Teoría de categorías y cálculo lambda

Mario Román García

25 de octubre de 2018

Grado en Ingeniería Informática y Matemáticas - Universidad de Granada

Outline

1. Categorías cartesianas cerradas

2. Categorías localmente cartesianas cerradas

3. Matemática constructivista

Esquema

Categorías: adjunciones

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

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

Categorías: adjunciones

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

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

Ejemplo (Grupos)

El funtor que crea grupos libres $F\colon \mathsf{Grp} \to \mathsf{Set}$, es el adjunto al funtor que a cada grupo le asigna su conjunto subyacente $\lfloor - \rfloor\colon \mathsf{Set} \to \mathsf{Grp}$. 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 funtor diagonal.

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

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

Categorías: adjunciones

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

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

Ejemplo (Grupos)

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

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

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 funtor diagonal.

$$\frac{X \to Y \qquad X \to Z}{\overline{X \longrightarrow Y \times Z}}$$

$$\frac{X + Y \longrightarrow Z}{\overline{X \to Z \qquad Y \to Z}}$$

Categorías cartesianas cerradas: definición

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

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

Joachim Lambek y Philip J Scott. Introduction to higher-order categorical logic. Vol. 7. Cambridge University Press, 1988.

Categorías cartesianas cerradas: definición

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

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

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

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

Teorema (Lambek)

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

Joachim Lambek y Philip J Scott. Introduction to higher-order categorical logic. Vol. 7. Cambridge University Press, 1988.

Cálculo lambda tipado: interpretación de Heyting-Kolmogorov

Bajo esta interpretación el cálculo lambda es un sistema de demostraciones de la lógica proposicional intuicionista.

$$\frac{}{\Gamma \vdash * : 1} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \times B} \qquad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_1\ m : A} \qquad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_2\ m : B}$$

$$\frac{\Gamma, x : A \vdash m : B}{\Gamma \vdash \lambda x.m : A \to B} \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f\ a : B}$$

$$\frac{\Gamma \vdash m : 0}{\Gamma \vdash \text{abort}_A\ m : A} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash \text{inl}\ a : A + B} \qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \text{inr}\ b : A + B}$$

$$\frac{\Gamma \vdash m : A + B \qquad \Gamma, a : A \vdash n : C \qquad \Gamma, b : B \vdash p : C}{\Gamma \vdash (\text{case}\ m\ \text{of}\ [a].n;\ [b].p) : C}$$

Philip Wadler. “Propositions As Types”. En: Commun. ACM 58.12 (nov. de 2015), págs. 75-84. ISSN: 0001-0782. DOI: 10.1145/2699407. URL: http://doi.acm.org/10.1145/2699407.

Cálculo lambda tipado: interpretación de Heyting-Kolmogorov

Bajo esta interpretación el cálculo lambda es un sistema de demostraciones de la lógica proposicional intuicionista.

$$\frac{}{\Gamma \vdash * : 1} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \times B} \qquad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_1\ m : A} \qquad \frac{\Gamma \vdash m : A \times B}{\Gamma \vdash \pi_2\ m : B}$$

$$\frac{\Gamma, x : A \vdash m : B}{\Gamma \vdash \lambda x.m : A \to B} \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f\ a : B}$$

$$\frac{\Gamma \vdash m : 0}{\Gamma \vdash \mathrm{abort}_A\ m : A} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash \mathrm{inl}\ a : A + B} \qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathrm{inr}\ b : A + B}$$

$$\frac{\Gamma \vdash m : A + B \qquad \Gamma, a : A \vdash n : C \qquad \Gamma, b : B \vdash p : C}{\Gamma \vdash (\mathrm{case}\ m\ \mathrm{of}\ [a].n;\ [b].p) : C}$$

Teorema (Curry, Howard)

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

Philip Wadler. “Propositions As Types”. En: Commun. ACM 58.12 (nov. de 2015), págs. 75-84. ISSN: 0001-0782. DOI: 10.1145/2699407. URL: http://doi.acm.org/10.1145/2699407.

