

Teoría de categorías y cálculo lambda

Mario Román García

21 de junio de 2018

Grado en Ingeniería Informática y Matemáticas - Universidad de Granada

El **cálculo lambda** es un sistema formal dado por ecuaciones sobre términos lambda. Abstracción y aplicación son nociones primitivas.

$$\text{Expr} := \begin{cases} x, & \text{variables, de un conjunto numerable,} \\ \text{Expr Expr}, & \text{aplicación de funciones,} \\ \lambda x. \text{Expr}, & \text{abstracción sobre una variable.} \end{cases}$$

Consideramos

- α -equivalencia, invariancia a renombramientos $(\lambda x. M[x]) \equiv (\lambda y. M[y])$;
- β -reducciones, aplicación de funciones $(\lambda x. M) N \longrightarrow_{\beta} M_{[N/x]}$;
- η -reducciones, extensionalidad $(\lambda x. f \ x) \equiv f$.

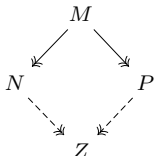
Alonzo Church. “A set of postulates for the foundation of logic”. En: Annals of mathematics (1932), págs. 346-366.

H.P. Barendregt. The lambda calculus: its syntax and semantics. Studies in logic and the foundations of mathematics. North-Holland, 1984. ISBN: 9780444867483.

Buscamos **formas normales** invariantes a reducciones. Existen términos que divergen como $\Omega = (\lambda x.(xx))(\lambda x.(xx))$.

Teorema (Church-Rosser)

La reducción es confluente.



Si existe la forma normal, es única.

Hay expresiones que se reducirán o no dependiendo del orden de evaluación, como $(\lambda x.\lambda y.y) \Omega (\lambda x.x)$.

Teorema (Barendregt)

Si existe una forma normal del término lambda, la estrategia que reduce a cada paso la aplicación más a la izquierda la encuentra.

La reducción da una forma de cálculo **Turing-completa**.

Robert Pollack. "Polishing Up the Tait-Martin-Löf Proof of the Church-Rosser Theorem". En: Proc. De Winternöte, Chalmers University (1995).

Ryo Kashima. "A Proof of the Standardization Theorem in Lambda-Calculus". En: Tokyo Institute of Technology (2000).

mario@kosmos ~ mikrokosmos

Welcome to the Mikrokosmos Lambda Interpreter!

Version 0.7.0. GNU General Public License Version 3.

mikro>

mario@kosmos ~ mikrokosmos

Welcome to the Mikrokosmos Lambda Interpreter!

Version 0.7.0. GNU General Public License Version 3.

mikro> 2 = \s.\z.s (s z)

mikro>

mario@kosmos ~ mikrokosmos

Welcome to the Mikrokosmos Lambda Interpreter!

Version 0.7.0. GNU General Public License Version 3.

mikro> 2 = \s.\z.s (s z)

mikro> plus = \n.\m.n succ m

mikro> plus 3 4

$\lambda a.\lambda b.a (a (a (a (a (a (a b)))))) \Rightarrow 7$

mikro>

Welcome to the Mikrokosmos Lambda Interpreter!

Version 0.7.0. GNU General Public License Version 3.

```
mikro> 2 = \s.\z.s (s z)
```

```
mikro> plus = \n.\m.n succ m
```

```
mikro> plus 3 4
```

```
λa.λb.a (a (a (a (a (a (a b)))))) ⇒ 7
```

```
mikro> sum = fold plus 0
```

```
mikro> sum (cons 1 (cons 2 (cons 3 nil)))
```

```
λa.λb.a (a (a (a (a (a b)))))) ⇒ 6
```

```
mikro> █
```

Welcome to the Mikrokosmos Lambda Interpreter!

Version 0.7.0. GNU General Public License Version 3.

```
mikro> 2 = \s.\z.s (s z)
```

```
mikro> plus = \n.\m.n succ m
```

```
mikro> plus 3 4
```

```
λa.λb.a (a (a (a (a (a (a b)))))) ⇒ 7
```

```
mikro> sum = fold plus 0
```

```
mikro> sum (cons 1 (cons 2 (cons 3 nil)))
```

```
λa.λb.a (a (a (a (a (a b))))) ⇒ 6
```

```
mikro> naturals := fix (compose (cons 0) (map (plus 1)))
```

```
mikro> sum (take 5 naturals)
```

```
λa.λb.a (a (a (a (a (a (a (a (a (a b)))))))) ⇒ 10
```

```
mikro> 
```


Welcome to the Mikrokosmos Lambda Interpreter!

