

Un nuevo conectivo lógico-computacional para describir la superposición cuántica

Alejandro Díaz-Caro

UNIVERSIDAD NACIONAL DE QUILMES

&

INSTITUTO DE CIENCIAS DE LA COMPUTACIÓN (CONICET / UBA)

Comunicaciones seguras: Métodos cuánticos y poscuánticos
ARSAT

22 y 23 de junio de 2022

Plan

- ▶ ¿Porqué estudiar lenguajes de programación?
- ▶ ¿Y eso qué tiene que ver con lógica?
- ▶ ¿Y eso porqué no es suficiente con lógica clásica?
- ▶ ⊙ el conector lógico de la superposición cuántica
- ▶ ⊙ en el corazón de los lenguajes de programación cuánticos

¿Porqué estudiar lenguajes de programación?

Algoritmo (cuántico o clásico) \longleftrightarrow escrito en algún lenguaje

¿Porqué estudiar lenguajes de programación?

Algoritmo (cuántico o clásico) \longleftrightarrow escrito en algún lenguaje

Lenguaje	Ventajas	Desventajas
Circuitos	<ul style="list-style-type: none">● Lenguaje gráfico● Fácil de visualizar● Muy aceptado	<ul style="list-style-type: none">● No todo algoritmo cuántico se puede expresar en circuitos● De muy bajo nivel (cercano al hardware)

¿Porqué estudiar lenguajes de programación?

Algoritmo (cuántico o clásico) \longleftrightarrow escrito en algún lenguaje

Lenguaje	Ventajas	Desventajas
Circuitos	<ul style="list-style-type: none">● Lenguaje gráfico● Fácil de visualizar● Muy aceptado	<ul style="list-style-type: none">● No todo algoritmo cuántico se puede expresar en circuitos● De muy bajo nivel (cercano al hardware)
Pseudocódigo (instrucciones en español)	<ul style="list-style-type: none">● Fácil de entender	<ul style="list-style-type: none">● No es muy formal● Puede ser ambiguo

¿Porqué estudiar lenguajes de programación?

Algoritmo (cuántico o clásico) \longleftrightarrow escrito en algún lenguaje

Lenguaje	Ventajas	Desventajas
Circuitos	<ul style="list-style-type: none">● Lenguaje gráfico● Fácil de visualizar● Muy aceptado	<ul style="list-style-type: none">● No todo algoritmo cuántico se puede expresar en circuitos● De muy bajo nivel (cercano al hardware)
Pseudocódigo (instrucciones en español)	<ul style="list-style-type: none">● Fácil de entender	<ul style="list-style-type: none">● No es muy formal● Puede ser ambiguo
Lenguajes de alto nivel	<ul style="list-style-type: none">● Verificable● Nos alejamos del hardware● Puede dar ideas para nuevos algoritmos	<ul style="list-style-type: none">● La mayoría de los algoritmos cuánticos no fueron escritos en esos lenguajes

¿Porqué estudiar lenguajes de programación?

Algoritmo (cuántico o clásico) \longleftrightarrow escrito en algún lenguaje

Lenguaje	Ventajas	Desventajas
Circuitos	<ul style="list-style-type: none">● Lenguaje gráfico● Fácil de visualizar● Muy aceptado	<ul style="list-style-type: none">● No todo algoritmo cuántico se puede expresar en circuitos● De muy bajo nivel (cercano al hardware)
Pseudocódigo (instrucciones en español)	<ul style="list-style-type: none">● Fácil de entender	<ul style="list-style-type: none">● No es muy formal● Puede ser ambiguo
Lenguajes de alto nivel	<ul style="list-style-type: none">● Verificable● Nos alejamos del hardware● Puede dar ideas para nuevos algoritmos	<ul style="list-style-type: none">● La mayoría de los algoritmos cuánticos no fueron escritos en esos lenguajes
Cálculo lambda	<ul style="list-style-type: none">● Formalismo matemático● En el “corazón” de los lenguajes de programación funcionales	<ul style="list-style-type: none">● No es práctico para programar (sólo para demostrar propiedades fundamentales)

¿Y eso qué tiene que ver con lógica?

La correspondencia de Curry-Howard-Lambek

Claculo lambda tipado	Teoría de la demostración	Teoría de categorías
Tipos	Proposiciones lógicas	Objetos de una categoría
Programas	Pruebas de proposiciones lógicas	Morfismos de una categoría

¿Y eso qué tiene que ver con lógica?