Cálculo lambda tipado: intérprete tipado

```
:types on

# Draws the deduction tree
@ \a.((\c.Case c Of inr; inl)(INL a))

# Simplifies the deduction tree
@@ \a.((\c.Case c Of inr; inl)(INL a))
```

evaluate

```
types: on
┌─────────────────────────────────────────────────────────────────────────────────┐
    c :: A                c :: B
 ────────────(ιnr)    ────────────(ιnl)
 ιnr c :: B + A       ιnl c :: B + A       b :: A + B
 ──────────────────────────────────────────────────(Case)
      CASE b OF λc.ιnr c; λc.ιnl c :: B + A                  a :: A
 ───────────────────────────────────────────────────────(λ)  ────────────(ιnl)
    λb.CASE b OF λc.ιnr c; λc.ιnl c :: (A + B) → B + A       ιnl a :: A + B
 ────────────────────────────────────────────────────────────────────────────(→)
               (λb.CASE b OF λc.ιnr c; λc.ιnl c) (ιnl a) :: B + A
 ───────────────────────────────────────────────────────────────────────────────(λ)
            λa.(λb.CASE b OF λc.ιnr c; λc.ιnl c) (ιnl a) :: A → B + A
└─────────────────────────────────────────────────────────────────────────────────┘

┌──────────────────────┐
      a :: A
  ────────────(ιnr)
  ιnr a :: B + A
 ─────────────────(λ)
 λa.ιnr a :: A → B + A
└──────────────────────┘
```

Mario Román. “Mikrokosmos: an educational lambda calculus interpreter”. En: Journal of Open Source Education 1.8 (2018), pág. 29. DOI: 10.21105/jose.00029. URL: https://doi.org/10.21105%2Fjose.00029.

Teorema (Lawvere)

En una categoría cartesiana cerrada, si existe un $d : A \to B^A$ sobreyectivo en puntos, cada $f : B \to B$ 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. $\square$

F. William Lawvere. “Diagonal arguments and Cartesian Closed Categories”. En: Reprints in Theory and Applications of Categories (2016), págs. 1-13.

Teorema (Lawvere)

En una categoría cartesiana cerrada, si existe un $d : A \to B^A$ sobreyectivo en puntos, cada $f : B \to B$ 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.

F. William Lawvere. "Diagonal arguments and Cartesian Closed Categories". En: Reprints in Theory and Applications of Categories (2016), págs. 1-13.

Categorías cartesianas cerradas: inducción

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

$$\begin{tikzcd} FX \arrow[r, dashed, "Fh"] \arrow[d, "\mu"] & FY \arrow[d, "\nu"] \\ X \arrow[r, dashed, "h"] & Y \end{tikzcd} \qquad \begin{tikzcd} 1+\mathbb{N} \arrow[r] \arrow[d, "\langle 0, \mathrm{succ} \rangle"'] & 1+X \arrow[d, "\langle x, f \rangle"] \\ \mathbb{N} \arrow[r, "\varphi"] & X \end{tikzcd}$$

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.

$$\begin{tikzcd} 1+\mathbb{N} \arrow[r] \arrow[d, "\langle 0, \mathsf{succ} \rangle"'] & 1+\hom(\mathbb{N},\mathbb{N}) \arrow[d, "\langle \mathsf{id}, \mathsf{succ} \circ - \rangle"] \\ \mathbb{N} \arrow[r, "+"] & \hom(\mathbb{N},\mathbb{N}) \end{tikzcd}$$

Nos da $0+m = (m) = m$ y además
$\mathrm{succ}(n) + m = (\mathrm{succ} \circ (n + \_))(m) = \mathrm{succ}(n+m)$.

Esquema

Categorías localmente cartesianas cerradas: cuantificadores

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

$$\begin{tikzcd}
\{P(\pi(a,b))\} \arrow[r] \arrow[d, hook] & \{P(a)\} \arrow[d, hook] \\
A \times B \arrow[r, "\pi"] & A
\end{tikzcd}$$

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

$$\begin{tikzcd}
& A \times B & \\
\{P(\pi(a,b))\} \arrow[ur] \arrow[rr] & & \{Q(a,b)\} \arrow[ul]
\end{tikzcd}$$

$$\begin{tikzcd}
\{P(a)\} \arrow[rr] \arrow[dr] & & \{\forall b \in B, Q(a,b)\} \arrow[dl] \\
& A &
\end{tikzcd}$$

