Fundamentos de lenguajes de programación cuántica

Día 3:
$\sim$ Introducción a λ-cálculo y teoría de tipos $\backsim$

Alejandro Díaz-Caro

Universidad Nacional de Quilmes

22º Escuela de Verano de Ciencias Informáticas

Río Cuarto, Córdoba – 9 al 14 de febrero de 2015

Historia e intuiciones

Introducido en 1936 por Alonzo Church (director de Alan Turing)

Motivación: Investigar los *fundamentos de la matemática*
(en particular, el concepto de recursión)

Historia e intuiciones

Introducido en 1936 por Alonzo Church (director de Alan Turing)

Motivación: Investigar los *fundamentos de la matemática*
(en particular, el concepto de recursión)

¿Porqué aún lo usamos?

- Las funciones recursivas son fundamentales en computación
- Es el modelo más simple para estudiar propiedades del cómputo

Historia e intuiciones

Introducido en 1936 por Alonzo Church (director de Alan Turing)

Motivación: Investigar los *fundamentos de la matemática*
(en particular, el concepto de recursión)

¿Porqué aún lo usamos?

- Las funciones recursivas son fundamentales en computación
- Es el modelo más simple para estudiar propiedades del cómputo

Dos simplificaciones fundamentales

- Anonimidad de funciones:
  Ejemplo: $sumcuad(x, y) = x^2 + y^2$
  se escribe anónimamente como $(x, y) \mapsto x^2 + y^2$

Los nombres no son necesarios

Historia e intuiciones

Introducido en 1936 por Alonzo Church (director de Alan Turing)

Motivación: Investigar los *fundamentos de la matemática*
(en particular, el concepto de recursión)

¿Porqué aún lo usamos?

- Las funciones recursivas son fundamentales en computación
- Es el modelo más simple para estudiar propiedades del cómputo

Dos simplificaciones fundamentales

- Anonimidad de funciones:
  Ejemplo: $sumcuad(x, y) = x^2 + y^2$
  se escribe anónimamente como $(x, y) \mapsto x^2 + y^2$

  Los nombres no son necesarios

- Todas las funciones son de una sola variable:
  Ejemplo: $(x, y) \mapsto x^2 + y^2$
  se escribe como $x \mapsto (y \mapsto x^2 + y^2)$

  Una función de 2 variables es una función de 1 variable que retorna una función de 1 variable la cual hace el cálculo

Formalización

Lenguaje de términos (gramática)

$$\mathbf{t}, \mathbf{r} \quad ::= \quad x \quad | \quad \lambda x.\mathbf{t} \quad | \quad \mathbf{t}\ \mathbf{r}$$

- Una variable $x \in \mathit{Vars}$ es un λ-término
- Si $\mathbf{t}$ es un término, y x una variable, $\lambda x.\mathbf{t}$ es un término $(x \mapsto \mathbf{t})$
- Si $\mathbf{t}$ y $\mathbf{r}$ son términos, $\mathbf{t}\ \mathbf{r}$ es un término (aplicación)

Estos son todos los términos posibles

Formalización

Lenguaje de términos (gramática)

$$\mathbf{t}, \mathbf{r} \quad ::= \quad x \quad | \quad \lambda x.\mathbf{t} \quad | \quad \mathbf{t}\ \mathbf{r}$$

- Una variable $x \in \mathit{Vars}$ es un λ-término
- Si $\mathbf{t}$ es un término, y x una variable, $\lambda x.\mathbf{t}$ es un término $(x \mapsto \mathbf{t})$
- Si $\mathbf{t}$ y $\mathbf{r}$ son términos, $\mathbf{t}\ \mathbf{r}$ es un término (aplicación)

Estos son todos los términos posibles

Una regla de reescritura (β-reducción)

$$(\lambda x.\mathbf{t})\ \mathbf{r} \to \mathbf{t}[\mathbf{r}/x]$$

Formalización

Lenguaje de términos (gramática)

$$\mathbf{t}, \mathbf{r} \quad ::= \quad x \quad | \quad \lambda x.\mathbf{t} \quad | \quad \mathbf{t}\ \mathbf{r}$$

