▶ $\lambda x : Bool. x : Bool \rightarrow Bool, \dots$
- ▶ El conjunto de términos es infinito, cómo definimos a la relación de tipado.

Sistema de tipado

Sistema de tipado

- Sistema formal de deducción (o derivación) que utiliza axiomas y reglas de inferencia para caracterizar un subconjunto de los términos llamados **tipados**.

Sistema de tipado

- ▶ Sistema formal de deducción (o derivación) que utiliza axiomas y reglas de inferencia para caracterizar un subconjunto de los términos llamados **tipados**.
- ▶ Definimos una relación de tipado a través de reglas de inferencia.
 - ▶ Los **axiomas de tipado** establecen que ciertos **juicios de tipado** son derivables.
 - ▶ Las **reglas de tipado** establecen que ciertos **juicios de tipado** son derivables siempre y cuando ciertos otros lo sean.

Términos de λ^b

Sea \mathcal{X} un conjunto infinito enumerable de variables y $x \in \mathcal{X}$. Los **términos** de λ^b están dados por

$$\begin{array}{lcl} M, N, P, Q & ::= & x \\ & | & \textit{true} \\ & | & \textit{false} \\ & | & \textit{if } M \textit{ then } P \textit{ else } Q \\ & | & \lambda x : \sigma. M \\ & | & M N \end{array}$$

Sistema de tipado

Sistema de tipado

Un **juicio de tipado** es una expresión de la forma $\Gamma \triangleright M : \sigma$ que se lee:

“el término M tiene tipo σ asumiendo el contexto de tipado Γ ”

Un **contexto de tipado** es un conjunto de pares $x_i : \sigma_i$, anotado $\{x_1 : \sigma_1, \dots, x_n : \sigma_n\}$ donde los $\{x_i\}_{i \in 1..n}$ son distintos. Usamos letras Γ, Δ, \dots para contextos de tipado.

Axiomas de tipado de λ^b

$$\frac{}{\Gamma \triangleright \text{true} : \text{Bool}} \text{ (T-TRUE)}$$

$$\frac{}{\Gamma \triangleright \text{false} : \text{Bool}} \text{ (T-FALSE)}$$

Axiomas de tipado de λ^b

$$\frac{}{\Gamma \triangleright \text{true} : \text{Bool}} \text{ (T-TRUE) }$$

$$\frac{}{\Gamma \triangleright \text{false} : \text{Bool}} \text{ (T-FALSE) }$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \triangleright x : \sigma} \text{ (T-VAR) }$$

Reglas de tipado de λ^b

$$\frac{\Gamma \triangleright M : Bool \quad \Gamma \triangleright P : \sigma \quad \Gamma \triangleright Q : \sigma}{\Gamma \triangleright \text{if } M \text{ then } P \text{ else } Q : \sigma} \text{ (T-IF)}$$

Reglas de tipado de λ^b

$$\frac{\Gamma \triangleright M : Bool \quad \Gamma \triangleright P : \sigma \quad \Gamma \triangleright Q : \sigma}{\Gamma \triangleright \text{if } M \text{ then } P \text{ else } Q : \sigma} \text{ (T-IF)}$$

$$\frac{\Gamma \triangleright M : \sigma \rightarrow \tau \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright M N : \tau} \text{ (T-APP)}$$

Reglas de tipado de λ^b

$$\frac{\Gamma \triangleright M : Bool \quad \Gamma \triangleright P : \sigma \quad \Gamma \triangleright Q : \sigma}{\Gamma \triangleright \text{if } M \text{ then } P \text{ else } Q : \sigma} \text{ (T-IF)}$$

$$\frac{\Gamma \triangleright M : \sigma \rightarrow \tau \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright M N : \tau} \text{ (T-APP)}$$

$$\frac{\Gamma, x : \sigma \triangleright M : \tau}{\Gamma \triangleright \lambda x : \sigma. M : \sigma \rightarrow \tau} \text{ (T-ABS)}$$

Sistema de tipado

- ▶ Si $\Gamma \triangleright M : \sigma$ puede derivarse usando los axiomas y reglas de tipado decimos que es **derivable**.
- ▶ Decimos que M es **tipable** si el juicio de tipado $\Gamma \triangleright M : \sigma$ puede derivarse, para algún Γ y σ .

Resultados básicos

Unicidad de tipos

Si $\Gamma \triangleright M : \sigma$ y $\Gamma \triangleright M : \tau$ 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 $\Gamma' \triangleright M : \sigma$

Variables libres

Una variable puede ocurrir **libre** o **ligada** en un término. Decimos que “ x ” ocurre **libre** si no se encuentra bajo el alcance de una ocurrencia de “ λx ”. Caso contrario ocurre ligada.

