Realizabilidad clásica y efectos colaterales: Extendiendo la correspondencia de Curry-Howard

Étienne Miquey etienne.miquey@inria.fr

IMERL, Facultad de Ingeniería | IRIF, Université Paris Diderot

¿Qué es una demostración?

En matemática, los teoremas se demuestran con una prueba, razonando desde ciertas hipótesis para llegar a la esperada conclusión. Sin embargo, la noción de **prueba** tiene que ser definida formalmente, lo cual llevó ¡miles de años! Para entender qué es una prueba, podemos examinar un silogismo famoso:

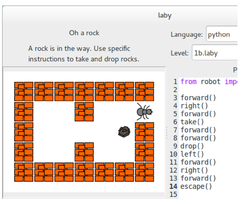
Sócrates es un gato. *Todos los gatos son mortales* .*Entonces,* *Sócrates es mortal*.
(*hipótesis*) (*conclusión*)

Intuitivamente, nuestro razonamiento sigue el proceso siguiente:

partiendo de ***hipótesis***
usamos ***reglas de deducción***
para conseguir un ***teorema***

Eso sugiere la necesidad de definir formalmente cuales son los axiomas y las reglas de deducción que consideramos válidos para razonar: es lo que llamamos una **teoría**.

¿Qué es un programa?



En Laby, un software pensado para que los niños descubran la programación, una hormiga está trancada en un laberinto y uno tiene que darle instrucciones para que se pueda escapar. La secuencia de esas instrucciones constituye en sí un **programa**. Intuitivamente, un programa se podría resumir en:

dado unos ***argumentos***
aplicar ***instrucciones***
para llegar al ***resultado***

Obviamente, las instrucciones disponibles así como la forma de componerlas dependen del **lenguaje de programación** elegido.

Secuentes y árboles de pruebas

Fue solamente a principios del siglo XX que aparecieron las primeras definiciones formales de pruebas. Entre ellas, se debe al matemático alemán Gentzen la noción de secuentes. Suponemos que nuestro lenguaje de fórmulas está constituido por fórmulas de base (atómicas), que notamos X, y por implicaciones de dos fórmulas A y B, que escribimos $A \Rightarrow B$. Un secuente, escrito $\Gamma \vdash A$, está definido por una lista de hipótesis Γ y una conclusión A:

Hipótesis $\Gamma \vdash A$ Conclusión

$A, B ::= X \mid A \Rightarrow B$ *(Fórmulas)*
$\Gamma ::= \varepsilon \mid \Gamma, A$ *(Hipótesis)*

Para probar que un secuente es demostrable, se usan reglas de deducción, que son de la forma:

$$\frac{J_1 \quad \dots \quad J_n}{J}\ (bla)$$

donde *bla* es el nombre de la regla, donde el secuente J es la conclusión de la regla y los secuentes $J_1, \dots, J_n$ son sus premisas. Por ejemplo, podemos considerar las reglas de deducción siguiente para nuestro lenguaje de fórmulas:

$$\frac{}{A \in \Gamma}\ (\text{Ax}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ (\Rightarrow_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\Rightarrow_E)$$

La regla (Ax) dice que todos los axiomas son demostrables, la regla $(\Rightarrow_I)$ se lee "*para probar $A \Rightarrow B$ con las hipótesis en Γ, basta con probar B con las hipótesis Γ, A*", mientras que la regla $(\Rightarrow_E)$, conocida cómo regla del *modus ponens*, dice que habiendo probado $A \Rightarrow B$ y A, uno puede deducir B.