La correspondencia de Curry-Howard-Labmek

Claculo lambda tipado	Teoría de la demostración	Teoría de categorías
Tipos	Proposiciones lógicas	Objetos de una categoría
Programas	Pruebas de proposiciones lógicas	Morfismos de una categoría

El programa que toma un argumento y devuelve el mismo tiene tipo $A \Rightarrow A$ para algún tipo A .

```
func(a:A){ return a;}
```

La proposición lógica $A \Rightarrow A$ se prueba asumiendo A para poder derivar A .

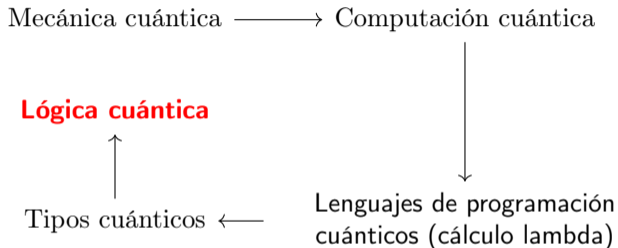
$$\frac{A \vdash A}{\vdash A \Rightarrow A}$$

El morfismo identidad va de cualquier objeto A en sí mismo.

$$A \xrightarrow{id} A$$

¿Y eso qué tiene que ver con lógica?

Cálculo lambda tipado \Leftrightarrow Lógica matemática



El objetivo es definir una **lógica cuántica usando métodos de **ciencias de la computación****

- ▶ Para poder verificar correctitud de programas
- ▶ Para poder verificar correctitud de lenguajes de programación
- ▶ Para entender mejor la física

¿Y eso qué tiene que ver con lógica?

El lambda cálculo no tipado

Introducido en 1936 por Alonzo Church

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

¿Y eso qué tiene que ver con lógica?

El lambda cálculo no tipado

Introducido en 1936 por Alonzo Church

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

Dos simplificaciones básicas sobre el concepto de función

▶ **Anonimidad:**

Ejemplo:

se escribe anónimamente como

Los nombres de funciones no son necesarios

$$\begin{aligned} sqsum(x, y) &= x^2 + y^2 \\ (x, y) &\mapsto x^2 + y^2 \end{aligned}$$

¿Y eso qué tiene que ver con lógica?

El lambda cálculo no tipado

Introducido en 1936 por Alonzo Church

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

Dos simplificaciones básicas sobre el concepto de función

▶ **Anonimidad:**

Ejemplo:

se escribe anónimamente como

$$\begin{aligned} sqsum(x, y) &= x^2 + y^2 \\ (x, y) &\mapsto x^2 + y^2 \end{aligned}$$

Los nombres de funciones no son necesarios

▶ **Todas las funciones son a una sola variable:**

Ejemplo:

Se escribe como

$$\begin{aligned} (x, y) &\mapsto x^2 + y^2 \\ x &\mapsto (y \mapsto x^2 + y^2) \end{aligned}$$

Una función a dos variables es una función a una variable que devuelve una función a una variable, la cual hace el cálculo

¿Y eso qué tiene que ver con lógica?

El lambda cálculo no tipado

Lenguaje de términos (una gramática)

- ▶ Una variable $x \in Vars$ es un término
- ▶ Si t es un término y x una variable, $\lambda x.t$ es un término
- ▶ Si t y r son dos términos, tr es un término

Esos son todos los términos posibles

$(x \mapsto t)$
(application)

$$t ::= x \mid \lambda x.t \mid tt$$

¿Y eso qué tiene que ver con lógica?

El lambda cálculo no tipado

Lenguaje de términos (una gramática)

- ▶ Una variable $x \in Vars$ es un término
- ▶ Si t es un término y x una variable, $\lambda x.t$ es un término
- ▶ Si t y r son dos términos, tr es un término

$(x \mapsto t)$
(application)

Esos son todos los términos posibles

$$t ::= x \mid \lambda x.t \mid tt$$

Una regla de reescritura

$$(\lambda x.t)r \longrightarrow [x := r]t$$

¿Y eso qué tiene que ver con lógica?

El lambda cálculo no tipado

Lenguaje de términos (una gramática)

- ▶ Una variable $x \in Vars$ es un término
- ▶ Si t es un término y x una variable, $\lambda x.t$ es un término
- ▶ Si t y r son dos términos, tr es un término

$(x \mapsto t)$
(application)

Esos son todos los términos posibles

$$t ::= x \mid \lambda x.t \mid tt$$

Una regla de reescritura

$$(\lambda x.t)r \longrightarrow [x := r]t$$

Ejemplo

$$f(g, x) = g(x) \quad \text{se escribe} \quad \lambda g.\lambda x.gx$$

¿Y eso qué tiene que ver con lógica?

