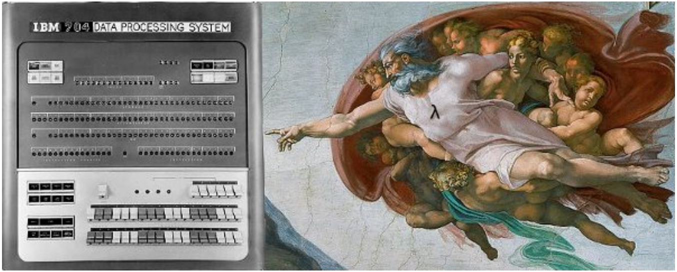
Lambda-Cálculo

Mauro Jaskelioff

13/09/2019



Origen del λ-cálculo

- El λ-cálculo fue inventado por Alonzo Church en la década de 1930.
- Originalmente fue inventado como parte de un sistema formal para modelar la mátematica.
  - ¡Pero es inconsistente!
- Es utilizado para estudiar la computabilidad.
  - En paralelo, Turing presenta su máquina.
- En los 1960s, Peter Landin muestra que se puede usar para dar semántica a los lenguajes de programación (imperativos).
- Los lenguajes funcionales están basados en el λ-cálculo.

SINTAXIS

Sintaxis

- Suponemos la existencia de un conjunto infinito de identificadores
  - x, y ,z, $\ldots$, x_0, x_1 denotan elementos de X
- El conjunto Λ de λ-términos se define inductivamente por las siguientes reglas:

$$\frac{x \in X}{x \in \Lambda} \qquad \frac{t \in \Lambda \quad u \in \Lambda}{(t\ u) \in \Lambda} \qquad \frac{x \in X \quad t \in \Lambda}{(\lambda x.\, t) \in \Lambda}$$

- Ejemplos:
  $x \quad (x\ y) \quad (\lambda x.\, x) \quad (\lambda x.\, (\lambda y.\, ((x\ y)\ y)))$

¿Esto es todo?

- ¡Con este pequeño lenguaje se pueden representar todas las funciones computables!
  - (Tesis de Church)
- Esta simpleza hace que:
  - Se facilite la prueba de propiedades.
  - Se use para dar semántica a lenguajes imperativos y funcionales.
  - Su use como metalenguaje para definir otras teorías y cálculos.
- ¡La elegancia hace que sea más práctico!

Convenciones

- Las mayúsculas indican λ-términos arbitrarios (ej: M,N,P)
- La aplicación asocia a la izquierda

$$M\ N\ P \quad \text{en lugar de} \quad ((M\ N)\ P)$$

- Las abstracciones se extienden tanto como sea posible, por lo que muchas veces los paréntesis no son necesarios.

$$\lambda x.\, P\ Q \quad \text{en lugar de} \quad (\lambda x.\, P\ Q)$$

La abstracción es sobre el término $(P\ Q)$

$$\lambda y.\, (\lambda x.\, P\ Q)\ R \quad \text{en lugar de} \quad (\lambda y.\, (\lambda x.\, P\ Q)\ R)$$

La abstracción interna es sobre $(P\ Q)$, sin R.

La abstracción de afuera es sobre $((\lambda x.\, P\ Q)\ R)$

- Podemos juntar varios lambdas.

$$\lambda x_1\, x_2 \ldots x_n.\, M \quad \text{en lugar de} \quad (\lambda x_1.\, (\lambda x_2.\, (\ldots (\lambda x_n.\, M) \ldots)))$$

Ejercicio

Ejercicio

Insertar todos los paréntesis y λs en los sig. términos abreviados:

- $x\ y\ z\ (y\ x)$
- $(\lambda x. v\ u\ u)\ z\ y$
- $(\lambda x\, y\, z. x\ z\ (y\ z))\ u\ v\ w$
- $u\ x\ (y\ z)\ (\lambda v. v\ y)$

Ocurrencias

- La **identidad sintáctica** se denota con $\equiv$
  - $M \equiv N$ sii M es exactamente el mismo término que N.