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.

$$\begin{tikzcd}
\{P(\pi(a,b))\} \arrow[r] \arrow[d, hook] & \{P(a)\} \arrow[d, hook] \\
A \times B \arrow[r, "\pi"] & A
\end{tikzcd}$$

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

$$\begin{array}{c}
\begin{tikzcd}
 & A \times B & \\
\{P(a,b)\} \arrow[ur] \arrow[rr] & & \{Q(\pi(a,b))\} \arrow[ul]
\end{tikzcd} \\
\hline\hline
\begin{tikzcd}
\{\exists b \in B \colon P(a)\} \arrow[rr] \arrow[dr] & & \{Q(a)\} \arrow[dl] \\
 & A &
\end{tikzcd}
\end{array}$$

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, $\mathrm{id}\colon \llbracket\Gamma\rrbracket \to \llbracket\Gamma\rrbracket$.

$$
\begin{tikzcd}
\llbracket\Gamma, a : A\rrbracket \arrow[d, "\pi_A"] \\
\llbracket\Gamma\rrbracket
\end{tikzcd}
\qquad
\begin{tikzcd}
\llbracket\Gamma\rrbracket \arrow[r, "a"] \arrow[dr, equal] & \llbracket\Gamma, a : A\rrbracket \arrow[d, "\pi_A"] \\
& \llbracket\Gamma\rrbracket
\end{tikzcd}
$$

Esta construcción es un par dependiente o tipo Sigma.

$$
\begin{tikzcd}
\llbracket\Gamma\rrbracket \arrow[r, "b"] \arrow[rr, bend left, "\langle a,b\rangle"] \arrow[dr, equal] & \llbracket\Gamma, y : B(a)\rrbracket \arrow[r] \arrow[d] & \llbracket\Gamma, x : A, y : B\rrbracket \arrow[d, "\pi_B"] \arrow[dd, bend left, "\pi_\Sigma"] \\
& \llbracket\Gamma\rrbracket \arrow[r, "a"] \arrow[dr, equal] & \llbracket\Gamma, x : A\rrbracket \arrow[d, "\pi_A"] \\
& & \llbracket\Gamma\rrbracket
\end{tikzcd}
$$

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, $\mathrm{id}\colon \llbracket\Gamma\rrbracket \to \llbracket\Gamma\rrbracket$.

$$
\begin{tikzcd}
\llbracket\Gamma, a : A\rrbracket \arrow[d, "\pi_A"] \\
\llbracket\Gamma\rrbracket
\end{tikzcd}
\qquad
\begin{tikzcd}
\llbracket\Gamma\rrbracket \arrow[r, "a"] \arrow[dr, equal] & \llbracket\Gamma, a : A\rrbracket \arrow[d, "\pi_A"] \\
& \llbracket\Gamma\rrbracket
\end{tikzcd}
$$

Esta construcción es un par dependiente o tipo Sigma.

$$
\begin{tikzcd}
\llbracket\Gamma\rrbracket \arrow[r, "b"] \arrow[rr, bend left, "\langle a,b\rangle"] \arrow[dr, equal] & \llbracket\Gamma, y : B(a)\rrbracket \arrow[r] \arrow[d] & \llbracket\Gamma, x : A, y : B\rrbracket \arrow[d, "\pi_B"] \arrow[dd, bend left, "\pi_\Sigma"] \\
& \llbracket\Gamma\rrbracket \arrow[r, "a"] \arrow[dr, equal] & \llbracket\Gamma, x : A\rrbracket \arrow[d, "\pi_A"] \\
& & \llbracket\Gamma\rrbracket
\end{tikzcd}
$$

Categorías localmente cartesianas cerradas: sigma

Vamos a construir $\exists$ en el cálculo lambda. Un tipo es un objeto de la sobrecategoría sobre un contexto, $\mathcal{C}/[\![\Gamma]\!]$. Sus elementos son morfismos desde el terminal de la sobrecategoría, $\mathrm{id}\colon [\![\Gamma]\!] \to [\![\Gamma]\!]$.