Componiendo esas reglas, uno construye un **árbol de prueba**, cuyas ramas tienen que terminar usando la regla (Ax). Por ejemplo, volviendo a nuestro ejemplo, lo podemos expresar mediante el árbol siguiente:

$$\Gamma \begin{cases} \textit{Sócrates es un gato.} \\ \textit{Si Sócrates es un gato, Sócrates es mortal.} \\ \textit{Entonces, Sócrates es mortal}. \end{cases} \qquad \dfrac{\dfrac{(A \Rightarrow B) \in \Gamma}{\Gamma \vdash A \Rightarrow B}\ (\text{Ax}) \quad \dfrac{A \in \Gamma}{\Gamma \vdash A}\ (\text{Ax})}{\Gamma \vdash B}\ (\Rightarrow_E)$$

Conclusión

Formalmente, una **teoría** matemática se define por tres componentes:

1. un lenguaje de fórmulas 2. un sistema de deducción 3. axiomas

Dada una teoría, las formulas que son demostrables a partir de los axiomas construyendo un árbol de prueba se llaman **teoremas**. Entre otras propiedades que una teoría debe tener, es fundamental asegurarse que *falso* no es un teorema. De ser así, se dice que la teoría es **consistente** (y sino inconsistente).

Consistencia

Una teoría $\mathcal{T}$ es consistente si no permite derivar una prueba de falso.

Aunque pueda parecer sorprendente, no es siempre posible determinar si una formula es un teorema o no. De hecho, Gödel demostró que para una clase amplia de teorías (y en la práctica todas las que los matemáticos usan) existen fórmulas A tales que ni A ni su negación $\neg A$ son demostrables: es su famoso *teorema de incompletitud.*

El λ-cálculo

El λ-cálculo, desarrollado por A. Church en los años 30, es un modelo de computación a su vez muy sencillo (se expresa con solamente tres construcciones sintácticas) y muy potente: es *Turing-completo*, es decir que es suficiente para llevar a cabo cualquier computación que se pueda hacer en cualquier lenguaje de programación. Formalmente, sus términos son definidos por:

$$t, u ::= x \mid \lambda x.t \mid t\ u \qquad \textit{(Términos)}$$

donde x es una variable, $\lambda x.t$ una función asociando t (que depende de x) a cada x, y $t\ u$ es la aplicación del término t al término u. Intuitivamente, uno puede por ejemplo pensar en x cómo siendo la variable de un polinomio $P(x)$, $\lambda x.t$ siendo la función $x \mapsto P(x)$ (si t representa P). De forma interesante, imaginando que u represente al entero 2, $(\lambda x.t)\ u$ corresponde a la aplicación formal de $x \mapsto P(x)$ a 2, pero no a $P(2)$ (P donde x ha sido sustituido por 2). Para llegar a ese resultado, se requiere un paso de computación, llamado *β-reducción* y definido por:

$$(\lambda x.t)\ u \longrightarrow_\beta t[u/x] \qquad \textit{(β-reducción)}$$

(donde $t[u/x]$ es el término t en el cual u remplaza x). Por ejemplo, es fácil comprobar que:

$$(\lambda x.x)\ t \longrightarrow_\beta t \qquad (\lambda x.\lambda y.y\ x)\ u\ t \longrightarrow_\beta (\lambda y.y\ u)\ t \longrightarrow_\beta t\ u$$
$$(\lambda x.x\ x)\ (\lambda x.x\ x) \longrightarrow_\beta (\lambda x.x\ x)\ (\lambda x.x\ x) \longrightarrow_\beta \dots$$

Observen que el último término se puede reducir indefinidamente. Para impedir esos comportamientos, una forma de restringir el cálculo es de considerar términos con una información extra, el **tipo**, que especifica la estructura de un término. El sistema de tipos lo más sencillo, los **tipos simples**, solo incluye dos categorías: los tipos atómicos, notados X, y los tipos de funciones de A hacia B, notados $A \to B$. Se consideran los secuentes siguientes:

Juicio de tipaje: $\Gamma \vdash t : A$

$A, B ::= X \mid A \to B$ *(Tipos simples)*
$\Gamma ::= \varepsilon \mid \Gamma, x : A$ *(Hipótesis)*

Las reglas de tipaje correspondiente para el λ-cálculo simplemente tipado son:

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B} \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