Definición (Ocurrencia)

La relación **P ocurre en Q** *(o P es un subtérmino de Q) se define inductivamente sobre la estructura de Q:*

- *P ocurre en P;*
- *si P ocurre en M o en N, entonces P ocurre en $(M\ N)$;*
- *si P ocurre en M o $P \equiv x$, entonces P ocurre en $(\lambda x.\, M)$.*

Ejercicio

Encontrar las ocurrencias de $(x\ y)$ en los términos

$(\lambda x\ y.\, x\ y)$ $\qquad$ $(z\ (x\ y)\ (\lambda x.\, y\ (x\ y))\ x\ y)$

Variables libres y ligadas

- Para una ocurrencia de $\lambda x.\, M$ en P, M es el **alcance** de la abstracción λx.
- Hay 3 tipos de ocurrencia de una variable x en un término P
  1. ocurrencia de ligadura (si es la x en un λx)
  2. ocurrencia ligada (si es una x en el alcance de un λx en P).
  3. ocurrencia libre (en cualquier otro caso).
- Llamamos $FV(P)$ al conjunto de las variables libres en P.
- Un **término cerrado** es un término sin variables libres.

Ejemplos

Determinar el tipo de ocurrencia de cada variable.

$$(\lambda x.\, x\ y) \qquad\qquad (\lambda x.\, x\ (\lambda x.\, x))\ x$$

- Observamos que
  - una misma variable puede ocurrir libre y ligada;
  - distintas ocurrencias pueden ligarse a distintas ocurrencias de ligadura;
  - la ligadura depende de toda la expresión (una ocurrencia puede cambiar de tipo al considerar una subexpresión en lugar de la expresión entera; ej: $(\lambda x.\, x)$ vs. x).

Ejercicio

Dar las variables libres y las ligaduras con sus alcances en el término

$$(\lambda y.\, y\ x\ (\lambda x.\, y\ (\lambda y.\, z)\ x))\ v\ w$$

Substitución

Definición (Substitución)

Para todo M, N, x se define $M[N/x]$ como el resultado de substituir N por toda ocurrencia libre de x en M. Más precisamente, por inducción sobre la estructura de M.

$$
\begin{array}{lll}
x[N/x] & \equiv N & \\
a[N/x] & \equiv a & (a \not\equiv x) \\
(P\ Q)[N/x] & \equiv (P[N/x]\ Q[N/x]) & \\
(\lambda x.\, P)[N/x] & \equiv \lambda x.\, P & \\
(\lambda y.\, P)[N/x] & \equiv \lambda y.\, P & \textit{si } x \notin FV(P) \wedge y \not\equiv x \\
(\lambda y.\, P)[N/x] & \equiv \lambda y.\, P[N/x] & \textit{si } x \in FV(P) \wedge y \notin FV(N) \\
(\lambda y.\, P)[N/x] & \equiv \lambda z.\, (P[z/y])[N/x] & \textit{si } x \in FV(P) \wedge y \in FV(N)
\end{array}
$$

Asumimos que $y \not\equiv x$ y que z es la 1er variable $\notin FV(N\ P)$

Ejercicios

Evaluar las siguientes substituciones

1. $(\lambda y.\, x\ (\lambda w.\, v\ w\ x))[(u\ v)/x]$
2. $(\lambda y.\, x\ (\lambda x.\, x))[(\lambda y.\, x\ y)/x]$
3. $(y\ (\lambda v.\, x\ v))[(\lambda y.\, v\ y)/x]$
4. $(\lambda x.\, x\ y)[(u\ v)/x]$

α-conversión

- Dado una ocurrencia de $\lambda x.\, M$ en un término P, si y no ocurre en M podemos reemplazar $\lambda x.\, M$ por:

$$\lambda y.\, (M[y/x])$$

- Esta operación se llama **cambio de variable ligada** o α-**conversión**.
- Si P puede cambiarse a Q por una serie finita de cambios de variable ligada decimos que P es **congruente** con Q, o que P α-convierte a Q, o

$$P \equiv_\alpha Q$$

- Ejemplo: $\lambda x\ y.\, x\ (x\ y) \equiv_\alpha \lambda u\ v.\, u\ (u\ v)$ (¡Probarlo!)

Propiedades de la α-conversión

Lema

a) *Si* $P \equiv_\alpha Q$ *entonces* $FV(P) = FV(Q)$

b) *La relación* $\equiv_\alpha$ *es una relación de equivalencia, o sea:*

es reflexiva	$P \equiv_\alpha P$
es simétrica	$P \equiv_\alpha Q \;\Rightarrow\; Q \equiv_\alpha P$
es transitiva	$P \equiv_\alpha Q \wedge Q \equiv_\alpha R \;\Rightarrow\; P \equiv_\alpha R$

Por lo tanto podemos hablar de α*-equivalencia.*

c) $M \equiv_\alpha M' \wedge N \equiv_\alpha N' \;\Rightarrow\; M[N/x] \equiv_\alpha M'[N'/x]$

- Salvo que se aclare lo contrario, escribiremos simplemente $\equiv$ en lugar de $\equiv_\alpha$.
  - Rara vez nos interesa diferenciar términos α-equivalentes.

