Cálculo Lambda Tipado (1/3)

1 de Febrero de 2018

¿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)

Expresiones de tipos de λ^b

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

$$\sigma, \tau \quad ::= \quad \mathit{Bool} \,|\, \sigma \to \tau$$

Descripción informal:

- *Bool* es el tipo de los booleanos,
- $\sigma \to \tau$ 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

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

Términos de λ^b

Descripción informal:

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

Ejemplos

- $\lambda x : Bool.x$
- $\lambda x : Bool.if\ x\ then\ false\ else\ true$
- $\lambda f : \sigma \to \tau.\lambda x : \sigma.f\, x$
- $(\lambda f : Bool \to Bool.f\, true)(\lambda y : Bool.y)$
- $x\, y$

Variables ligadas y 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.

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

Variables libres: Definición formal

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

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

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

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

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

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.

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

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

Sistema de tipado

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

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

> "*el término M tiene tipo σ asumiendo el contexto de tipado Γ*"

El significado de $\Gamma \rhd M : \sigma$ se establece a través de la introducción de axiomas y reglas de tipado.

Axiomas de tipado de λ^b

$$\frac{}{\Gamma \rhd \mathit{true} : \mathit{Bool}}\ (\text{T-True}) \qquad \frac{}{\Gamma \rhd \mathit{false} : \mathit{Bool}}\ (\text{T-False})$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \rhd x : \sigma}\ (\text{T-Var})$$

Reglas de tipado de λ^b

$$\frac{\Gamma \rhd M : Bool \quad \Gamma \rhd P : \sigma \quad \Gamma \rhd Q : \sigma}{\Gamma \rhd \mathit{if}\ \ M\ \ \mathit{then}\, P\, \mathit{else}\, Q : \sigma}\,(\textsc{T-If})$$

$$\frac{\Gamma, x : \sigma \rhd M : \tau}{\Gamma \rhd \lambda x : \sigma.M : \sigma \to \tau}\,(\textsc{T-Abs}) \qquad \frac{\Gamma \rhd M : \sigma \to \tau \quad \Gamma \rhd N : \sigma}{\Gamma \rhd M\, N : \tau}\,(\textsc{T-App})$$

Ejemplos de derivaciones de juicios de tipado

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

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

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 $\sigma = \tau$

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

- 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

$$M \to N$$

  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 - Expr. booleanas

Valores

$$V \quad ::= \quad \mathit{true} \mid \mathit{false}$$

Semántica Operacional - Expr. booleanas

$$\frac{}{\textit{if true then } M_2 \textit{ else } M_3 \to M_2}\;(\textsc{E-IfTrue})$$

$$\frac{}{\textit{if false then } M_2 \textit{ else } M_3 \to M_3}\;(\textsc{E-IfFalse})$$

$$\frac{M_1 \to M_1'}{\textit{if } M_1 \textit{ then } M_2 \textit{ else } M_3 \to \textit{if } M_1' \textit{ then } M_2 \textit{ else } M_3}\;(\textsc{E-If})$$

Ejemplos

$$\dfrac{\dfrac{}{\textit{if false then false else true} \to \textit{true}}\,(\textsc{E-IfFalse})}{\begin{array}{c}\textit{if (if false then false else true) then false else true} \\ \to \\ \textit{if true then false else true}\end{array}}\,(\textsc{E-If})$$

Observar que

- No existe M tal que $\textit{true} \to M$ (idem con *false*).

Ejemplos

$$\begin{array}{rl} & \textit{if true then } (\textit{if false then false else true}) \textit{ else true} \\ \not\to & \textit{if true then true else true} \end{array}$$

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 $M \to M'$ 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, $M \to N$)

Lema

Todo valor está en forma normal

- No vale el recíproco: ejemplos
  - *if x then true else false*
  - *x*
  - *true false*

Evaluación en muchos pasos

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

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

$$\begin{array}{ll} & \mathit{if\ true\ then}\ (\mathit{if\ false\ then\ false\ else\ true})\ \mathit{else\ true} \\ \twoheadrightarrow & \mathit{true} \end{array}$$

Evaluación en muchos pasos - Propiedades

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

$$V \quad ::= \quad \mathit{true} \,|\, \mathit{false} \,|\, \lambda x : \sigma.M$$

Todo término bien-tipado y cerrado de tipo

- *Bool* evalúa, en cero o más pasos, a *true*, *false*
- $\sigma \to \tau$ 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

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

$$\frac{M_2 \to M_2'}{(\lambda x : \sigma.M)\, M_2 \to (\lambda x : \sigma.M)\, M_2'} \; (\text{E-App2} \; / \; \nu)$$

$$\frac{}{(\lambda x : \sigma.M)\, V \to M\{x \leftarrow V\}} \; (\text{E-AppAbs} \; / \; \beta)$$

Además de (E-IfTrue), (E-IfFalse), (E-If)

Ejemplos

- $(\lambda y : Bool.y)\, true \to true$
- $(\lambda x : Bool \to Bool.x\, true)\, (\lambda y : Bool.y) \to (\lambda y : Bool.y)\, true$
- $(\lambda z : Bool.z)\, ((\lambda y : Bool.y)\, true) \to (\lambda z : Bool.z)\, true$
- No existe M' tal que $x \to M'$
  - 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

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

Objetivo de un sistema de tipos

Garantizar la ausencia de estados de error

- Si un término cerrado está bien tipado (Â¡y termina!), 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 $M \to M'$

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

Preservación

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

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 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:

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

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

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

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

Valores

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

Juicio de evaluación en un paso (1/2)

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

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

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

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

Juicio de evaluación en un paso (2/2)

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

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

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

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