El lambda cálculo no tipado

Lenguaje de términos (una gramática)

- ▶ Una variable $x \in Vars$ es un término
- ▶ Si t es un término y x una variable, $\lambda x.t$ es un término
- ▶ Si t y r son dos términos, tr es un término

$(x \mapsto t)$
(application)

Esos son todos los términos posibles

$$t ::= x \mid \lambda x.t \mid tt$$

Una regla de reescritura

$$(\lambda x.t)r \longrightarrow [x := r]t$$

Ejemplo

$f(g, x) = g(x)$ se escribe $\lambda g.\lambda x.gx$

$f(\mathbf{h}, \mathbf{y})$ se escribe $(\lambda g.\lambda x.gx)\mathbf{h}\mathbf{y}$

$$\underbrace{(\lambda g.\lambda x.gx)}_{(\lambda x.t)} \underbrace{\mathbf{h}}_r \mathbf{y} \longrightarrow \underbrace{(\lambda x.\mathbf{h}x)}_{[x:=r]t} \mathbf{y} \longrightarrow \mathbf{h}\mathbf{y}$$

¿Y eso qué tiene que ver con lógica?

Codificación de Church

$$\begin{array}{ll} \text{True} = \lambda x. \lambda y. x & (f(x, y) = x) \\ \text{False} = \lambda x. \lambda y. y & (f(x, y) = y) \end{array}$$

¿Y eso qué tiene que ver con lógica?

Codificación de Church

True = $\lambda x.\lambda y.x$

$(f(x, y) = x)$

False = $\lambda x.\lambda y.y$

$(f(x, y) = y)$

If c Then t Else $e = c t e$

¿Y eso qué tiene que ver con lógica?

Codificación de Church

True = $\lambda x.\lambda y.x$

$(f(x, y) = x)$

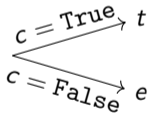
False = $\lambda x.\lambda y.y$

$(f(x, y) = y)$

If c Then t Else $e = c t e$

Ejemplos

If c Then t Else $e = c t e$



¿Y eso qué tiene que ver con lógica?

Codificación de Church

True = $\lambda x.\lambda y.x$

$(f(x, y) = x)$

False = $\lambda x.\lambda y.y$

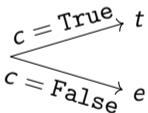
$(f(x, y) = y)$

If c Then t Else $e = c t e$

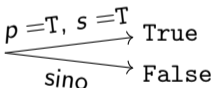
p and $s = p s p$

Ejemplos

If c Then t Else $e = c t e$



p and $s = p s p$



¿Y eso qué tiene que ver con lógica?

Codificación de Church

$$\text{True} = \lambda x. \lambda y. x$$

$$(f(x, y) = x)$$

$$\text{False} = \lambda x. \lambda y. y$$

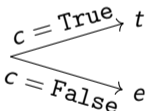
$$(f(x, y) = y)$$

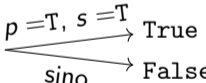
$$\text{If } c \text{ Then } t \text{ Else } e = c t e$$

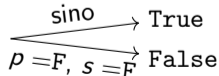
$$p \text{ and } s = p s p$$

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

Ejemplos

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


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


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


¿Y eso qué tiene que ver con lógica?

Lambda cálculo tipado

Una manera de clasificar términos *estáticamente* (es decir, sin “ejecutar el programa”)

¿Y eso qué tiene que ver con lógica?

Lambda cálculo tipado

Una manera de clasificar términos *estáticamente* (es decir, sin “ejecutar el programa”)

Términos $t ::= x \mid \lambda x^A.t \mid tt$

Tipos $A ::= \tau \mid A \Rightarrow A$

► τ es un *tipo básico* (como Int o Bool)

► $A \Rightarrow A$ es el tipo de las funciones

¿Y eso qué tiene que ver con lógica?

Lambda cálculo tipado

Una manera de clasificar términos *estáticamente* (es decir, sin “ejecutar el programa”)

Términos $t ::= x \mid \lambda x^A.t \mid tt$

Tipos $A ::= \tau \mid A \Rightarrow A$

► τ es un *tipo básico* (como Int o Bool)

► $A \Rightarrow A$ es el tipo de las funciones

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

$\Gamma \vdash t : A$ “ t tiene tipo A en el contexto Γ ”

¿Y eso qué tiene que ver con lógica?

Lambda cálculo tipado

Una manera de clasificar términos *estáticamente* (es decir, sin “ejecutar el programa”)

Términos $t ::= x \mid \lambda x^A.t \mid tt$

Tipos $A ::= \tau \mid A \Rightarrow A$

► τ es un *tipo básico* (como Int o Bool)

► $A \Rightarrow A$ es el tipo de las funciones

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

$\Gamma \vdash t : A$ “ t tiene tipo A en el contexto Γ ”

Reglas de tipado

¿Y eso qué tiene que ver con lógica?