Version 0.7.0. GNU General Public License Version 3.

```
mikro> 2 = \s.\z.s (s z)
```

```
mikro> plus = \n.\m.n succ m
```

```
mikro> plus 3 4
```

```
λa.λb.a (a (a (a (a (a (a b)))))) ⇒ 7
```

```
mikro> sum = fold plus 0
```

```
mikro> sum (cons 1 (cons 2 (cons 3 nil)))
```

```
λa.λb.a (a (a (a (a (a b))))) ⇒ 6
```


```
mikro> naturals := fix (compose (cons 0) (map (plus 1)))
```

```
mikro> sum (take 5 naturals)
```

```
λa.λb.a (a (a (a (a (a (a (a (a (a (a b)))))))))) ⇒ 10
```

```
mikro> mu (\n.eq (mult 3 n) (plus 6 n))
```

```
λa.λb.a (a (a b)) ⇒ 3
```

```
mikro> 
```

Cálculo lambda tipado: reglas de tipado

El cálculo lambda simplemente tipado consta de tres constructores de tipos: un tipo **unidad** con un solo elemento 1, un **producto** para cada dos tipos, y el tipo **función** entre dos tipos cualquiera.

$$\begin{array}{c} \frac{}{\Gamma \vdash * : 1} \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \times B} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_1 m : A} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_2 m : B} \\[2ex] \frac{\Gamma, x : A \vdash m : B}{\Gamma \vdash \lambda x. m : A \rightarrow B} \quad \frac{\Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash f a : B} \end{array}$$

Cálculo lambda tipado: reglas de tipado

El cálculo lambda simplemente tipado consta de tres constructores de tipos: un tipo **unidad** con un solo elemento 1, un **producto** para cada dos tipos, y el tipo **función** entre dos tipos cualquiera. Además, podemos añadir un tipo **vacío** 0 y un tipo **unión** $A + B$.

$$\begin{array}{c} \frac{}{\Gamma \vdash * : 1} \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \times B} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_1 m : A} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_2 m : B} \\[10pt] \frac{\Gamma, x : A \vdash m : B}{\Gamma \vdash \lambda x. m : A \rightarrow B} \quad \frac{\Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash f a : B} \\[10pt] \frac{\Gamma \vdash m : 0}{\Gamma \vdash \text{abort}_A m : A} \quad \frac{\Gamma \vdash a : A}{\Gamma \vdash \text{inl } a : A + B} \quad \frac{\Gamma \vdash b : B}{\Gamma \vdash \text{inr } b : A + B} \\[10pt] \frac{\Gamma \vdash m : A + B \quad \Gamma, a : A \vdash n : C \quad \Gamma, b : B \vdash p : C}{\Gamma \vdash (\text{case } m \text{ of } [a].n; [b].p) : C} \end{array}$$

Cálculo lambda tipado: reglas de tipado

El cálculo lambda simplemente tipado consta de tres constructores de tipos: un tipo **unidad** con un solo elemento 1, un **producto** para cada dos tipos, y el tipo **función** entre dos tipos cualquiera. Además, podemos añadir un tipo **vacío** 0 y un tipo **unión** $A + B$.

$$\begin{array}{c} \frac{}{\Gamma \vdash * : 1} \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \times B} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_1 m : A} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_2 m : B} \\[10pt] \frac{\Gamma, x : A \vdash m : B}{\Gamma \vdash \lambda x. m : A \rightarrow B} \quad \frac{\Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash f a : B} \\[10pt] \frac{\Gamma \vdash m : 0}{\Gamma \vdash \text{abort}_A m : A} \quad \frac{\Gamma \vdash a : A}{\Gamma \vdash \text{inl } a : A + B} \quad \frac{\Gamma \vdash b : B}{\Gamma \vdash \text{inr } b : A + B} \\[10pt] \frac{\Gamma \vdash m : A + B \quad \Gamma, a : A \vdash n : C \quad \Gamma, b : B \vdash p : C}{\Gamma \vdash (\text{case } m \text{ of } [a].n; [b].p) : C} \end{array}$$

Teorema (Tait, Girard)

Todo término tipado normaliza.

Heyting y Kolmogorov describen las implicaciones de la lógica intuicionista $A \rightarrow B$ como funciones que transforman demostraciones de A en transformaciones de B .

Heyting y Kolmogorov describen las implicaciones de la lógica intuicionista $A \rightarrow B$ como funciones que transforman demostraciones de A en transformaciones de B . Bajo esta interpretación el cálculo lambda es un sistema de demostraciones de la lógica proposicional intuicionista.

$$\begin{array}{c} \frac{}{\Gamma \vdash * : 1} \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \times B} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_1 m : A} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_2 m : B} \\[10pt] \frac{\Gamma, x : A \vdash m : B}{\Gamma \vdash \lambda x. m : A \rightarrow B} \quad \frac{\Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash f a : B} \\[10pt] \frac{\Gamma \vdash m : 0}{\Gamma \vdash \text{abort}_A m : A} \quad \frac{\Gamma \vdash a : A}{\Gamma \vdash \text{inl } a : A + B} \quad \frac{\Gamma \vdash b : B}{\Gamma \vdash \text{inr } b : A + B} \\[10pt] \frac{\Gamma \vdash m : A + B \quad \Gamma, a : A \vdash n : C \quad \Gamma, b : B \vdash p : C}{\Gamma \vdash (\text{case } m \text{ of } [a].n; [b].p) : C} \end{array}$$