- ▶ $\lambda x : \text{Bool}. \underbrace{\text{if } x \text{ then true else false}}_{\text{ligada}}$
- ▶ $\lambda x : \text{Bool}. \lambda y : \text{Bool}. \underbrace{\text{if true then } x \text{ else } y}_{\substack{\text{ligada} \quad \text{ligada}}}$
- ▶ $\lambda x : \text{Bool}. \underbrace{\text{if } x \text{ then true else } y}_{\substack{\text{ligada} \quad \text{libre}}}$
- ▶ $(\lambda x : \text{Bool}. \underbrace{\text{if } x \text{ then true else false}}_{\text{ligada}}) \underbrace{x}_{\text{libre}}$

Variables libres: Definición formal

$$\begin{aligned}FV(x) &\stackrel{\text{def}}{=} \{x\} \\FV(\text{true}) = FV(\text{false}) &\stackrel{\text{def}}{=} \emptyset \\FV(\text{if } M \text{ then } P \text{ else } Q) &\stackrel{\text{def}}{=} FV(M) \cup FV(P) \cup FV(Q) \\FV(M N) &\stackrel{\text{def}}{=} FV(M) \cup FV(N) \\FV(\lambda x : \sigma. M) &\stackrel{\text{def}}{=} FV(M) \setminus \{x\}\end{aligned}$$

Semántica

- ▶ Habiendo definido la sintaxis de λ^b , nos interesa formular cómo se **evalúan** o **ejecutan** los términos
- ▶ Hay varias maneras de definir **rigurosamente** la semántica de un lenguaje de programación
 - ▶ Operacional
 - ▶ Denotacional
 - ▶ Axiomática
- ▶ Vamos a definir una **semántica operacional** para λ^b

¿Qué es semántica operacional?

- ▶ Consiste en
 - ▶ interpretar a los **términos como estados** de una máquina abstracta y
 - ▶ definir una **función de transición** que indica, dado un estado, cuál es el siguiente estado

¿Qué es semántica operacional?

- ▶ Consiste en
 - ▶ interpretar a los **términos como estados** de una máquina abstracta y
 - ▶ definir una **función de transición** que indica, dado un estado, cuál es el siguiente estado
- ▶ **Significado** de un término M : el estado final que alcanza la máquina al comenzar con M como estado inicial

¿Qué es semántica operacional?

- ▶ Consiste en
 - ▶ interpretar a los **términos como estados** de una máquina abstracta y
 - ▶ definir una **función de transición** que indica, dado un estado, cuál es el siguiente estado
- ▶ **Significado** de un término M : el estado final que alcanza la máquina al comenzar con M como estado inicial
- ▶ Formas de definir semántica operacional
 1. **Small-step**: la función de transición describe un paso de computación
 2. **Big-step** (o **Natural Semantics**): la función de transición, en un paso, evalúa el término a su resultado

Semántica operacional

- ▶ La formulación se hace a través de **juicios de evaluación**

$$M \rightarrow N$$

que se leen: “*el término M reduce, en un paso, al término N* ”

- ▶ El significado de un juicio de evaluación se establece a través de:
 - ▶ **Axiomas de evaluación**: establecen que ciertos juicios de evaluación son derivables.
 - ▶ **Reglas de evaluación** establecen que ciertos juicios de evaluación son derivables siempre y cuando ciertos otros lo sean.

Semántica operacional small-step de λ^b

- ▶ Vamos a presentar una semántica operacional **small-step** para el cálculo λ^b
- ▶ Además de introducir la función de transición es necesario introducir también los **valores**
 - ▶ **Valores**: Los posibles resultados de evaluación de términos bien-tipados y cerrados

Semántica Operacional - Expr. booleanas

Valores

$$V ::= \text{true} \mid \text{false}$$

Semántica Operacional - Expr. booleanas

Juicio de evaluación en un paso

$$\frac{}{\text{if true then } M_2 \text{ else } M_3 \rightarrow M_2} \text{ (E-IFTRUE)}$$

$$\frac{}{\text{if false then } M_2 \text{ else } M_3 \rightarrow M_3} \text{ (E-IFFALSE)}$$

$$\frac{M_1 \rightarrow M'_1}{\text{if } M_1 \text{ then } M_2 \text{ else } M_3 \rightarrow \text{if } M'_1 \text{ then } M_2 \text{ else } M_3} \text{ (E-IF)}$$

Ejemplos

$$\frac{\frac{}{\text{if false then false else true} \rightarrow \text{true}} \text{ (E-IFFALSE)}}{\text{if (if false then false else true) then false else true} \rightarrow \text{if true then false else true}} \text{ (E-IF)}$$

Observar que

- No existe M tal que $\text{true} \rightarrow M$ (idem con false).