Por ejemplo, se puede comprobar que el término que intuitivamente representa la composición de funciones $(f, g \mapsto f \circ g)$ tiene el tipo correspondiente:

$$\frac{\dfrac{}{f : A \to B \vdash \lambda g.\lambda x.g\ (f\ x) : (B \to C) \to (A \to C)}\ (\to_I)}{\vdash \lambda f.\lambda g.\lambda x.g\ (f\ x) : (A \to B) \to (B \to C) \to (A \to C)}\ (\to_I)$$

De forma interesante, los términos simplemente tipados cumplen ciertas propiedades, cómo por ejemplo el hecho que sus reducciones siempre terminan (*normalizan*) o que los tipos son preservados por la β-reducción:

Subject reduction	**Normalización**
Si $\Gamma \vdash t : A$ y $t \longrightarrow_\beta t'$, $\Gamma \vdash t' : A$.	Si $\Gamma \vdash t : A$, entonces t normaliza.

La correspondencia de Curry-Howard

Una flagrante semejanza Observando las reglas de la deducción natural y las de tipaje para el λ-cálculo con tipos simples, es muy fácil de ver que comparten la misma estructura: borrando los términos de las reglas de tipaje se obtiene exactamente las reglas de deducción. Además de ver los λ-términos como términos representando funciones matemáticas, podemos también considerarlos como **términos de prueba**. Observando la regla (λ), uno la puede leer como: si t es una prueba de B bajo la hipótesis de una prueba x de A, entonces $\lambda x.t$ es una prueba de $A \to B$, es decir un término esperando una prueba de A para dar una prueba de B. Del mismo modo, la regla $(@)$ coincide con el *modus ponens*: si t es una prueba de $A \to B$ y u una prueba de A, entonces $t\ u$ es una prueba de B.

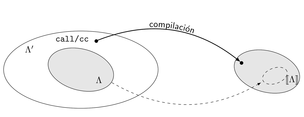
Matemáticas	**Informática**
Prueba	Programa
Fórmula	Tipo
Regla de deducción $\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$	Regla de tipaje $\dfrac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$
A implica B A y B $\exists x \in A.B(x)$ axioma	función $A \to B$ par de A y B par de $t : A$ y $p : B(t)$ primitiva

Limitaciones Descubierta independientemente por Curry en 1934 y Howard en 1969, esa correspondencia era limitada en su forma original. Del lado de la matemática, la correspondencia incluye solamente un fragmento llamado **intuicionista**, lo cual excluye principios de razonamiento **clásicos**, como el principio de tercero excluido que afirma que para toda formula A, $A \vee \neg A$ se cumple. De forma dual, los programas involucrados por la correspondencia son todos puramente **funcionales**. En particular, la correspondencia no toma en cuenta ningún efecto colateral, como el acceso a una memoria, el recurso a excepciones o la posibilidad de tener puntos de control para volver atrás en una computación (*backtracking*).

¿Cómo extender esa correspondencia?

Desde ya, tenemos dos puntos de vista para entender cómo extender la correspondencia de Curry-Howard: o bien nos preguntamos cómo se puede agregar un axioma nuevo a una teoría (sin afectar su consistencia), o bien tratamos de extender un lenguaje de programación con una nueva instrucción. En ambos casos, una solución consiste en definir una traducción desde el lenguaje extendido hacía una lenguaje más restringido, de tal forma que se mantengan las buenas propiedades del lenguaje de llegada (consistencia, normalización,...) a través de la traducción.

De forma muy interesante, definir una traducción lógica (es decir de una teoría hacía otra) puede corresponder exactamente al definición de una traducción de programas (es decir una compilación), y viceversa. Esquemáticamente, eso corresponde a la situación escrita en rojo.