Heyting y Kolmogorov describen las implicaciones de la lógica intuicionista $A \rightarrow B$ como funciones que transforman demostraciones de A en transformaciones de B . Bajo esta interpretación el cálculo lambda es un sistema de demostraciones de la lógica proposicional intuicionista.

$$\begin{array}{c}
 \frac{}{\Gamma \vdash * : 1} \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \times B} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_1 m : A} \quad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_2 m : B} \\
 \\
 \frac{\Gamma, x : A \vdash m : B}{\Gamma \vdash \lambda x. m : A \rightarrow B} \quad \frac{\Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash f a : B} \\
 \\
 \frac{\Gamma \vdash m : 0}{\Gamma \vdash \text{abort}_A m : A} \quad \frac{\Gamma \vdash a : A}{\Gamma \vdash \text{inl } a : A + B} \quad \frac{\Gamma \vdash b : B}{\Gamma \vdash \text{inr } b : A + B} \\
 \\
 \frac{\Gamma \vdash m : A + B \quad \Gamma, a : A \vdash n : C \quad \Gamma, b : B \vdash p : C}{\Gamma \vdash (\text{case } m \text{ of } [a].n; [b].p) : C}
 \end{array}$$

Teorema (Curry, Howard)

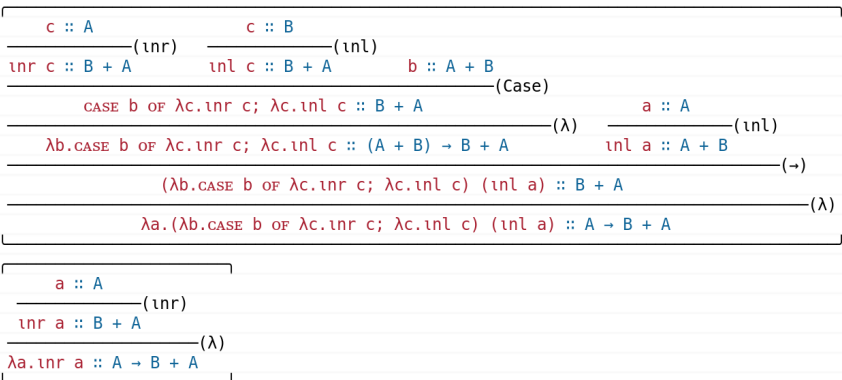
Las proposiciones son tipos, las demostraciones sus elementos. Evaluar elementos equivale a simplificar demostraciones manteniendo su significado.

Mikrokosmos: intérprete tipado

```
1 :types on
2
3 # Draws the deduction tree
4 @ \a.((\c.Case c Of inr; inl)(INL a))
5
6 # Simplifies the deduction tree
7 @@ \a.((\c.Case c Of inr; inl)(INL a))
```

evaluate

types: on



Una **adjunción** $F \dashv G$ entre categorías \mathcal{C} y \mathcal{D} es un par de funtores $F: \mathcal{C} \rightarrow \mathcal{D}$ y $G: \mathcal{D} \rightarrow \mathcal{C}$ con una biyección natural $\varphi: \text{hom}(FX, Y) \cong \text{hom}(X, GY)$.

$$\frac{FX \xrightarrow{f} Y}{X \xrightarrow{\varphi(f)} GY}$$

Una **adjunción** $F \dashv G$ entre categorías \mathcal{C} y \mathcal{D} es un par de funtores $F: \mathcal{C} \rightarrow \mathcal{D}$ y $G: \mathcal{D} \rightarrow \mathcal{C}$ con una biyección natural $\varphi: \text{hom}(FX, Y) \cong \text{hom}(X, GY)$.

$$\frac{FX \xrightarrow{f} Y}{X \xrightarrow{\varphi(f)} GY}$$

Ejemplo (Grupos)

El funtor que crea grupos libres $F: \mathbf{Grp} \rightarrow \mathbf{Set}$, es el adjunto al funtor que a cada grupo le asigna su conjunto subyacente $[-]: \mathbf{Set} \rightarrow \mathbf{Grp}$. El homomorfismo queda determinado por la elección de imágenes de los generadores.

$$\frac{FA \xrightarrow{\phi} M}{A \xrightarrow{f} [M]}$$

Ejemplo (Productos y coproductos)

Los productos y los coproductos pueden definirse sin hacer referencia a conjuntos como adjuntos $+ \dashv \Delta \dashv \times$ para Δ el funtor diagonal.