- Una variable $x \in \mathit{Vars}$ es un λ-término
- Si $\mathbf{t}$ es un término, y x una variable, $\lambda x.\mathbf{t}$ es un término $(x \mapsto \mathbf{t})$
- Si $\mathbf{t}$ y $\mathbf{r}$ son términos, $\mathbf{t}\ \mathbf{r}$ es un término (aplicación)

Estos son todos los términos posibles

Una regla de reescritura (β-reducción)

$$(\lambda x.\mathbf{t})\ \mathbf{r} \to \mathbf{t}[\mathbf{r}/x]$$

Ejemplo: Sea $x^2 + 1$ un λ-término (con alguna codificación)

$$f(x) = x^2 + 1 \qquad \text{se escribe} \qquad \lambda x.x^2 + 1$$

Formalización

Lenguaje de términos (gramática)

$$\mathbf{t}, \mathbf{r} \quad ::= \quad x \quad | \quad \lambda x.\mathbf{t} \quad | \quad \mathbf{t}\ \mathbf{r}$$

- Una variable $x \in \mathit{Vars}$ es un λ-término
- Si $\mathbf{t}$ es un término, y x una variable, $\lambda x.\mathbf{t}$ es un término $(x \mapsto \mathbf{t})$
- Si $\mathbf{t}$ y $\mathbf{r}$ son términos, $\mathbf{t}\ \mathbf{r}$ es un término (aplicación)

Estos son todos los términos posibles

Una regla de reescritura (β-reducción)

$$(\lambda x.\mathbf{t})\ \mathbf{r} \to \mathbf{t}[\mathbf{r}/x]$$

Ejemplo: Sea $x^2 + 1$ un λ-término (con alguna codificación)

$$f(x) = x^2 + 1 \qquad \text{se escribe} \qquad \lambda x.x^2 + 1$$

$f(\mathbf{t})$ se escribe $(\lambda x.x^2 + 1)\ \mathbf{t}$ y β-reduce a

$$(x^2 + 1)[\mathbf{t}/x] \quad = \quad \mathbf{t}^2 + 1$$

Codificación de Church

Booleanos

$$\texttt{True} = \lambda x.\lambda y.x \qquad (f(x,y) = x)$$
$$\texttt{False} = \lambda x.\lambda y.y \qquad (f(x,y) = y)$$

Codificación de Church

Booleanos

$$\texttt{True} = \lambda x.\lambda y.x \qquad (f(x,y) = x)$$

$$\texttt{False} = \lambda x.\lambda y.y \qquad (f(x,y) = y)$$

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e$$

Codificación de Church

Booleanos

$$\texttt{True} = \lambda x.\lambda y.x \qquad (f(x,y) = x)$$

$$\texttt{False} = \lambda x.\lambda y.y \qquad (f(x,y) = y)$$

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e$$

Ejemplos

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e \begin{cases} c = \texttt{True} \to t \\ c = \texttt{False} \to e \end{cases}$$

Codificación de Church

Booleanos

$$\texttt{True} = \lambda x.\lambda y.x \qquad (f(x,y) = x)$$

$$\texttt{False} = \lambda x.\lambda y.y \qquad (f(x,y) = y)$$

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e$$

$$p \texttt{ and } s = p\ s\ p$$

Ejemplos

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e \begin{cases} c = \texttt{True} \rightarrow t \\ c = \texttt{False} \rightarrow e \end{cases}$$

$$p \texttt{ and } s = p\ s\ p \begin{cases} p = \texttt{T},\ s = \texttt{T} \rightarrow \texttt{True} \\ \text{otro caso} \rightarrow \texttt{False} \end{cases}$$