SEMÁNTICA

β-reducción

- ¿Cómo calcular con el λ-cálculo?
- Un término $(\lambda x.\, M)\ N$ representa un operador $(\lambda x.\, M)$ aplicado a un argumento N.
- El "resultado" se obtiene usando la substitución $M[N/x]$.

Definición (redex, contracción, $\to_\beta$, $\to_\beta^*$)

*Un término $(\lambda x.\, M)\ N$ es un β-**redex** y $M[N/x]$ su **contracción**. Si al reemplazar un β-redex en un término P por su contracción obtenemos un término P', decimos que P se β-**contrae** a P' y escribimos*

$$P \to_\beta P'$$

Escribimos $\to_\beta^*$ para la clausura reflexiva-transitiva de $\to_\beta$ y decimos que P β-reduce a Q sii $P \to_\beta^* Q$.

Semántica operacional

$$\frac{t_1 \to_\beta t_1'}{t_1 \; t_2 \to_\beta t_1' \; t_2} \qquad (\textsc{E-App1})$$

$$\frac{t_2 \to_\beta t_2'}{t_1 \; t_2 \to_\beta t_1 \; t_2'} \qquad (\textsc{E-App2})$$

$$\frac{t_1 \to_\beta t_1'}{\lambda x.\, t_1 \to_\beta \lambda x.\, t_1'} \qquad (\textsc{E-Abs})$$

$$(\lambda x.\, t_1) \; t_2 \to_\beta t_1[t_2/x] \qquad (\textsc{E-AppAbs})$$

- La semántica es no determinística:
  - Para un término dado puede existir más de una forma de reducirlo.
- Las reglas E-App1, E-App2 y E-Abs son reglas de congruencia, E-AppAbs es una regla de computación.

Ejemplos

$$
\begin{array}{lcl}
(\lambda x.\, x\ (x\ y))\ N & \to_\beta & N\ (N\ y) \\
(\lambda x.\, y)\ N & \to_\beta & y \\
(\lambda x.\, (\lambda y.\, y\ x)\ z)\ v & \to_\beta & ((\lambda y.\, y\ x)\ z)[v/x] \equiv (\lambda y.\, y\ v)\ z \\
(\lambda x.\, x\ x)\ (\lambda x.\, x\ x) & \to_\beta & (x\ x)[(\lambda x.\, x\ x)/x] \equiv (\lambda x.\, x\ x)\ (\lambda x.\, x\ x) \\
 & \to_\beta & (x\ x)[(\lambda x.\, x\ x)/x] \equiv (\lambda x.\, x\ x)\ (\lambda x.\, x\ x) \\
 & \to_\beta & \ldots \\
(\lambda x.\, x\ x\ y)\ (\lambda x.\, x\ x\ y) & \to_\beta & (\lambda x.\, x\ x\ y)\ (\lambda x.\, x\ x\ y)\ y \\
 & \to_\beta & (\lambda x.\, x\ x\ y)\ (\lambda x.\, x\ x\ y)\ y\ y \\
 & \to_\beta & \ldots
\end{array}
$$

- ¡En los dos últimos ejemplos la reducción es infinita!

Forma Normal β

Definición (Formal Normal β)

Una **forma normal** *β o β-nf es un término que no contiene β-redexes.*

- Si un término P β-reduce a una β-nf Q decimos que Q es una forma normal β de P.

Ejercicio

Reducir los siguientes términos a β-nf.

$$(\lambda x.\, x\ y)\ (\lambda u.\, v\ u\ u) \qquad\qquad (\lambda x.\, x\ x\ y)\ (\lambda y.\, y\ z)$$

Propiedades de $\to_\beta^*$

- Nada nuevo es introducido en una reducción.

Lema

$$P \to_\beta^* Q \quad \Rightarrow \quad FV(P) \supseteq FV(Q)$$

- La relación $\to_\beta^*$ es preservada por la substitución

Lema

$$P \to_\beta^* P' \;\wedge\; Q \to_\beta^* Q' \quad \Rightarrow \quad Q[P/x] \to_\beta^* Q'[P'/x]$$

Confluencia

- Algunos términos tienen más de una reducción

$$
\begin{array}{lclcl}
(\lambda x.\,(\lambda y.\,y\ x)\ z)\ v & \to_\beta & (\lambda y.\,y\ v)\ z & \to_\beta & z\ v \\
(\lambda x.\,(\lambda y.\,y\ x)\ z)\ v & \to_\beta & (\lambda x.\,z\ x)\ v & \to_\beta & z\ v
\end{array}
$$

- ¿Reducen siempre a la misma forma normal?