$$\frac{X, X \longrightarrow Y, Z}{X \longrightarrow Y \times Z}$$

$$\frac{X + Y \longrightarrow Z}{X, Y \longrightarrow Z, Z}$$

Una **adjunción** $F \dashv G$ entre categorías \mathcal{C} y \mathcal{D} es un par de funtores $F: \mathcal{C} \rightarrow \mathcal{D}$ y $G: \mathcal{D} \rightarrow \mathcal{C}$ con una biyección natural $\varphi: \text{hom}(FX, Y) \cong \text{hom}(X, GY)$.

$$\frac{FX \xrightarrow{f} Y}{X \xrightarrow{\varphi(f)} GY}$$

Ejemplo (Grupos)

El functor que crea grupos libres $F: \mathbf{Grp} \rightarrow \mathbf{Set}$, es el adjunto al functor que a cada grupo le asigna su conjunto subyacente $\lfloor - \rfloor: \mathbf{Set} \rightarrow \mathbf{Grp}$. El homomorfismo queda determinado por la elección de imágenes de los generadores.

$$\frac{FA \xrightarrow{\phi} M}{A \xrightarrow{f} \lfloor M \rfloor}$$

Ejemplo (Productos y coproductos)

Los productos y los coproductos pueden definirse sin hacer referencia a conjuntos como adjuntos $+ \dashv \Delta \dashv \times$ para Δ el functor diagonal.

$$\frac{X \rightarrow Y \quad X \rightarrow Z}{X \longrightarrow Y \times Z}$$

$$\frac{X + Y \longrightarrow Z}{X \rightarrow Z \quad Y \rightarrow Z}$$

Una **categoría cartesiana** \mathcal{C} es aquella en la que el funtor terminal $*$: $\mathcal{C} \rightarrow 1$, el funtor diagonal $\Delta: \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ y sus funtores producto $(- \times A): \mathcal{C} \rightarrow \mathcal{C}$ tienen adjuntos derechos. Los llamamos **unidad**, **producto** y **exponencial**.

$$\frac{* \longrightarrow *}{\frac{}{C \xrightarrow{!} 1}} \qquad \frac{C, C \xrightarrow{f,g} A, B}{\frac{}{C \xrightarrow{\langle f,g \rangle} A \times B}} \qquad \frac{C \times A \xrightarrow{f} B}{\frac{}{C \xrightarrow{\tilde{f}} B^A}}$$

Categorías cartesianas cerradas: definición

Una **categoría cartesiana** \mathcal{C} es aquella en la que el funtor terminal $*$: $\mathcal{C} \rightarrow 1$, el funtor diagonal Δ : $\mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ y sus funtores producto $(- \times A)$: $\mathcal{C} \rightarrow \mathcal{C}$ tienen adjuntos derechos. Los llamamos **unidad**, **producto** y **exponencial**.

$$\frac{* \longrightarrow *}{C \xrightarrow{!} 1} \qquad \frac{C, C \xrightarrow{f,g} A, B}{C \xrightarrow{\langle f,g \rangle} A \times B} \qquad \frac{C \times A \xrightarrow{f} B}{C \xrightarrow{\tilde{f}} B^A}$$

Si interpretamos los tipos y contextos como objetos y los elementos como morfismos, son las reglas para el cálculo lambda tipado.

$$\frac{}{\Gamma \vdash * : 1} \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \times B} \qquad \frac{\Gamma, a : A \vdash b : B}{\Gamma \vdash (\lambda a. b) : A \rightarrow B}$$

Teorema (Lambek)

Hay una equivalencia entre categorías cartesianas cerradas y teorías sobre el cálculo lambda CCC $\simeq \lambda\text{Th}$.

Teorema (Lawvere)

En una categoría cartesiana cerrada, si existe un $d : A \rightarrow B^A$ sobreyectivo en puntos, cada $f : B \rightarrow B$ tiene un punto fijo $b : B$, tal que $f(b) = b$.

Demostración. Por sobreyectividad, existe $d\ x = \lambda a.f(d\ a\ a)$. Entonces $d\ x\ x \equiv (\lambda a.f\ (d\ a\ a))\ x \equiv f\ (d\ x\ x)$ es un punto fijo. \square

Categorías cartesianas cerradas: argumentos diagonales

Teorema (Lawvere)

En una categoría cartesiana cerrada, si existe un $d : A \rightarrow B^A$ sobreyectivo en puntos, cada $f : B \rightarrow B$ tiene un punto fijo $b : B$, tal que $f(b) = b$.

Demostración. Por sobreyectividad, existe $d\ x = \lambda a.f(d\ a\ a)$. Entonces $d\ x\ x \equiv (\lambda a.f\ (d\ a\ a))\ x \equiv f\ (d\ x\ x)$ es un punto fijo. \square

Ejemplo (Cantor)

Para todo A , el conjunto 2^A tiene mayor cardinalidad.