Codificación de Church

Booleanos

$$\texttt{True} = \lambda x.\lambda y.x \qquad (f(x,y) = x)$$

$$\texttt{False} = \lambda x.\lambda y.y \qquad (f(x,y) = y)$$

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e$$

$$p \texttt{ and } s = p\ s\ p$$

$$p \texttt{ or } s = p\ p\ s$$

Ejemplos

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e \xrightarrow{c = \texttt{True}} t \qquad \xrightarrow{c = \texttt{False}} e$$

$$p \texttt{ and } s = p\ s\ p \xrightarrow{p = \texttt{T},\ s = \texttt{T}} \texttt{True} \qquad \xrightarrow{\text{otro caso}} \texttt{False}$$

$$p \texttt{ or } s = p\ p\ s \xrightarrow{\text{otro caso}} \texttt{True} \qquad \xrightarrow{p = \texttt{F},\ s = \texttt{F}} \texttt{False}$$

Codificación de Church

Booleanos

$$\texttt{True} = \lambda x.\lambda y.x \qquad (f(x,y) = x)$$

$$\texttt{False} = \lambda x.\lambda y.y \qquad (f(x,y) = y)$$

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e$$

$$p \texttt{ and } s = p\ s\ p$$

$$p \texttt{ or } s = p\ p\ s$$

Ejemplos

$$\texttt{If } c \texttt{ Then } t \texttt{ Else } e = c\ t\ e \xrightarrow{c = \texttt{True}} t \qquad \xrightarrow{c = \texttt{False}} e$$

$$p \texttt{ and } s = p\ s\ p \xrightarrow{p = \texttt{T},\ s = \texttt{T}} \texttt{True} \qquad \xrightarrow{\text{otro caso}} \texttt{False}$$

$$p \texttt{ or } s = p\ p\ s \xrightarrow{\text{otro caso}} \texttt{True} \qquad \xrightarrow{p = \texttt{F},\ s = \texttt{F}} \texttt{False}$$

Tarea:
not y xor

Codificación de Church

Enteros

- Los números son representados con funciones

Codificación de Church

Enteros

- Los números son representados con funciones
- n es $F(f, x) = f^n(x)$

Codificación de Church

Enteros

- Los números son representados con funciones
- n es $F(f, x) = f^n(x)$

Es decir:

$$\underline{n} = \lambda f.\lambda x.f^n\ x$$

El valor del número es la cantidad de veces que se aplica la función al argumento

Codificación de Church

Enteros

- Los números son representados con funciones
- n es $F(f, x) = f^n(x)$

Es decir:

$$\underline{n} = \lambda f.\lambda x.f^n\ x$$

El valor del número es la cantidad de veces que se aplica la función al argumento

Ejemplos:

$$\underline{0} = \lambda f.\lambda x.x$$
$$\underline{1} = \lambda f.\lambda x.f\ x$$
$$\underline{2} = \lambda f.\lambda x.f(f\ x)$$
$$\vdots$$
$$\underline{7} = \lambda f.\lambda x.f(f(f(f(f(f(f\ x))))))$$

Codificación de Church

Enteros

- Los números son representados con funciones
- n es $F(f, x) = f^n(x)$

Es decir:

$$\underline{n} = \lambda f.\lambda x.f^n\ x$$

El valor del número es la cantidad de veces que se aplica la función al argumento

Ejemplos:

$$\underline{0} = \lambda f.\lambda x.x$$
$$\underline{1} = \lambda f.\lambda x.f\ x$$
$$\underline{2} = \lambda f.\lambda x.f(f\ x)$$
$$\vdots$$
$$\underline{7} = \lambda f.\lambda x.f(f(f(f(f(f(f\ x))))))$$

Operaciones (algunas):

Suma

$$\lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x)$$

Multiplicación

$$\lambda m.\lambda n.\lambda f.m\ (n\ f)$$

Formas normales

No todo cómputo termina bien...

Consideremos $\lambda x.xx$

(la función que toma un argumento y lo aplica a sí mismo)

Formas normales

No todo cómputo termina bien...

Consideremos $\lambda x.xx$

(la función que toma un argumento y lo aplica a sí mismo)

$$\Omega = (\lambda x.xx)(\lambda x.xx)$$

Formas normales

No todo cómputo termina bien...

Consideremos $\lambda x.xx$

(la función que toma un argumento y lo aplica a sí mismo)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \rightarrow xx[\lambda x.xx/x]$$

Formas normales

No todo cómputo termina bien...