Lambda cálculo tipado

Una manera de clasificar términos *estáticamente* (es decir, sin “ejecutar el programa”)

Términos	$t ::= x \mid \lambda x^A.t \mid tt$
Tipos	$A ::= \tau \mid A \Rightarrow A$

► τ es un *tipo básico* (como Int o Bool)

► $A \Rightarrow A$ es el tipo de las funciones

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

$\Gamma \vdash t : A$ “ t tiene tipo A en el contexto Γ ”

Reglas de tipado

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

¿Y eso qué tiene que ver con lógica?

Lambda cálculo tipado

Una manera de clasificar términos *estáticamente* (es decir, sin “ejecutar el programa”)

Términos	$t ::= x \mid \lambda x^A.t \mid tt$
Tipos	$A ::= \tau \mid A \Rightarrow A$

► τ es un *tipo básico* (como Int o Bool)

► $A \Rightarrow A$ es el tipo de las funciones

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

$\Gamma \vdash t : A$ “ t tiene tipo A en el contexto Γ ”

Reglas de tipado

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

¿Y eso qué tiene que ver con lógica?

Lambda cálculo tipado

Una manera de clasificar términos *estáticamente* (es decir, sin “ejecutar el programa”)

Términos	$t ::= x \mid \lambda x^A.t \mid tt$
Tipos	$A ::= \tau \mid A \Rightarrow A$

► τ es un *tipo básico* (como Int o Bool)

► $A \Rightarrow A$ es el tipo de las funciones

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

$\Gamma \vdash t : A$ “ t tiene tipo A en el contexto Γ ”

Reglas de tipado

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

¿Y eso qué tiene que ver con lógica?

Lambda cálculo tipado

Una manera de clasificar términos *estáticamente* (es decir, sin “ejecutar el programa”)

Términos	$t ::= x \mid \lambda x^A.t \mid tt$
Tipos	$A ::= \tau \mid A \Rightarrow A$

- ▶ τ es un *tipo básico* (como Int o Bool)
- ▶ $A \Rightarrow A$ es el tipo de las funciones

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

$\Gamma \vdash t : A$ “ t tiene tipo A en el contexto Γ ”

Reglas de tipado

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

Ejemplo

$$\frac{\frac{\frac{}{x : \tau \Rightarrow \tau \vdash x : \tau \Rightarrow \tau} \text{ax}}{\vdash \lambda x^{\tau \Rightarrow \tau}.x : (\tau \Rightarrow \tau) \Rightarrow (\tau \Rightarrow \tau)} \Rightarrow_I}{\vdash (\lambda x^{\tau \Rightarrow \tau}.x)(\lambda x^{\tau}.x) : \tau \Rightarrow \tau} \Rightarrow_E}{\vdash \lambda x^{\tau}.x : \tau \Rightarrow \tau} \Rightarrow_I$$

Verificación: $(\lambda x^{\tau \Rightarrow \tau}.x)(\lambda x^{\tau}.x)$ reescribe a $\lambda x^{\tau}.x$ (de tipo $\tau \Rightarrow \tau$)

¿Y eso qué tiene que ver con lógica?

La correspondencia de Curry-Howard

Lógica intuicionista mínima (sómo con implicación)

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

¿Y eso qué tiene que ver con lógica?

La correspondencia de Curry-Howard

Lógica intuicionista mínima (sómo con implicación)

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

¿Se parece a algo?

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

Los términos son la prueba de los predicados

Las pruebas... ¡son programas!

Haskell Curry & William Howard,
entre 1934 y 1969

Lógicas más complejas corresponden a tipos más complejos.

Por ejemplo, la conjunción corresponde a los pares. La disjunción corresponde al “match”, etc.

¿Y eso qué tiene que ver con lógica?

Las pruebas son programas. . . que nos dicen cómo construir la prueba

Sea $\Gamma := A \Rightarrow B, B \Rightarrow C, A$

$$\frac{\frac{\frac{\frac{\frac{\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}}{\Gamma \vdash B \Rightarrow C}}{\Gamma \vdash C}}{A \Rightarrow B, B \Rightarrow C \vdash A \Rightarrow C}}{A \Rightarrow B \vdash (B \Rightarrow C) \Rightarrow A \Rightarrow C}}{\vdash (A \Rightarrow B) \Rightarrow (B \Rightarrow C) \Rightarrow (A \Rightarrow C)}}$$