Ejemplos

if true then (if false then false else true) else true
↗ *if true then true else true*

La estrategia de evaluación corresponde con el orden habitual en lenguajes de programación.

1. Primero evaluar la guarda del condicional
2. Una vez que la guarda sea un valor, seguir con la expresión del then o del else, según corresponda

Propiedades

Lema (Determinismo del juicio de evaluación en un paso)

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

Propiedades

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

Recordar que un valor es el resultado al que puede evaluar un término bien-tipado y cerrado

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):
 - ▶ *if x then true else false*
 - ▶ x
 - ▶ *true false*

Evaluación en muchos pasos

El juicio de **evaluación en muchos pasos** \rightarrow es la clausura reflexiva, transitiva de \rightarrow . Es decir, la menor relación tal que

1. Si $M \rightarrow M'$, entonces $M \rightarrow M'$
2. $M \rightarrow M$ para todo M
3. Si $M \rightarrow M'$ y $M' \rightarrow M''$, entonces $M \rightarrow M''$

if true then (if false then false else true) else true
 \rightarrow *true*

Evaluación en muchos pasos - Propiedades

Para el cálculo de expresiones booleanas valen:

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 $M \twoheadrightarrow N$

Semántica operacional de λ^b

Valores

$$V ::= \text{true} \mid \text{false} \mid \lambda x : \sigma. M$$

Introduciremos una noción de evaluación en λ^b tal que valgan los lemas previos y también el siguiente resultado:

Teorema

Todo término bien-tipado y cerrado de tipo

- ▶ Bool evalúa, en **cero o más** pasos, a true , false
- ▶ $\sigma \rightarrow \tau$ evalúa, en **cero o más** pasos, a $\lambda x : \sigma. M$, para alguna variable x , para algún término M

Semántica operacional de λ^b

Juicio de evaluación en un paso

$$\frac{M_1 \rightarrow M'_1}{M_1 M_2 \rightarrow M'_1 M_2} \text{ (E-APP1 / } \mu \text{)}$$

Semántica operacional de λ^b

Juicio de evaluación en un paso

$$\frac{M_1 \rightarrow M'_1}{M_1 M_2 \rightarrow M'_1 M_2} \text{ (E-APP1 / } \mu \text{)}$$

$$\frac{M_2 \rightarrow M'_2}{(\lambda x : \sigma.M) M_2 \rightarrow (\lambda x : \sigma.M) M'_2} \text{ (E-APP2 / } \nu \text{)}$$

Semántica operacional de λ^b

Juicio de evaluación en un paso

$$\frac{M_1 \rightarrow M'_1}{M_1 M_2 \rightarrow M'_1 M_2} \text{ (E-APP1 / } \mu \text{)}$$

$$\frac{M_2 \rightarrow M'_2}{(\lambda x : \sigma.M) M_2 \rightarrow (\lambda x : \sigma.M) M'_2} \text{ (E-APP2 / } \nu \text{)}$$

$$\frac{}{(\lambda x : \sigma.M) V \rightarrow} \text{ (E-APPABS / } \beta \text{)}$$

Semántica operacional de λ^b

Juicio de evaluación en un paso

$$\frac{M_1 \rightarrow M'_1}{M_1 M_2 \rightarrow M'_1 M_2} \text{ (E-APP1 / } \mu \text{)}$$

$$\frac{M_2 \rightarrow M'_2}{(\lambda x : \sigma.M) M_2 \rightarrow (\lambda x : \sigma.M) M'_2} \text{ (E-APP2 / } \nu \text{)}$$

$$\frac{}{(\lambda x : \sigma.M) V \rightarrow M\{x \leftarrow V\}} \text{ (E-APPABS / } \beta \text{)}$$

Además de (E-IFTRUE), (E-IFFALSE), (E-IF)

Sustitución

$$M\{x \leftarrow N\}$$

- ▶ “*Sustituir todas las ocurrencias **libres** de x en el término M por el término N* ”
- ▶ Operación importante que se usa para darle semántica a la aplicación de funciones (entre otras)
- ▶ Es sencilla de definir **pero** requiere cuidado en el tratamiento de los ligadores de variables (i.e. con “ λx ”)