Teorema (Church-Rosser para $\to_\beta$)

Si $P \to_\beta^* M$ y $P \to_\beta^* N$, entonces existe T tal que

$$
\begin{tikzcd}
 & P \arrow[dl, "\beta"', "*"] \arrow[dr, "\beta"', "*"] & \\
M \arrow[dr, "\beta"', "*"] & & N \arrow[dl, "\beta"', "*"] \\
 & T &
\end{tikzcd}
$$

Corolario

Si P tiene β-nf, ésta es única (módulo $\equiv_\alpha$).

β-equivalencia

Definición (β-equivalencia)

P es β-equivalente a Q (escribimos $P =_\beta Q$) sii Q puede ser obtenido partiendo de P y realizando una serie finita de β-contracciones, β-expansiones (β-contracciones inversas) y α-conversiones.

Lema (Substitución y $=_\beta$)

$$M =_\beta M' \quad \wedge \quad N =_\beta N' \qquad \Rightarrow \qquad M[N/x] =_\beta M'[N'/x]$$

Teorema (Church-Rosser para $=_\beta$)

Si $P =_\beta Q$ entonces existe T tal que

$$P \to_\beta^* T \qquad \wedge \qquad Q \to_\beta^* T$$

Extensionalidad

- Las λ-abstracciones representan funciones.
- Sin embargo, $\lambda x.\, f\ x \neq_{\beta} f$.
- Para tener un cálculo extensional, agregamos una nuevo redex (η-redex)
$$\lambda x.\, f\ x \to_{\eta} f$$
- $P \to_{\beta\eta} P' \quad \Leftrightarrow \quad P \to_{\beta} P'$ o $P \to_{\eta} P'$.
- En forma análoga al caso de β se obtiene $\to^{*}_{\beta\eta}$, forma normal $\beta\eta$ y equivalencia $=_{\beta\eta}$.
- El cálculo $\lambda\beta\eta$ es confluente (hay un teorema de Church-Rosser para $\beta\eta$).

Estrategias de reducción

- Por Church-Rosser si un término tiene una forma normal, ésta es única (¡probarlo!)
- Ya vimos que $\Omega \equiv (\lambda x.\, x\ x)\ (\lambda x.\, x\ x)$ tiene infinitas contracciones.
- Por lo tanto $P \equiv (\lambda x\ y.\, y)\ \Omega$ también.
- Sin embargo P tiene una forma normal $(\lambda y.\, y)$.
  - Claramente, la elección del redex a contraer es importante.
- ¿Cómo pruebo que un término no tiene forma normal?
- ¿Cómo puedo asegurarme de encontrar la forma normal? (si esta existe)

Reducción Normal

- Un redex es **maximal** si no está contenido en algún otro redex.
- Un redex es **maximal izquierdo** si es el redex maximal de más a la izquierda.
- La estrategia de reducción **normal** es elegir siempre el redex maximal izquierdo.

Reducción Normal: Evaluación

$$\frac{na_1 \to t_1'}{na_1\ t_2 \to t_1'\ t_2} \qquad \text{(E-App1)}$$

$$\frac{t_2 \to t_2'}{neu_1\ t_2 \to neu_1\ t_2'} \qquad \text{(E-App2)}$$

$$\frac{t_1 \to t_1'}{\lambda x.\, t_1 \to \lambda x.\, t_1'} \qquad \text{(E-Abs)}$$

$$(\lambda x.\, t_1)\ t_2 \to t_1[t_2/x] \qquad \text{(E-AppAbs)}$$

$$\begin{aligned} nf &::= \lambda x.\, nf \mid neu \\ neu &::= x \mid neu\ nf \\ na &::= x \mid t_1\ t_2 \end{aligned}$$

Reducción Normal (cont.)

Teorema

Si la reducción normal de un término X es infinita, X no tiene forma normal.

- Para probar que un término no tiene forma normal basta probarlo para la reducción normal.
- Si una forma normal existe, la estrategia de reducción normal la encontrará.

Resumen

- El λ-cálculo es un cálculo muy simple, pero muy poderoso.
- Ligadura de variables (binding)
- Nociones de reducción y de equivalencia.
- Estrategia de reducción normal.

Referencias

- *Lambda-Calculus and Combinators*. J. R. Hindley and J. P. Seldin. Cambridge University Press (2008).
- *Theories of Programming Languages*. J. Reynolds (1998).
- *Types and Programming Languages*. B.C. Pierce (2002).