Sea $\Gamma := x : A \Rightarrow B, y : B \Rightarrow C, z : A$

$$\frac{\frac{\frac{\frac{\frac{\frac{\Gamma \vdash x : A \Rightarrow B \quad \Gamma \vdash z : A}{\Gamma \vdash xz : B}}{\Gamma \vdash y : B \Rightarrow C}}{\Gamma \vdash y(xz)C}}{x : A \Rightarrow B, y : B \Rightarrow C \vdash \lambda z^A. y(xz) : A \Rightarrow C}}{x : A \Rightarrow B \vdash \lambda y^{B \Rightarrow C}. \lambda z^A. y(xz) : (B \Rightarrow C) \Rightarrow A \Rightarrow C}}{\vdash \lambda x^{A \Rightarrow B}. \lambda y^{B \Rightarrow C}. \lambda z^A. y(xz) : (A \Rightarrow B) \Rightarrow (B \Rightarrow C) \Rightarrow (A \Rightarrow C)}}$$

¿Y eso qué tiene que ver con lógica?

Las pruebas son programas. . . que se pueden ejecutar para simplificar las pruebas

$$\frac{\frac{\overline{A \Rightarrow A \vdash A \Rightarrow A}}{\vdash (A \Rightarrow A) \Rightarrow A \Rightarrow A} \quad \frac{\overline{A \vdash A}}{\vdash A \Rightarrow A}}{\vdash A \Rightarrow A}$$

$$\frac{\frac{\overline{x : A \Rightarrow A \vdash x : A \Rightarrow A}}{\vdash \lambda x^{A \Rightarrow A}. x : (A \Rightarrow A) \Rightarrow A \Rightarrow A} \quad \frac{\overline{y : A \vdash y : A}}{\vdash \lambda y^A. y : A \Rightarrow A}}{\vdash (\lambda x^{A \Rightarrow A}. x) \lambda y^A. y : A \Rightarrow A}$$

¿Y eso qué tiene que ver con lógica?

Las pruebas son programas. . . que se pueden ejecutar para simplificar las pruebas

$$\frac{\frac{\overline{A \Rightarrow A \vdash A \Rightarrow A}}{\vdash (A \Rightarrow A) \Rightarrow A \Rightarrow A} \quad \frac{\overline{A \vdash A}}{\vdash A \Rightarrow A}}{\vdash A \Rightarrow A}$$

$$\frac{\frac{\overline{x : A \Rightarrow A \vdash x : A \Rightarrow A}}{\vdash \lambda x^{A \Rightarrow A}. x : (A \Rightarrow A) \Rightarrow A \Rightarrow A} \quad \frac{\overline{y : A \vdash y : A}}{\vdash \lambda y^A. y : A \Rightarrow A}}{\vdash (\lambda x^{A \Rightarrow A}. x) \lambda y^A. y : A \Rightarrow A} \longrightarrow \frac{\overline{y : A \vdash y : A}}{\vdash \lambda y^A. y : A \Rightarrow A}$$

¿Y eso qué tiene que ver con lógica?

Agregando a Lambek con un ejemplo

Categoría **Set**: Los objetos son conjuntos, los morfismos son funciones.

$$[[\tau]] = \tau$$

$$[[A \Rightarrow B]] = [[A], [[B]]]$$

¿Y eso qué tiene que ver con lógica?

Agregando a Lambek con un ejemplo

Categoría **Set**: Los objetos son conjuntos, los morfismos son funciones.

$$\llbracket \tau \rrbracket = \tau \qquad \llbracket A \Rightarrow B \rrbracket = \llbracket [A], [B] \rrbracket$$

$$\llbracket \overline{\Gamma, x : A \vdash x : A} \quad \text{ax} \rrbracket = \Gamma \times A \xrightarrow{\pi_A} A \xrightarrow{\text{Id}} A$$

¿Y eso qué tiene que ver con lógica?

Agregando a Lambek con un ejemplo

Categoría **Set**: Los objetos son conjuntos, los morfismos son funciones.

$$[[\tau]] = \tau \qquad [[A \Rightarrow B]] = [[A], [B]]$$

$$\begin{aligned} \left[\overline{\Gamma, x:A \vdash x:A} \right]^{ax} &= \Gamma \times A \xrightarrow{\pi_A} A \xrightarrow{\text{Id}} A \\ \left[\frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x^A. t : A \Rightarrow B} \Rightarrow_I \right] &= \Gamma \xrightarrow{\eta^A} [A, \Gamma \times A] \xrightarrow{[\text{Id}, t]} [A, B] \end{aligned}$$

¿Y eso qué tiene que ver con lógica?