$$
\begin{tikzcd}
{[\![\Gamma, a : A]\!]} \arrow[d, "\pi_A"] \\
{[\![\Gamma]\!]}
\end{tikzcd}
\qquad
\begin{tikzcd}
{[\![\Gamma]\!]} \arrow[r, "a"] \arrow[dr, equal] & {[\![\Gamma, a : A]\!]} \arrow[d, "\pi_A"] \\
& {[\![\Gamma]\!]}
\end{tikzcd}
$$

Esta construcción es un par dependiente o tipo Sigma.

$$
\begin{tikzcd}
{[\![\Gamma]\!]} \arrow[r, "b"] \arrow[dr, equal] \arrow[rr, bend left, "\langle a{,}b\rangle"] & {[\![\Gamma, y : B(a)]\!]} \arrow[r] \arrow[d] & {[\![\Gamma, x : A, y : B]\!]} \arrow[d, "\pi_B"] \arrow[dd, bend left, "\pi_\Sigma"] \\
& {\color{red}[\![\Gamma]\!]} \arrow[r, "\color{red}a"] \arrow[dr, equal] & {\color{red}[\![\Gamma, x : A]\!]} \arrow[d, "\pi_A"] \\
& & {[\![\Gamma]\!]}
\end{tikzcd}
$$

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, $\mathrm{id}\colon \llbracket\Gamma\rrbracket \to \llbracket\Gamma\rrbracket$.

$$
\begin{tikzcd}
\llbracket \Gamma, a : A \rrbracket \arrow[d, "\pi_A"] \\
\llbracket \Gamma \rrbracket
\end{tikzcd}
\qquad
\begin{tikzcd}
\llbracket \Gamma \rrbracket \arrow[r, "a"] \arrow[dr, equal] & \llbracket \Gamma, a : A \rrbracket \arrow[d, "\pi_A"] \\
& \llbracket \Gamma \rrbracket
\end{tikzcd}
$$

Esta construcción es un par dependiente o tipo Sigma.

$$
\begin{tikzcd}
\llbracket \Gamma \rrbracket \arrow[r, "b"] \arrow[rr, bend left, "\langle a,b \rangle"] \arrow[dr, equal] & {\color{red}\llbracket \Gamma, y : B(a) \rrbracket} \arrow[r] \arrow[d] & \llbracket \Gamma, x : A, y : B \rrbracket \arrow[d, "\pi_B"] \arrow[dd, bend left, "\pi_\Sigma"] \\
& \llbracket \Gamma \rrbracket \arrow[r, "a"] \arrow[dr, equal] & \llbracket \Gamma, x : A \rrbracket \arrow[d, "\pi_A"] \\
& & \llbracket \Gamma \rrbracket
\end{tikzcd}
$$

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, $\mathrm{id}\colon \llbracket\Gamma\rrbracket \to \llbracket\Gamma\rrbracket$.

$$
\begin{tikzcd}
\llbracket\Gamma, a : A\rrbracket \arrow[d, "\pi_A"] \\
\llbracket\Gamma\rrbracket
\end{tikzcd}
\qquad
\begin{tikzcd}
\llbracket\Gamma\rrbracket \arrow[r, "a"] \arrow[dr, equal] & \llbracket\Gamma, a : A\rrbracket \arrow[d, "\pi_A"] \\
& \llbracket\Gamma\rrbracket
\end{tikzcd}
$$

Esta construcción es un par dependiente o tipo Sigma.

$$
\begin{tikzcd}
{\color{red}\llbracket\Gamma\rrbracket} \arrow[r, "{\color{red}b}"] \arrow[rr, bend left, "{\langle a,b\rangle}"] \arrow[dr, equal] & {\color{red}\llbracket\Gamma, y : B(a)\rrbracket} \arrow[r] \arrow[d] & \llbracket\Gamma, x : A, y : B\rrbracket \arrow[d, "\pi_B"] \arrow[dd, bend left, "\pi_\Sigma"] \\
& \llbracket\Gamma\rrbracket \arrow[r, "a"] \arrow[dr, equal] & \llbracket\Gamma, x : A\rrbracket \arrow[d, "\pi_A"] \\
& & \llbracket\Gamma\rrbracket
\end{tikzcd}
$$