Ejemplo (Russell)

Tomar toda colección como conjunto lleva a inconsistencia.

Ejemplo (Combinador de punto fijo)

El cálculo lambda sin tipos puede verse como una teoría sobre el cálculo tipado. Todo combinador tiene un punto fijo.

Ejemplo (Tarski, Gödel)

Una teoría consistente no puede expresar su propia verdad.

Consideramos objetos $2, A^0, A^1, A^2, \dots$ y los morfismos entre ellos son expresiones de la teoría. La teoría es **consistente** si existe $\text{not} : 2 \rightarrow 2$ sin puntos fijos. La **verdad** es expresable si existe $\text{truth} : A \times A \rightarrow 2$ tal que para cada $\varphi : A \rightarrow 2$ existe un $g : A$ **número de Gödel** tal que $\text{truth}(g, a) = \varphi(a)$.

Un **álgebra inicial** sobre un funtor viene dada por una construcción con la siguiente propiedad universal. Un ejemplo son los números naturales.

$$\begin{array}{ccc}
 FX & \xrightarrow{Fh} & FY \\
 \downarrow \mu & & \downarrow \nu \\
 X & \xrightarrow{h} & Y
 \end{array}
 \qquad
 \begin{array}{ccc}
 1 + \mathbb{N} & \longrightarrow & 1 + X \\
 \downarrow \langle 0, \text{succ} \rangle & & \downarrow \langle x, f \rangle \\
 \mathbb{N} & \xrightarrow{\varphi} & X
 \end{array}$$

Lo interesante es que así ganamos una forma de expresar los naturales y los principios de inducción. El cálculo lambda que se obtiene cuando añadimos los naturales se llama **System T** y fue desarrollado por Gödel.

$$\begin{array}{ccc}
 1 + \mathbb{N} & \longrightarrow & 1 + \text{hom}(\mathbb{N}, \mathbb{N}) \\
 \downarrow \langle 0, \text{succ} \rangle & & \downarrow \langle \text{id}, \text{succ} \circ - \rangle \\
 \mathbb{N} & \xrightarrow{+} & \text{hom}(\mathbb{N}, \mathbb{N})
 \end{array}$$

Nos da $0 + m = (m) = m$ y además

$\text{succ}(n) + m = (\text{succ} \circ (n + _))(m) = \text{succ}(n + m)$.

Categorías localmente cartesianas cerradas: cuantificadores

Los productos fibrados determinan **sustituciones**. Un caso particular es el **debilitamiento lógico**. Determina un funtor entre **sobrecategorías**.

$$\begin{array}{ccc} \{P(\pi(a, b))\} & \longrightarrow & \{P(a)\} \\ \downarrow & & \downarrow \\ A \times B & \xrightarrow{\pi} & A \end{array}$$

Este debilitamiento tiene dos adjuntos en sobrecategorías, $\exists \dashv \pi \dashv \forall$, que son los cuantificadores lógicos.

$$\begin{array}{ccc} & A \times B & \\ \nearrow & & \nwarrow \\ \{P(\pi(a, b))\} & \xrightarrow{\quad} & \{Q(a, b)\} \\ \hline \{P(a)\} & \xrightarrow{\quad} & \{\forall b \in B, Q(a, b)\} \\ \searrow & & \swarrow \\ & A & \end{array}$$

Teorema

Si existen los productos fibrados, la adjunción izquierda, **existe**, viene dada por composición con π . Si además existe la adjunción derecha, **para todo**, llamamos a la categoría **localmente cartesiana cerrada**.

Categorías localmente cartesianas cerradas: cuantificadores

Los productos fibrados determinan **sustituciones**. Un caso particular es el **debilitamiento lógico**. Determina un funtor entre **sobrecategorías**.

$$\begin{array}{ccc} \{P(\pi(a, b))\} & \longrightarrow & \{P(a)\} \\ \downarrow & & \downarrow \\ A \times B & \xrightarrow{\pi} & A \end{array}$$

Este debilitamiento tiene dos adjuntos en sobrecategorías, $\exists \dashv \pi \dashv \forall$, que son los cuantificadores lógicos.

$$\begin{array}{ccc} & A \times B & \\ \nearrow & & \nwarrow \\ \{P(a, b)\} & \xrightarrow{\quad} & \{Q(\pi(a, b))\} \\ \hline \{ \exists b \in B: P(a) \} & \xrightarrow{\quad} & \{Q(a)\} \\ \searrow & & \swarrow \\ & A & \end{array}$$

Teorema

Si existen los productos fibrados, la adjunción izquierda, **existe**, viene dada por composición con π . Si además existe la adjunción derecha, **para todo**, llamamos a la categoría **localmente cartesiana cerrada**.