Nuevo axioma	Primitiva de programación
$A \vee \neg A$	call/cc
⇕ Traducción lógica	⇕ Traducción de programas
Traducción negativa	Traducción CPS

Un primer caso: lógica clásica y *backtracking*

Aprovechando del hecho de que las dobles negaciones de los principios de razonamiento clásicos (*e.g.* $\neg\neg(A \vee \neg A)$) son válidas en lógica intuicionista, Gödel y Gentzen definieron en los años 30 una traducción lógica permitiendo encajar la lógica clásica en la lógica intuicionista. Independientemente, en los años 70 se estudió una clase de traducciones de programas (dicha en *continuation-passing style* (CPS)) permitiendo controlar explícitamente el futuro de una computación. Durante años, esas dos traducciones coexistieron hasta que Griffin se dio cuenta, en el año 90, de que la traducción de tipos correspondiendo a una traducción CPS era exactamente una traducción negativa de Gödel-Gentzen. Más aun, él observó que esas traducciones CPS permitían compilar un operador de control (la instrucción call/cc, que brinda *backtracking*), cuyo tipo coincidía con un principio de razonamiento clásico.

Teoría vs. Modelos

Ya que ciertas fórmulas no son ni *ciertas* ni *falsas* en la teoría ambiente (*e.g.* $A \vee \neg A$ en lógica intuicionista), ¿no se podrá decir más nada de ellas? En realidad, existe otro punto de vista: así cómo se puede estudiar la demostrabilidad en una teoría, se puede estudiar también la validez en una estructura particular donde sepamos interpretar todas las fórmulas. Semejante estructura, junto con un criterio de validez tal que todos los teoremas sean válidos, se llama un *modelo*. Por ejemplo, considerando el conjunto $\{\checkmark, \boldsymbol{\times}\}$, las tablas de verdad permiten definir una interpretación de las fórmulas proposicionales ($A \wedge B, A \vee B, A \Rightarrow B, \dots$):

$A \Rightarrow B$	A ✓	A ✗
B ✓	✓	✗
B ✗	✗	✓

$A \wedge B$	B ✓	B ✗
A ✓	✓	✗
A ✗	✗	✗

$A \vee B$	B ✓	B ✗
A ✓	✓	✓
A ✗	✓	✗

A	$\neg A$	$A \vee \neg A$
✓	✗	✓
✗	✓	✓

En ese marco, considerando una fórmula válida si su interpretación es ✓ en todos los casos, observamos que el principio de tercero excluido es válido. Sin embargo, existen otros modelos en los cuales no lo es: eso demuestra la independencia del principio de tercero excluido respecto a la lógica intuicionista, es decir que ni él ni su negación son demostrables en lógica intuicionista. Por lo tanto, los modelos nos ofrecen herramientas poderosas para estudiar fórmulas de la cuales no se sabe si son axiomas o no, o si son demostrables (o negables).

Realizabilidad clásica

Impulsada por Krivine desde los años 2000, la **realizabilidad clásica** propone la definición de nuevos modelos de diferentes teorías clásicas. En esos modelos, cada formula A es interpretada por un conjunto de términos cuyo comportamiento computacional es dirigido por A. Esos términos, notados $t \Vdash A$, se llaman los **realizadores** de la fórmula A:

$$A \mapsto \{t : t \Vdash A\} \qquad \textit{(Modelo de realizabilidad)}$$

Esas interpretaciones se mostraron particularmente valiosas para estudiar el contenido computacional de distintas pruebas clásicas. Por ejemplo, las usamos nuevamente para contestar a la pregunta a la cual se dedica la mitad de esta tesis:

¿Cómo definir un término de prueba en lógica clásica para el axioma de elección dependiente usando efectos colaterales?

Además, respecto a los modelos que define, la realizabilidad clásica trajo resultados nuevos y muy sorprendentes. No obstante, esos modelos son muy complejos y entender sus estructuras define una nueva línea de investigación en sí. La otra mitad de esta tesis fue así consagrada a la pregunta siguiente:

¿Cómo analizar la estructura de los modelos inducidos por la realizabilidad clásica desde un punto de vista algebraico?