Agregando a Lambek con un ejemplo

Categoría **Set**: Los objetos son conjuntos, los morfismos son funciones.

$$[[\tau]] = \tau$$

$$[[A \Rightarrow B]] = [[A], [[B]]]$$

$$\begin{aligned} & \left[\overline{\Gamma, x:A \vdash x:A} \right]^{ax} = \Gamma \times A \xrightarrow{\pi_A} A \xrightarrow{\text{Id}} A \\ & \left[\frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x^A. t : A \Rightarrow B} \Rightarrow_I \right] = \Gamma \xrightarrow{\eta^A} [A, \Gamma \times A] \xrightarrow{[\text{Id}, t]} [A, B] \\ & \left[\frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash r : A}{\Gamma \vdash tr : B} \Rightarrow_E \right] = \Gamma \xrightarrow{\delta} \Gamma \times \Gamma \xrightarrow{t \times r} [A, B] \times A \xrightarrow{\varepsilon} B \end{aligned}$$

¿Y eso qué tiene que ver con lógica?

Agregando a Lambek con un ejemplo

Categoría **Set**: Los objetos son conjuntos, los morfismos son funciones.

$$[[\tau]] = \tau$$

$$[[A \Rightarrow B]] = [[A], [[B]]]$$

$$\left[\overline{\Gamma, x : A \vdash x : A} \right]^{ax} = \Gamma \times A \xrightarrow{\pi_A} A \xrightarrow{\text{Id}} A$$

$$\left[\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x^A. t : A \Rightarrow B} \Rightarrow_I \right] = \Gamma \xrightarrow{\eta^A} [A, \Gamma \times A] \xrightarrow{[\text{Id}, t]} [A, B]$$

$$\left[\frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash r : A}{\Gamma \vdash tr : B} \Rightarrow_E \right] = \Gamma \xrightarrow{\delta} \Gamma \times \Gamma \xrightarrow{t \times r} [A, B] \times A \xrightarrow{\varepsilon} B$$

Teorema 1 $\left. \begin{array}{l} \Gamma \vdash t : A \\ t \rightarrow r \end{array} \right\} \implies [[\Gamma \vdash t : A]] = [[\Gamma \vdash r : A]]$

¿Y eso qué tiene que ver con lógica?

Agregando a Lambek con un ejemplo

Categoría **Set**: Los objetos son conjuntos, los morfismos son funciones.

$$[[\tau]] = \tau \qquad [[A \Rightarrow B]] = [[A], [B]]$$

$$\begin{aligned} \left[\overline{\Gamma, x:A \vdash x:A} \right]^{ax} &= \Gamma \times A \xrightarrow{\pi_A} A \xrightarrow{\text{Id}} A \\ \left[\frac{\Gamma, x:A \vdash t:B}{\Gamma \vdash \lambda x^A. t: A \Rightarrow B} \Rightarrow_I \right] &= \Gamma \xrightarrow{\eta^A} [A, \Gamma \times A] \xrightarrow{[\text{Id}, t]} [A, B] \\ \left[\frac{\Gamma \vdash t: A \Rightarrow B \quad \Gamma \vdash r: A}{\Gamma \vdash tr: B} \Rightarrow_E \right] &= \Gamma \xrightarrow{\delta} \Gamma \times \Gamma \xrightarrow{t \times r} [A, B] \times A \xrightarrow{\varepsilon} B \end{aligned}$$

Teorema 1 $\left. \begin{array}{l} \Gamma \vdash t: A \\ t \rightarrow r \end{array} \right\} \implies [[\Gamma \vdash t: A]] = [[\Gamma \vdash r: A]]$

Teorema 2: $[[\Gamma \vdash t: A]] = [[\Gamma \vdash r: A]] \implies t \sim r.$

¿Y eso porqué no es suficiente con lógica clásica?

La superposición

Principio de superposición: el estado de un sistema cuántico puede ser la superposición de diferentes estados contradictorios.

¿Y eso porqué no es suficiente con lógica clásica?

La superposición

Principio de superposición: el estado de un sistema cuántico puede ser la superposición de diferentes estados contradictorios.

Intento 1 de modelar la superposición de A y B

$$A \wedge B$$

La propiedad A y la B se cumplen a la vez

Pero de $A \wedge B$ puedo “extraer” A o B a piacere. No hay ninguna indeterminación.

¿Y eso porqué no es suficiente con lógica clásica?

La superposición

Principio de superposición: el estado de un sistema cuántico puede ser la superposición de diferentes estados contradictorios.

Intento 1 de modelar la superposición de A y B

$$A \wedge B$$

La propiedad A y la B se cumplen a la vez

Pero de $A \wedge B$ puedo “extraer” A o B a piacere. No hay ninguna indeterminación.