Consideremos $\lambda x.xx$

(la función que toma un argumento y lo aplica a sí mismo)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \rightarrow xx[\lambda x.xx/x] = (\lambda x.xx)(\lambda x.xx) = \Omega$$

Así que $\Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$

Formas normales

No todo cómputo termina bien...

Consideremos $\lambda x.xx$

(la función que toma un argumento y lo aplica a sí mismo)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \rightarrow xx[\lambda x.xx/x] = (\lambda x.xx)(\lambda x.xx) = \Omega$$

Así que $\Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$

Normalización

t está en forma normal, si no reescribe a nada ej. $\lambda x.x$

Formas normales

No todo cómputo termina bien...

Consideremos $\lambda x.xx$

(la función que toma un argumento y lo aplica a sí mismo)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \rightarrow xx[\lambda x.xx/x] = (\lambda x.xx)(\lambda x.xx) = \Omega$$

Así que $\Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$

Normalización

t está en forma normal, si no reescribe a nada ej. $\lambda x.x$

t es normalizante si *puede* terminar ej. $(\lambda x.\lambda y.y)\Omega$

Formas normales

No todo cómputo termina bien...

Consideremos $\lambda x.xx$

(la función que toma un argumento y lo aplica a sí mismo)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \rightarrow xx[\lambda x.xx/x] = (\lambda x.xx)(\lambda x.xx) = \Omega$$

Así que $\Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$

Normalización

t está en forma normal, si no reescribe a nada	ej.	$\lambda x.x$
t es normalizante si *puede* terminar	ej.	$(\lambda x.\lambda y.y)\Omega$
t es fuertemente normalizante si *siempre* termina	ej.	$(\lambda x.x)(\lambda x.x)$

Formas normales

No todo cómputo termina bien...

Consideremos $\lambda x.xx$

(la función que toma un argumento y lo aplica a sí mismo)

$$\Omega = (\lambda x.xx)(\lambda x.xx) \rightarrow xx[\lambda x.xx/x] = (\lambda x.xx)(\lambda x.xx) = \Omega$$

Así que $\Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \cdots$

Normalización

t está en forma normal, si no reescribe a nada	ej.	$\lambda x.x$
t es normalizante si *puede* terminar	ej.	$(\lambda x.\lambda y.y)\Omega$
t es fuertemente normalizante si *siempre* termina	ej.	$(\lambda x.x)(\lambda x.x)$

¿Cómo saber si un término es (fuertemente) normalizante?

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax$$

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B}\ \Rightarrow_I$$

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B}\ \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B}\ \Rightarrow_E$$

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B} \Rightarrow_E$$

Ejemplo de derivación de tipo

$$\frac{}{x : A \vdash x : A}\ ax$$

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B} \Rightarrow_E$$

Ejemplo de derivación de tipo

$$\dfrac{\dfrac{}{x : A \vdash x : A}\ ax}{\vdash \lambda x.x : A \Rightarrow A} \Rightarrow_I$$

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

$$\begin{array}{lrcl} \textbf{Términos} & \mathbf{t}, \mathbf{r}, \mathbf{s} & ::= & x \mid \lambda x.\mathbf{t} \mid \mathbf{t}\ \mathbf{r} \\ \textbf{Tipos} & A, B, C & ::= & \tau \mid A \Rightarrow B \end{array}$$

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\quad \Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A \quad$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B} \Rightarrow_E$$

Ejemplo de derivación de tipo

$$\frac{\dfrac{}{y : A \Rightarrow A \vdash y : A \Rightarrow A}\ ax}{\vdash \lambda y.y : (A \Rightarrow A) \Rightarrow (A \Rightarrow A)} \Rightarrow_I \qquad \frac{\dfrac{}{x : A \vdash x : A}\ ax}{\vdash \lambda x.x : A \Rightarrow A} \Rightarrow_I$$

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\quad \Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A \quad$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B} \Rightarrow_E$$