Categorías localmente cartesianas cerradas: sigma

Vamos a construir \exists en el cálculo lambda. Un tipo es un objeto de la sobrecategoría sobre un contexto, $\mathcal{C}/\llbracket \Gamma \rrbracket$. Sus elementos son morfismos desde el terminal de la sobrecategoría, $\text{id}: \llbracket \Gamma \rrbracket \rightarrow \llbracket \Gamma \rrbracket$.

$$\begin{array}{ccc} \llbracket \Gamma, a : A \rrbracket & & \llbracket \Gamma \rrbracket \xrightarrow{a} \llbracket \Gamma, a : A \rrbracket \\ \downarrow \pi_A & & \searrow \parallel \downarrow \pi_A \\ \llbracket \Gamma \rrbracket & & \llbracket \Gamma \rrbracket \end{array}$$

Esta construcción es un **par dependiente** o **tipo Sigma**.

$$\begin{array}{ccccc} & & \langle a, b \rangle & & \\ & \swarrow & & \searrow & \\ \llbracket \Gamma \rrbracket & \xrightarrow{b} & \llbracket \Gamma, y : B(a) \rrbracket & \longrightarrow & \llbracket \Gamma, x : A, y : B \rrbracket \\ & \searrow \parallel & \downarrow & & \downarrow \pi_B \\ & & \llbracket \Gamma \rrbracket & \xrightarrow{a} & \llbracket \Gamma, x : A \rrbracket \\ & & & \searrow \parallel & \downarrow \pi_A \\ & & & & \llbracket \Gamma \rrbracket \end{array} \quad \begin{array}{l} \nearrow \pi_\Sigma \end{array}$$

La construcción del **cuantificador universal** son las **funciones dependientes** o **tipos Π** . Es directa desde la adjunción suponiendo que estamos en una categoría localmente cartesiana cerrada.

$$\begin{array}{c}
 \begin{array}{ccc}
 & \curvearrowright \llbracket \Gamma, A \rrbracket \curvearrowleft & \\
 \llbracket \Gamma, A \rrbracket & \xrightarrow{b} & \llbracket \Gamma, A, B \rrbracket \\
 \hline
 \llbracket \Gamma \rrbracket & \xrightarrow{(\lambda a. b)} & \llbracket \Gamma, \prod_{a:A} B \rrbracket \\
 & \curvearrowleft \llbracket \Gamma \rrbracket \curvearrowright &
 \end{array}
 \qquad
 \begin{array}{ccc}
 & \curvearrowright \llbracket \Gamma, A \rrbracket \curvearrowleft & \\
 \llbracket \Gamma, A, \prod_{a:A} B \rrbracket & \xrightarrow{app} & \llbracket \Gamma, A, B \rrbracket \\
 \hline
 \llbracket \Gamma, \prod_{a:A} B \rrbracket & \xrightarrow{id} & \llbracket \Gamma, \prod_{a:A} B \rrbracket \\
 & \curvearrowleft \llbracket \Gamma \rrbracket \curvearrowright &
 \end{array}
 \end{array}$$

Nótese cómo la imagen de la identidad bajo la adjunción es la aplicación de una función sobre un elemento.

Obtenemos las siguientes reglas para un sistema de tipos dependiente.

$$\begin{array}{c}
 \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B[a/x]}{\Gamma \vdash \langle a, b \rangle : \sum_{x:A} B} \quad
 \frac{\Gamma \vdash m : \sum_{x:A} C}{\Gamma \vdash \mathbf{fst}(m) : A} \quad
 \frac{\Gamma \vdash m : \sum_{x:A} C}{\Gamma \vdash \mathbf{snd}(m) : C[(m)/a]} \\
 \\
 \frac{\Gamma, a : A \vdash b : B}{\Gamma \vdash (\lambda a. b) : \prod_{a:A} B} \quad
 \frac{\Gamma \vdash a : A \quad \Gamma \vdash f : \prod_{a:A} B}{\Gamma \vdash f \ a : B(a)}
 \end{array}$$

Ejemplo: diferencia entre vectores y vectores uniformes.

$$v : \sum_{n:\mathbb{N}} \mathbf{Vect}(n) \quad \text{y} \quad w : \prod_{n:\mathbb{N}} \mathbf{Vect}(n).$$

```
-- Definition of the sigma type as a record with two constructors.
-- Its elimination principle can be directly derived from the
-- definition.
record  $\Sigma$  (S : Set) (T : S  $\rightarrow$  Set) : Set where
  constructor _,-
  field
    fst : S
    snd : T fst
open  $\Sigma$  public

first : {A : Set} {B : A  $\rightarrow$  Set} {C : Set}  $\rightarrow$   $\Sigma$  A B  $\rightarrow$  A
first (a , b) = a

second : {A : Set} {B : A  $\rightarrow$  Set} {C : Set}  $\rightarrow$  (r :  $\Sigma$  A B)  $\rightarrow$  B (fst r)
second (a , b) = b
```