Intento 2 de modelar la superposición de A y B

$$A \vee B$$

La propiedad A o la propiedad B se cumple, y yo no puedo saber a priori cual.

Pero $A \vee B$ modela mi ignorancia sobre el sistema, no la superposición.

⊙ el conectivo lógico de la superposición cuántica

[Díaz-Caro & Dowek ICTAC 2021]

Conjunción

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_i$$

$$\frac{\Gamma \vdash A \wedge B \quad \Gamma, A \vdash C}{\Gamma \vdash C} \wedge_e^1$$

$$\frac{\Gamma \vdash A \wedge B \quad \Gamma, B \vdash C}{\Gamma \vdash C} \wedge_e^2$$

⊙ el conectivo lógico de la superposición cuántica

[Díaz-Caro & Dowek ICTAC 2021]

Conjunción

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_i$$

$$\frac{\Gamma \vdash A \wedge B \quad \Gamma, A \vdash C}{\Gamma \vdash C} \wedge_e^1$$

$$\frac{\Gamma \vdash A \wedge B \quad \Gamma, B \vdash C}{\Gamma \vdash C} \wedge_e^2$$

Disjunción

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee_i^1$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee_i^2$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee_e$$

⊙ el conectivo lógico de la superposición cuántica

[Díaz-Caro & Dowek ICTAC 2021]

Conjunción

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_i$$

$$\frac{\Gamma \vdash A \wedge B \quad \Gamma, A \vdash C}{\Gamma \vdash C} \wedge_e^1$$

$$\frac{\Gamma \vdash A \wedge B \quad \Gamma, B \vdash C}{\Gamma \vdash C} \wedge_e^2$$

Disjunción

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee_i^1$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee_i^2$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee_e$$

Superposición

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \odot B} \odot_i$$

$$\frac{\Gamma \vdash A \odot B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \odot_e$$

⊙ el conectivo lógico de la superposición cuántica

[Díaz-Caro & Dowek ICTAC 2021]

Conjunción

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \wedge B} \wedge_i$$
$$\frac{\Gamma \vdash t : A \wedge B \quad \Gamma, x : A \vdash r : C}{\Gamma \vdash \delta_{\wedge}^1(t, x.r) : C} \wedge_e^1$$
$$\frac{\Gamma \vdash t : A \wedge B \quad \Gamma, y : B \vdash s : C}{\Gamma \vdash \delta_{\wedge}^2(t, y.s) : C} \wedge_e^2$$
$$\delta_{\wedge}^1(\langle a, b \rangle, x.r) \longrightarrow (x := a)r$$

⊙ el conectivo lógico de la superposición cuántica

[Díaz-Caro & Dowek ICTAC 2021]

Conjunción

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \wedge B} \wedge_i \quad \frac{\Gamma \vdash t : A \wedge B \quad \Gamma, x : A \vdash r : C}{\Gamma \vdash \delta_{\wedge}^1(t, x.r) : C} \wedge_e^1 \quad \frac{\Gamma \vdash t : A \wedge B \quad \Gamma, y : B \vdash s : C}{\Gamma \vdash \delta_{\wedge}^2(t, y.s) : C} \wedge_e^2$$
$$\delta_{\wedge}^1(\langle a, b \rangle, x.r) \longrightarrow (x := a)r$$

Disjunción

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \text{inl}(a) : A \vee B} \vee_i^1 \quad \frac{\Gamma \vdash b : B}{\Gamma \vdash \text{inr}(b) : A \vee B} \vee_i^2 \quad \frac{\Gamma \vdash t : A \vee B \quad \Gamma, x : A \vdash r : C \quad \Gamma, y : B \vdash s : C}{\Gamma \vdash \delta_{\vee}(t, x.r, y.s) : C} \vee_e$$
$$\delta_{\vee}(\text{inl}(a), x.r, y.s) \longrightarrow (x := a)r$$

⊙ el conectivo lógico de la superposición cuántica

[Díaz-Caro & Dowek ICTAC 2021]

Conjunción

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash \langle a, b \rangle : A \wedge B} \wedge_i \quad \frac{\Gamma \vdash t : A \wedge B \quad \Gamma, x : A \vdash r : C}{\Gamma \vdash \delta_{\wedge}^1(t, x.r) : C} \wedge_e^1 \quad \frac{\Gamma \vdash t : A \wedge B \quad \Gamma, y : B \vdash s : C}{\Gamma \vdash \delta_{\wedge}^2(t, y.s) : C} \wedge_e^2$$
$$\delta_{\wedge}^1(\langle a, b \rangle, x.r) \longrightarrow (x := a)r$$