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

first : {A : Set} {B : A → Set} {C : Set} → Σ A B → A
first (a , b) = a

second : {A : Set} {B : A → Set} {C : Set} → (r : Σ A B) → 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
```

Obtenemos las siguientes reglas para un sistema de tipos dependiente.

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash \langle a, b \rangle : \sum_{x:A} B} \qquad \frac{\Gamma \vdash m : \sum_{x:A} C}{\Gamma \vdash \mathsf{fst}(m) : A} \qquad \frac{\Gamma \vdash m : \sum_{x:A} C}{\Gamma \vdash \mathsf{snd}(m) : C[(m)/a]}$$

$$\frac{\Gamma, a : A \vdash b : B}{\Gamma \vdash (\lambda a.b) : \prod_{a:A} B} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash f : \prod_{a:A} B}{\Gamma \vdash f\ a : B(a)}$$

Ejemplo: diferencia entre vectores y vectores uniformes.

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

Per Martin-Löf. "An intuitionistic theory of types: Predicative part". En: Studies in Logic and the Foundations of Mathematics. Vol. 80. Elsevier, 1975, págs. 73-118.

Teoría de tipos: igualdad

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

$$
\begin{tikzcd}
\triangle^* C \arrow[r] \arrow[d] & C \arrow[d, "\pi"] \\
A \arrow[r, "\triangle"] \arrow[u, dashed, bend left, "\widetilde{k}"] & A \times A
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
A \arrow[rr, dashed, "k"] \arrow[dr, "\triangle"'] & & C \arrow[dl, "\pi"] \\
& A \times A &
\end{tikzcd}
$$

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.

$$
\frac{\begin{array}{cc} \Gamma \vdash a : A & \Gamma \vdash b : A \\ \Gamma, x : A \vdash c : C(x,x) & \Gamma \vdash p : a = b \end{array}}{\Gamma \vdash J_C(c,p) : C(a,b)}
$$

F. William Lawvere. "Equality in hyperdoctrines and comprehension schema as an adjoint functor". En: Applications of Categorical Algebra 17 (1970), págs. 1-14.

```agda
-- 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

Teoría de tipos: proposiciones

Un subobjeto clasificador es un Ω con un monomorfismo $\mathrm{true}\colon 1 \to \Omega$ tal que, para todo monomorfismo $m : [\![\Gamma, P]\!] \to [\![\Gamma]\!]$, existe un único χ tal que el siguiente diagrama es un producto fibrado.

$$\begin{tikzcd}
[\![\Gamma, x : P]\!] \arrow[r] \arrow[d, hook] & 1 \arrow[d, "\mathrm{true}"] \\
[\![\Gamma]\!] \arrow[r, dashed, "\chi_P"] & \Omega
\end{tikzcd}$$

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

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

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

Esquema

1. Categorías cartesianas cerradas

2. Categorías localmente cartesianas cerradas

3. Matemática constructivista

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 $f, g \colon A \to B$, hay igualdad $f = g$ si y sólo si $f(a) = g(a)$ para cualquier $a \colon 1 \to A$;
- cumpliendo el axioma de elección, los morfismos sobreyectivos en puntos tienen una sección.

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

F. William Lawvere. "An elementary theory of the category of sets". En: Proceedings of the national academy of sciences 52.6 (1964), págs. 1506-1511.

```
-- 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
```

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 $f\colon \{U, V\} \to U \cup V$ 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.

Radu Diaconescu. “Change of base for toposes with generators”. En: Journal of Pure and Applied Algebra 6.3 (1975), págs. 191-218.

```
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 λ { (Q , q) → Ex-elim q Ex-isProp λ { (u , v) → u ,, v  }})
   ∨-isProp λ { (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
```

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 Brower-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 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 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 : Ω S¹ base ≃ ℤ
fundamental-group-of-the-circle = equiv-family base

preserves-composition : ∀ n m → loops (n +ᶻ m) == loops n · loops m
preserves-composition n m = z-act+ (Ω-st S¹ base) n m loop

preserves-inverses : ∀ n → loops (- n) == inv (loops n)
preserves-inverses n = z-actm (Ω-st S¹ base) n loop
```

```
- 4.9k  FundGroupCircle.agda  Agda  :Checked    unix | 136: 0  Bottom
```

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