Sustitución

$$x\{x \leftarrow N\} \stackrel{\text{def}}{=}$$

Sustitución

$$\begin{aligned} x\{x \leftarrow N\} &\stackrel{\text{def}}{=} N \\ a\{x \leftarrow N\} &\stackrel{\text{def}}{=} \end{aligned}$$

Sustitución

$$x\{x \leftarrow N\} \stackrel{\text{def}}{=} N$$

$$a\{x \leftarrow N\} \stackrel{\text{def}}{=} a \quad \text{si } a \in \{true, false\} \cup \mathcal{X} \setminus \{x\}$$

$$(if\ M\ then\ P\ else\ Q)\{x \leftarrow N\} \stackrel{\text{def}}{=}$$

Sustitución

$$x\{x \leftarrow N\} \stackrel{\text{def}}{=} N$$

$$a\{x \leftarrow N\} \stackrel{\text{def}}{=} a \quad \text{si } a \in \{true, false\} \cup \mathcal{X} \setminus \{x\}$$

$$(if\ M\ then\ P\ else\ Q)\{x \leftarrow N\} \stackrel{\text{def}}{=} \begin{array}{l} if\ M\{x \leftarrow N\} \\ \quad then\ P\{x \leftarrow N\} \\ \quad else\ Q\{x \leftarrow N\} \end{array}$$

$$(M_1\ M_2)\{x \leftarrow N\} \stackrel{\text{def}}{=}$$

Sustitución

$$x\{x \leftarrow N\} \stackrel{\text{def}}{=} N$$

$$a\{x \leftarrow N\} \stackrel{\text{def}}{=} a \quad \text{si } a \in \{true, false\} \cup \mathcal{X} \setminus \{x\}$$

$$(if \ M \ then \ P \ else \ Q)\{x \leftarrow N\} \stackrel{\text{def}}{=} \begin{array}{l} if \ M\{x \leftarrow N\} \\ \quad then \ P\{x \leftarrow N\} \\ \quad else \ Q\{x \leftarrow N\} \end{array}$$

$$(M_1 \ M_2)\{x \leftarrow N\} \stackrel{\text{def}}{=} M_1\{x \leftarrow N\} \ M_2\{x \leftarrow N\}$$

$$(\lambda y : \sigma. M)\{x \leftarrow N\} \stackrel{\text{def}}{=}$$

Sustitución

$$x\{x \leftarrow N\} \stackrel{\text{def}}{=} N$$

$$a\{x \leftarrow N\} \stackrel{\text{def}}{=} a \quad \text{si } a \in \{true, false\} \cup \mathcal{X} \setminus \{x\}$$

$$(if \ M \ then \ P \ else \ Q)\{x \leftarrow N\} \stackrel{\text{def}}{=} \begin{array}{l} if \ M\{x \leftarrow N\} \\ \quad then \ P\{x \leftarrow N\} \\ \quad else \ Q\{x \leftarrow N\} \end{array}$$

$$(M_1 \ M_2)\{x \leftarrow N\} \stackrel{\text{def}}{=} M_1\{x \leftarrow N\} \ M_2\{x \leftarrow N\}$$

$$(\lambda y : \sigma. M)\{x \leftarrow N\} \stackrel{\text{def}}{=} ?$$

Captura de variables

“Sustituir la variable x por el término z ”

$$(\lambda z : \sigma. x) \{x \leftarrow z\} = \lambda z : \sigma. z$$

Captura de variables

“Sustituir la variable x por el término z ”

$$(\lambda z : \sigma.x)\{x \leftarrow z\} = \lambda z : \sigma.z$$

- ▶ ¡Hemos convertido a la función constante $\lambda z : \sigma.x$ en la función identidad!
- ▶ **El problema:** “ $\lambda z : \sigma$ ” capturó la ocurrencia libre de z
- ▶ **Hipótesis:** los nombres de las variables ligadas no son relevantes
 - ▶ la ecuación de arriba debería ser comparable con

$$(\lambda w : \sigma.x)\{x \leftarrow z\} = \lambda w : \sigma.z$$

- ▶ **Conclusión:** Para definir $(\lambda y : \sigma.M)\{x \leftarrow N\}$ asumiremos que la variable ligada y se renombró de tal manera que **no** ocurre libre en N

