

Cálculo Lambda Tipado (1/3)

Alejandro Ríos

Departamento de Computación, FCEyN, UBA

“There may, indeed, be other applications of the system other than its use as a logic”, Alonzo Church, 1932

24 de agosto de 2017

¿Qué es el Cálculo Lambda?

- ▶ Modelo de computación basado en **funciones**
 - ▶ da origen a la programación funcional
- ▶ Introducido por **Alonzo Church** en 1934
- ▶ Computacionalmente completo (i.e. Turing completo)
- ▶ También considerado como **modelo fiable de lenguajes de programación** en general
 - ▶ THE NEXT 700 PROGRAMMING LANGUAGES, Peter Landin, 1966
 - ▶ Lema: Usar Cálculo Lambda para probar nuevos conceptos de programación

¿Por qué Cálculo Lambda?

- ▶ En un lenguaje “industrial-strength” es **difícil determinar** con precisión/rigurosidad
 - ▶ propiedades básicas sobre semántica y sistema de tipos
 - ▶ efecto de extender el lenguaje con nuevas construcciones
 - ▶ la relación con otros lenguajes o paradigmas
- ▶ Es conveniente **restringir el lenguaje** a un subconjunto que sea
 - ▶ representativo (del paradigma o área de problemas)
 - ▶ conciso
 - ▶ reducido en cuanto a primitivas
 - ▶ riguroso en su formulación

¿Qué vamos a estudiar sobre Cálculo Lambda?

- ▶ La formulación original es **sin tipos**
- ▶ Dado nuestro interés en lenguajes de programación, vamos a estudiar el **Cálculo Lambda Tipado** (A. Church, 1941)
- ▶ En el marco del Cálculo Lambda Tipado vamos a presentar
 1. Tipos, términos, tipado, evaluación (Clase 1/3 y 2/3)
 2. Inferencia de tipos (Clase 3/3)
- ▶ Comenzaremos con **Cálculo Lambda Tipado con expresiones booleanas** y luego iremos enriqueciendo el lenguaje con otras construcciones

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 tipo σ en tipo τ

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

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

Términos de λ^b

Descripción informal:

- ▶ x es una **variable de términos**,
- ▶ *true* y *false* son las **constantes de verdad**,
- ▶ *if M then P else Q* es el **condicional**,
- ▶ $\lambda x : \sigma. M$ es una **función** cuyo parámetro formal es x y cuyo cuerpo es M
- ▶ $M N$ es la **aplicación** de la función denotada por el término M al argumento N .

Ejemplos

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

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.

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

Variables libres: Definición formal

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

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

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

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

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

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

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

Captura de variables

“Sustituir la variable x por el término z ”

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

- ▶ ¡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

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

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

Sistema de tipado

- ▶ Sistema formal de deducción (o derivación) que utiliza axiomas y reglas de tipado para caracterizar un subconjunto de los términos llamados **tipados**.
 - ▶ 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.
- ▶ Motivar juicios de tipado:
 - ▶ ¿Qué tipo le asignaría a *true*?
 - ▶ ¿Y a *if x then false else true*?

Sistema de tipado

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.

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 ”

Sistema de tipado

- ▶ El significado de $\Gamma \triangleright M : \sigma$ se establece a través de la introducción de **axiomas y reglas 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 .
- ▶ A continuación presentaremos los axiomas y reglas de tipado de λ^b

Axiomas de tipado de λ^b

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

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

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

Reglas de tipado de λ^b

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

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

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

Ejemplos de derivaciones de juicios de tipado

Vamos a mostrar que los siguientes juicios de tipado son derivables:

1. $\triangleright \lambda x : \text{Bool}. \lambda f : \text{Bool} \rightarrow \text{Bool}. f\ x : \text{Bool} \rightarrow (\text{Bool} \rightarrow \text{Bool}) \rightarrow \text{Bool}$
2. $x : \text{Bool}, y : \text{Bool} \triangleright \text{if } x \text{ then } y \text{ else } y : \text{Bool}$
3. $\triangleright \lambda f : \rho \rightarrow \tau. \lambda g : \sigma \rightarrow \rho. \lambda x : \sigma. f(g\ x) : (\rho \rightarrow \tau) \rightarrow (\sigma \rightarrow \rho) \rightarrow \sigma \rightarrow \tau$
4. ¿Existen Γ y σ tal que $\Gamma \triangleright x\ x : \sigma$?