Ejemplo de derivación de tipo

$$\frac{\dfrac{\dfrac{}{y : A \Rightarrow A \vdash y : A \Rightarrow A}\ ax}{\vdash \lambda y.y : (A \Rightarrow A) \Rightarrow (A \Rightarrow A)} \Rightarrow_I \qquad \dfrac{\dfrac{}{x : A \vdash x : A}\ ax}{\vdash \lambda x.x : A \Rightarrow A} \Rightarrow_I}{\vdash (\lambda y.y)\ (\lambda x.x) : A \Rightarrow A} \Rightarrow_E$$

Tipos simples

Clasificación estática de λ-términos (o sea, sin reducirlos)

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	τ	$\mid$	$A \Rightarrow B$	

- τ es un *tipo de base*
- $A \Rightarrow B$ es el tipo función

Contexto: conjunto de variables tipadas $\quad \Gamma = x_1 : A_1, \ldots, x_n : A_n$

$\Gamma \vdash \mathbf{t} : A \quad$ "$\mathbf{t}$ tiene tipo A en el contexto Γ"

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B} \Rightarrow_E$$

Ejemplo de derivación de tipo

$$\frac{\dfrac{\dfrac{}{y : A \Rightarrow A \vdash y : A \Rightarrow A}\ ax}{\vdash \lambda y.y : (A \Rightarrow A) \Rightarrow (A \Rightarrow A)} \Rightarrow_I \qquad \dfrac{\dfrac{}{x : A \vdash x : A}\ ax}{\vdash \lambda x.x : A \Rightarrow A} \Rightarrow_I}{\vdash (\lambda y.y)\ (\lambda x.x) : A \Rightarrow A} \Rightarrow_E$$

Verificación: $(\lambda y.y)\ (\lambda x.x)$ reescribe a $\lambda x.x$ (de tipo $A \Rightarrow A$)

Normalización

Ω no tiene tipo en esta teoría

Normalización

Ω no tiene tipo en esta teoría

Más aún...

Teorema (Normalización fuerte)

Si $\mathbf{t}$ tiene tipo, $\mathbf{t}$ es fuertemente normalizante.

Normalización

Ω no tiene tipo en esta teoría

Más aún...

Teorema (Normalización fuerte)

Si **t** tiene tipo, **t** es fuertemente normalizante.

Eslogan: *"Well-typed programs cannot go wrong"* — `[R. Milner'78]`

¿Cómo se relaciona esto con lógica?

Una palabra sobre la correspondencia de Curry-Howard

Lógica clásica: una proposición se asume verdadera o falsa

¿Cómo se relaciona esto con lógica?

Una palabra sobre la correspondencia de Curry-Howard

Lógica clásica: una proposición se asume verdadera o falsa

Lógica intuicionista: una proposición es verdadera si hay una prueba constructiva que muestre que lo es.

¡La ley del tercero excluido no es un axioma! (y tampoco se puede demostrar) en lógica intuicionista	$A \vee \neg A$

¿Cómo se relaciona esto con lógica?

Una palabra sobre la correspondencia de Curry-Howard

Lógica clásica: una proposición se asume verdadera o falsa

Lógica intuicionista: una proposición es verdadera si hay una prueba constructiva que muestre que lo es.

¡La ley del tercero excluido no es un axioma! (y tampoco se puede demostrar) en lógica intuicionista	$A \vee \neg A$

Lógica intuicionista mínima

$$\frac{}{\Gamma, A \vdash A}\, ax \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow_E$$

¿Cómo se relaciona esto con lógica?

Una palabra sobre la correspondencia de Curry-Howard

Lógica clásica: una proposición se asume verdadera o falsa

Lógica intuicionista: una proposición es verdadera si hay una prueba constructiva que muestre que lo es.

¡La ley del tercero excluido no es un axioma! (y tampoco se puede demostrar) en lógica intuicionista	$A \vee \neg A$

Lógica intuicionista mínima

$$\frac{}{\Gamma, A \vdash A}\, ax \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow_E$$

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\, ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B} \Rightarrow_E$$

¿Cómo se relaciona esto con lógica?