- 1.8k Base.agda Agda :Checked Git:master Mod unix | 57: 0 44%

% 0 *All Done* AgdaInfo unix | 1: 0 All

La **igualdad** será el tipo dado por la diagonal $\Delta: A \rightarrow A \times A$. La propiedad universal del producto fibrado nos da una equivalencia entre los siguientes diagramas con la diagonal.

$$\begin{array}{ccc} \Delta^* C & \longrightarrow & C \\ \tilde{k} \downarrow \scriptstyle \nearrow & & \downarrow \scriptstyle \pi \\ A & \xrightarrow{\Delta} & A \times A \end{array}$$

$$\begin{array}{ccc} A & \overset{k}{\dashrightarrow} & C \\ \Delta \searrow & & \swarrow \pi \\ & A \times A & \end{array}$$

Cuando lo interpretamos en tipos, a la izquierda tenemos un $x : A \vdash c : C(x, x)$ y a la derecha tenemos $x : A, y : A, p : x = y \vdash c : C(x, y)$. Esto nos da una regla para usar la igualdad, que en teoría de tipos se llama **eliminador J**.

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma, x : A \vdash c : C(x, x) \qquad \Gamma \vdash p : a = b}{\Gamma \vdash J_C(c, p) : C(a, b)}$$

```
-- Equality is defined as an inductive type. Its induction principle
-- is the J-eliminator.
data _==_ {ℓ} {A : Type ℓ} : A → A → Type ℓ where
  refl : (a : A) → a == a

-- Composition of paths
infixl 50 _·_
_·_ : ∀{ℓ} {A : Type ℓ} {a b c : A} → a == b → b == c → a == c
refl a · q = q
```

– 1.9k agda-hott/Base.agda Agda :Checked ▮ unix | 70: 0 Bottom

% 0 *All Done* AgdaInfo ▮ unix | 1: 0 All

Un **subobjeto clasificador** es un Ω con un monomorfismo $\text{true} : 1 \rightarrow \Omega$ tal que, para todo monomorfismo $m : \llbracket \Gamma, P \rrbracket \rightarrow \llbracket \Gamma \rrbracket$, existe un único χ tal que el siguiente diagrama es un producto fibrado.

$$\begin{array}{ccc} \llbracket \Gamma, x : P \rrbracket & \longrightarrow & 1 \\ \downarrow & & \downarrow \text{true} \\ \llbracket \Gamma \rrbracket & \overset{\chi_P}{\dashrightarrow} & \Omega \end{array}$$

Los tipos dados por un monomorfismo se llaman **proposiciones** y se pueden ver como elementos del tipo Ω .

$$\frac{\Gamma \vdash P : \Omega \quad \Gamma \vdash a : P \quad \Gamma \vdash b : P}{\Gamma \vdash \text{isProp}_P(a, b) : a = b}$$

Además, mediante adjunciones hay formas de **truncar** cada tipo A en una proposición $|A|$.

Con toda la estructura considerada podemos interpretar fundaciones categóricas de las matemáticas dentro de un lenguaje de programación. Un ejemplo es la *Elementary Theory of the Category of Sets* de W. Lawvere. Axiomas:

- una categoría localmente cartesiana cerrada con todos los límites finitos, el álgebra inicial de $1 + X$ y un subobjeto clasificador (un *topos con un objeto de números naturales*);
- *punteada*, para cada $f, g: A \rightarrow B$, hay igualdad $f = g$ si y sólo si $f(a) = g(a)$ para cualquier $a: 1 \rightarrow A$;
- cumpliendo el *axioma de elección*, los morfismos sobreyectivos en puntos tienen una sección.

$$\left(\prod_{(a:A)} \left\| \sum_{(b:B)} f(b) = a \right\| \right) \rightarrow \left\| \sum_{(g:A \rightarrow B)} \prod_{(a:A)} f(g(a)) = a \right\|$$

```
-- We internally postulate well-pointedness.
```

```
postulate
```

```
wellPointed : {A : Set} {B : A → Set} → {f g : (a : A) → B a}  
  → ((x : A) → f x ≡ g x) → f ≡ g
```

```
-- We internally postulate the Axiom of Choice.
```

```
postulate
```

```
AxiomOfChoice : {A : Set} {B : Set} {R : A → B → Set}  
  → ((a : A) → (∃ b ∈ B , (R a b)))  
  -----  
  → (∃ g ∈ (A → B), ((a : A) → R a (g a)))
```

```
□
```

```
- 2.5k  Etcs.agda  Agda  :Checked  Git:master Mod  ||  unix | 35: 0  18%
```

```
% 0  *All Done*  AgdaInfo  ||  unix | 1: 0  All
```

```
Wrote /home/mario/projects/ctlc/agda-mltt/Etcs.agda
```

Teorema (Diaconescu)

El axioma de elección implica el tercio excluso.

Demostración.