α -equivalencia

- ▶ Dos términos M y N que difieren solamente en el nombre de sus variables ligadas se dicen α -equivalentes
 - ▶ α -equivalencia es una relación de equivalencia
 - ▶ De aquí en más identificaremos términos α -equivalentes.
-
- ▶ $\lambda x : Bool.x =_{\alpha} \lambda y : Bool.y$
 - ▶ $\lambda x : Bool.y =_{\alpha} \lambda z : Bool.y$
 - ▶ $\lambda x : Bool.y \neq_{\alpha} \lambda x : Bool.z$
 - ▶ $\lambda x : Bool.\lambda x : Bool.x \neq_{\alpha} \lambda y : Bool.\lambda x : Bool.y$

Sustitución - Revisada

$$\begin{aligned}x\{x \leftarrow N\} &\stackrel{\text{def}}{=} N \\a\{x \leftarrow N\} &\stackrel{\text{def}}{=} a \quad \text{si } a \in \{true, false\} \cup \mathcal{X} \setminus \{x\} \\(if\ M\ then\ P\ else\ Q)\{x \leftarrow N\} &\stackrel{\text{def}}{=} if\ M\{x \leftarrow N\}\ then\ P\{x \leftarrow N\} \\&\quad else\ Q\{x \leftarrow N\} \\(M_1\ M_2)\{x \leftarrow N\} &\stackrel{\text{def}}{=} M_1\{x \leftarrow N\}\ M_2\{x \leftarrow N\} \\(\lambda y : \sigma.M)\{x \leftarrow N\} &\stackrel{\text{def}}{=} \lambda y : \sigma.M\{x \leftarrow N\} \quad x \neq y, \ y \notin FV(N)\end{aligned}$$

1. NB: la condición $x \neq y, \ y \notin FV(N)$ **siempre** puede cumplirse renombrando apropiadamente
2. Técnicamente, la sustitución está definida sobre **clases de α -equivalencia** de términos

Estado de error

- ▶ 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

- ▶ $\text{if } x \text{ then } M \text{ else } N$
 - ▶ Obs: no es cerrado
- ▶ $\text{true } M$
 - ▶ Obs: no es tipable

Objetivo de un sistema de tipos

Garantizar la **ausencia** de estados de error

- ▶ Decimos que un término **termina** o que es **fuertemente normalizante** si no hay cadenas de reducción infinitas a partir de él.

Teorema

- ▶ Todo término bien tipado termina
- ▶ Si un término cerrado está bien tipado, entonces evalúa a un valor

Corrección

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

Progreso

Si M es cerrado y bien tipado entonces

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

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

Preservación

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

La evaluación preserva tipos

Tipos y términos de λ^{bn}

$$\sigma ::= Bool \mid Nat \mid \sigma \rightarrow \rho$$

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

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:

$$\frac{}{\Gamma \triangleright 0 : \text{Nat}} \text{ (T-ZERO)}$$

$$\frac{\Gamma \triangleright M : \text{Nat}}{\Gamma \triangleright \text{succ}(M) : \text{Nat}} \text{ (T-SUCC)}$$

$$\frac{\Gamma \triangleright M : \text{Nat}}{\Gamma \triangleright \text{pred}(M) : \text{Nat}} \text{ (T-PRED)}$$

$$\frac{\Gamma \triangleright M : \text{Nat}}{\Gamma \triangleright \text{iszero}(M) : \text{Bool}} \text{ (T-ISZERO)}$$

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

Valores

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

Juicio de evaluación en un paso (1/2)

$$\frac{M_1 \rightarrow M'_1}{\text{succ}(M_1) \rightarrow \text{succ}(M'_1)} \text{ (E-SUCC)}$$

$$\frac{}{\text{pred}(0) \rightarrow 0} \text{ (E-PREDZERO)}$$

$$\frac{}{\text{pred}(\underline{n+1}) \rightarrow \underline{n}} \text{ (E-PREDSUCC)}$$

$$\frac{M_1 \rightarrow M'_1}{\text{pred}(M_1) \rightarrow \text{pred}(M'_1)} \text{ (E-PRED)}$$

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

Juicio de evaluación en un paso (2/2)

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

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

$$\frac{M_1 \rightarrow M'_1}{iszero(M_1) \rightarrow iszero(M'_1)} \text{ (E-ISZERO)}$$

Además de los juicios de evaluación en un paso de λ^b .

Tipos y términos de λ^{bnu}

$$\sigma ::= Bool \mid Nat \mid Unit \mid \sigma \rightarrow \rho$$

$$M ::= \dots \mid unit$$

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:

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

NB:

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

$$V ::= \dots \mid unit$$

Bibliografía

Benjamin C. Pierce: **Types and programming languages**. MIT Press 2002, ISBN 0-262-16209-1. (Parte II - Simple Types)