Una palabra sobre la correspondencia de Curry-Howard

Lógica clásica: una proposición se asume verdadera o falsa

Lógica intuicionista: una proposición es verdadera si hay una prueba constructiva que muestre que lo es.

¡La ley del tercero excluido no es un axioma! (y tampoco se puede demostrar) en lógica intuicionista	$A \vee \neg A$

Lógica intuicionista mínima

$$\frac{}{\Gamma, A \vdash A}\, ax \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\, \Rightarrow_I \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\, \Rightarrow_E$$

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\, ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B}\, \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B}\, \Rightarrow_E$$

El λ-término es la prueba de la propocisión

¡Las pruebas. . . son programas!

Haskell Curry y William Howard entre 1934 y 1969

¿Cómo se relaciona esto con lógica?

Una palabra sobre la correspondencia de Curry-Howard

Lógica clásica: una proposición se asume verdadera o falsa

Lógica intuicionista: una proposición es verdadera si hay una prueba constructiva que muestre que lo es.

¡La ley del tercero excluido no es un axioma!
(y tampoco se puede demostrar) en lógica intuicionista

$$A \vee \neg A$$

Lógica intuicionista mínima

$$\frac{}{\Gamma, A \vdash A}\, ax \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\, \Rightarrow_I \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}\, \Rightarrow_E$$

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\, ax \qquad \frac{\Gamma, x : A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x.\mathbf{t} : A \Rightarrow B}\, \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B}\, \Rightarrow_E$$

El λ-término es la prueba de la propocisión

¡Las pruebas. . . son programas!

Haskell Curry y William Howard
entre 1934 y 1969

Lógicas más complejas se corresponden a sistemas más complejos

Polimorfismo. . . (en un slide)

$$\lambda x.x : \tau \Rightarrow \tau$$
$$\lambda x.x : (\tau \Rightarrow \tau) \rightarrow (\tau \Rightarrow \tau)$$
$$\lambda x.x : A \Rightarrow A$$

Polimorfismo. . . (en un slide)

$$\lambda x.x : \tau \Rightarrow \tau$$
$$\lambda x.x : (\tau \Rightarrow \tau) \rightarrow (\tau \Rightarrow \tau)$$
$$\lambda x.x : A \Rightarrow A$$

$$\boxed{\lambda x.x : \forall X.X \Rightarrow X}$$

Polimorfismo. . . (en un slide)

$\lambda x.x : \tau \Rightarrow \tau$

$\lambda x.x : (\tau \Rightarrow \tau) \to (\tau \Rightarrow \tau)$

$\lambda x.x : A \Rightarrow A$

$$\boxed{\lambda x.x : \forall X.X \Rightarrow X}$$

Términos	$\mathbf{t}, \mathbf{r}, \mathbf{s}$	$::=$	x	$\mid$	$\lambda x.\mathbf{t}$	$\mid$ $\mathbf{t}\ \mathbf{r}$
Tipos	A, B, C	$::=$	X	$\mid$	$A \Rightarrow B$	$\mid$ $\forall X.A$

$$X \in TVar$$

Reglas de tipado

$$\frac{}{\Gamma, x^A \vdash x : A}\ ax \qquad \frac{\Gamma, x^A \vdash \mathbf{t} : B}{\Gamma \vdash \lambda x^A.\mathbf{t} : A \Rightarrow B} \Rightarrow_I \qquad \frac{\Gamma \vdash \mathbf{t} : A \Rightarrow B \quad \Gamma \vdash \mathbf{r} : A}{\Gamma \vdash \mathbf{t}\ \mathbf{r} : B} \Rightarrow_E$$

$$\frac{\Gamma \vdash \mathbf{t} : A}{\Gamma \vdash \mathbf{t} : \forall X.A}\ \forall_I \text{ If } X \notin FV(\Gamma) \qquad \frac{\Gamma \vdash \mathbf{t} : \forall X.A}{\Gamma \vdash \mathbf{t} : A[B/X]}\ \forall_E$$

Se corresponde con lógica proposicional intuisionista de segundo orden