Dada P , definimos $U = \{x \in \{0, 1\} \mid (x = 0) \vee P\}$ y $V = \{x \in \{0, 1\} \mid (x = 1) \vee P\}$, cada no vacío. Por elección, existe $f: \{U, V\} \rightarrow U \cup V$ tal que $f(U) \in U$ y $f(V) \in V$. Decidimos si $f(U)$ y $f(V)$ son 0 o no por inducción. Si $f(U) = 1$ o $f(V) = 0$, entonces P es cierto; y si $f(U) = 0$ y además $f(V) = 1$, tendríamos $\neg P$, porque si P , entonces $U = V$, luego $0 = f(U) = f(V) = 1$. □

Así que al aceptar axioma de elección, hemos vuelto a la matemática clásica. Esto es una posible vía, pero no la única.

postulate

```
AxiomOfChoice : {A : Set} {B : Set} {R : A → B → Set}
→ ((a : A) → (∃ b ∈ B , (R a b)))
-----
→ (∃ f ∈ (A → B), ((a : A) → R a (f a)))
```

```
LawOfExcludedMiddle : {P : Set} → P ∨ ¬ P
```

```
LawOfExcludedMiddle {P} = Ex-elim
```

```
(AxiomOfChoice ∧ { (Q , q) → Ex-elim q Ex-isProp ∧ { (u , v) → u , v } })
V-isProp ∧ { (f , α) → byCases f α }
```

where

```
A : Set
```

```
A = Σ (Bool → Set) (λ Q → Ex Bool (λ b → Q b))
```

```
R : A → Bool → Set
```

```
R (P , _) b = P b
```

```
U : Bool → Set
```

```
U b = (b ≡ true) ∨ P
```

```
V : Bool → Set
```

```
V b = (b ≡ false) ∨ P
```

* 5.1k Snippets.lagda Agda :Checked Git-master | unix | 170: 0 58%

% 0 *All Done* AgdaInfo | unix | 1: 0 All

Cada topos da un modelo de matemática constructivista.

- En el **topos de realizabilidad** de Hayland, toda función es computable y podemos estudiar computabilidad sintética y realizabilidad de Kleene
- En el **topos de Dubuc** podemos formalizar los infinitesimales y hacer geometría diferencial sintética.
- En el **topos topológico** de Johnstone, podemos razonar sobre espacios y funciones continuas.

Además, al aceptar tercio excluso perdemos la interpretación computacional de Brouwer-Heyting-Kolmogorov.

Jaap Van Oosten. Realizability: an introduction to its categorical side. Vol. 152. Elsevier, 2008.

Eduardo J Dubuc. “Integración de campos vectoriales y geometría diferencial sintética”. En: Revista de la Unión Matemática Argentina 35 (1989), págs. 151-162.

Peter T Johnstone. “On a topological topos”. En: Proceedings of the London mathematical society 3.2 (1979), págs. 237-271.

```
-- The following is a construction of the positive real numbers using
-- Dedekind cuts. A cut is a propositional predicate over the rational
-- numbers that determines which rationals are greater or equal than the
-- real number. A real number must be
--   * bounded, meaning that the cut must be inhabited;
--   * rounded, meaning that the cut must be an upper-unbounded and
--     nonclosed interval.
```

```
record  $\mathbb{R}^+$  : Set where
```

```
  constructor real
```

```
  field
```

```
    -- Dedekind cut
```

```
    cut : F → Set
```

```
    isprop : (q : F) → isProp (cut q)
```

```
    bound : ∃ q ∈ F , cut q
```

```
    round1 : (q : F) → cut q → ∃ p ∈ F , ((p < q ≡ true) × cut p)
```

```
    round2 : (q : F) → (∃ p ∈ F , ((p < q ≡ true) × cut p)) → cut q
```

```
open  $\mathbb{R}^+$  {...} public
```

* 7.7k Reals.agda Agda :Checked Git-master || unix | 29: 0 3%

% 0 *All Done* AgdaInfo || unix | 1: 0 All

```
-- The fundamental group of the circle is the integers. In this
-- proof, univalence is crucial. The next lemma will prove that the
-- equivalence in fact preserves the group structure.
```

```
fundamental-group-of-the-circle :  $\Omega S^1 \text{ base} \simeq \mathbb{Z}$ 
```

```
fundamental-group-of-the-circle = equiv-family base
```

```
preserves-composition :  $\forall n m \rightarrow \text{loops } (n +^z m) == \text{loops } n \cdot \text{loops } m$ 
```

```
preserves-composition n m = z-act+ ( $\Omega\text{-st } S^1 \text{ base}$ ) n m loop
```

```
preserves-inverses :  $\forall n \rightarrow \text{loops } (- n) == \text{inv } (\text{loops } n)$ 
```

```
preserves-inverses n = z-actm ( $\Omega\text{-st } S^1 \text{ base}$ ) n loop
```