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 $\Gamma' \triangleright M : \sigma$

Sustitución

Si $\Gamma, x : \sigma \triangleright M : \tau$ y $\Gamma \triangleright N : \sigma$ son derivables, entonces $\Gamma \triangleright M\{x \leftarrow N\} : \tau$ es derivable

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
- ▶ **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
- ▶ Lo haremos por partes
 - ▶ Primero abordamos las expresiones booleanas
 - ▶ Luego el resto de las expresiones
- ▶ Además de introducir la función de transición es conveniente introducir también los **valores**
 - ▶ **Valores**: Los posibles resultados de evaluación de términos bien-tipados (¿por qué?) y cerrados (¿por qué?)

Semántica Operacional - Expr. booleanas

Valores

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

Todo término bien-tipado y cerrado de tipo *Bool* evalúa, en **ceros o más** pasos, a *true* o *false*

- Este resultado se demuestra formalmente

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 valdrá 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 y término M

Semántica operacional de λ^b

Juicio de evaluación en un paso

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

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

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

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

Ejemplos

- ▶ $(\lambda y : \text{Bool}.y) \text{ true} \rightarrow \text{true}$
- ▶ $(\lambda x : \text{Bool} \rightarrow \text{Bool}.x \text{ true}) (\lambda y : \text{Bool}.y) \rightarrow (\lambda y : \text{Bool}.y) \text{ true}$
- ▶ $(\lambda z : \text{Bool}.z) ((\lambda y : \text{Bool}.y) \text{ true}) \rightarrow (\lambda z : \text{Bool}.z) \text{ true}$
- ▶ No existe M' tal que $x \rightarrow M'$
 - ▶ x está en forma normal pero **no** es un valor

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

- ▶ *if x then M else N*
 - ▶ Obs: no es cerrado
- ▶ *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 (díficil)
- ▶ Si un término cerrado está bien tipado, entonces evalúa a un valor (consecuencia de lo anterior y de los resultados de corrección que siguen)

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 decrementar
- ▶ $iszero(M)$: 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 $unit$

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

Utilidad

- ▶ Su utilidad principal es en lenguajes con efectos laterales (próxima clase)
- ▶ En estos lenguajes es útil poder evaluar varias expresiones en **secuencia**

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

- ▶ 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 $\lambda^{\dots let}$

$$M ::= \dots \mid let\ x : \sigma = M\ in\ N$$

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

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

Tipado de $\lambda^{.../let}$

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

Semántica operacional de $\lambda^{\dots/let}$

$$\frac{M_1 \rightarrow M'_1}{let\ x : \sigma = M_1\ in\ M_2 \rightarrow let\ x : \sigma = M'_1\ in\ M_2} \text{ (E-LET)}$$

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

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

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

$$\sigma ::= \dots \mid \{l_i : \sigma_i \mid i \in 1..n\}$$

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

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

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

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

Descripción informal:

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

Ejemplos

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

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

$$\frac{\Gamma \triangleright M_i : \sigma_i \quad \text{para cada } i \in 1..n}{\Gamma \triangleright \{l_i = M_i \mid i \in 1..n\} : \{l_i : \sigma_i \mid i \in 1..n\}} \text{ (T-RCD)}$$

$$\frac{\Gamma \triangleright M : \{l_i : \sigma_i \mid i \in 1..n\} \quad j \in 1..n}{\Gamma \triangleright M.l_j : \sigma_j} \text{ (T-PROJ)}$$

Semántica operacional de λ^{cr}

Valores

$$V ::= \dots \mid \{l_i = V_i \mid i \in 1..n\}$$

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

$$\frac{j \in 1..n}{\{l_i = V_i \mid i \in 1..n\}.l_j \rightarrow V_j} \text{ (E-PROJCD)}$$

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

$$\frac{M_j \rightarrow M'_j}{\{l_i = V_i \mid i \in 1..j-1, l_j = M_j, l_i = M_i \mid i \in j+1..n\} \rightarrow \{l_i = V_i \mid i \in 1..j-1, l_j = M'_j, l_i = M_i \mid i \in j+1..n\}} \text{ (E-RCD)}$$