Disjunción

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \text{inl}(a) : A \vee B} \vee_i^1 \quad \frac{\Gamma \vdash b : B}{\Gamma \vdash \text{inr}(b) : A \vee B} \vee_i^2 \quad \frac{\Gamma \vdash t : A \vee B \quad \Gamma, x : A \vdash r : C \quad \Gamma, y : B \vdash s : C}{\Gamma \vdash \delta_{\vee}(t, x.r, y.s) : C} \vee_e$$
$$\delta_{\vee}(\text{inl}(a), x.r, y.s) \longrightarrow (x := a)r$$

Superposición

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B}{\Gamma \vdash [a, b] : A \odot B} \odot_i \quad \frac{\Gamma \vdash t : A \odot B \quad \Gamma, x : A \vdash r : C \quad \Gamma, y : B \vdash s : C}{\Gamma \vdash \delta_{\odot}(t, x.r, y.s) : C} \odot_e$$
$$\delta_{\odot}([a, b], x.r, y.s) \longrightarrow (x := a)r \quad \delta_{\odot}([a, b], x.r, y.s) \longrightarrow (y := b)s$$

⊙ en el corazón de los lenguajes de programación cuánticos

Tipo básico \top con término \star

$$\overline{\Gamma \vdash \star : \top} \quad \top_i$$

Bit: $\top \vee \top$ $0 = \text{inl}(\star)$ $1 = \text{inr}(\star)$

If b Then r Else s $\delta_{\vee}(b, x.r, y.s)$

⊙ en el corazón de los lenguajes de programación cuánticos

Tipo básico \top con término \star

$$\overline{\Gamma \vdash \star : \top} \quad \top_i$$

Bit: $\top \vee \top$ $0 = \text{inl}(\star)$ $1 = \text{inr}(\star)$

If b Then r Else s $\delta_{\vee}(b, x.r, y.s)$

Tipo básico \top con términos $\alpha.\star$, con $\alpha \in \mathcal{S}$

$$\overline{\Gamma \vdash \alpha.\star : \top} \quad \top_i(\alpha)$$

Qubit: $\top \odot \top$ $\left(\begin{smallmatrix} \alpha \\ \beta \end{smallmatrix}\right) = [\alpha.\star, \beta.\star]$

Medición de un qubit (asociando las probabilidades correspondientes)

$\delta_{\odot}(q, x.r, y.s)$

⊙ en el corazón de los lenguajes de programación cuánticos

Tipo básico \top con término \star

$$\overline{\Gamma \vdash \star : \top} \top_i$$

Bit: $\top \vee \top$ $0 = \text{inl}(\star)$ $1 = \text{inr}(\star)$

If b Then r Else s $\delta_{\vee}(b, x.r, y.s)$

Si agregamos las eliminaciones de la conjunción...

$$\frac{\Gamma \vdash t : A \odot B \quad \Gamma, x : A \vdash r : C}{\Gamma \vdash \delta_{\odot}^1(t, x.r) : C} \odot_e^1$$

$$\frac{\Gamma \vdash t : A \odot B \quad \Gamma, y : B \vdash s : C}{\Gamma \vdash \delta_{\odot}^2(t, y.s) : C} \odot_e^2$$

podemos codificar matrices:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \lambda x. (\delta_{\odot}^1(x, y. \delta_{\top}(y, [a.\star, b.\star])) + \delta_{\odot}^2(x, z. \delta_{\top}(z, [c.\star, d.\star])))$$

(ok, queda para otra charla que les cuente sobre ese $+$... o pueden bajar el paper)

Tipo básico \top con términos $\alpha.\star$, con $\alpha \in \mathcal{S}$

$$\overline{\Gamma \vdash \alpha.\star : \top} \top_i(\alpha)$$

Qubit: $\top \odot \top$ $\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = [\alpha.\star, \beta.\star]$

Medición de un qubit (asociando las probabilidades correspondientes)

$$\delta_{\odot}(q, x.r, y.s)$$

Resumiendo

- ▶ Introdujimos un nuevo operador lógico, \odot , que modela la superposición. Tiene la introducción de la conjunción y la eliminación de la disjunción
- ▶ Diseñamos un lenguaje de programación cuántico, con el operador lógico \odot

Resumiendo

- ▶ Introdujimos un nuevo operador lógico, \odot , que modela la superposición. Tiene la introducción de la conjunción y la eliminación de la disjunción
- ▶ Diseñamos un lenguaje de programación cuántico, con el operador lógico \odot

Lo que no conté

- ▶ Mostramos la relación de un fragmento de este cálculo con la lógica lineal intuicionista ([Díaz-Caro & Dowek, FSCD 2022])
- ▶ Dimos una interpretación categórica para ese fragmento ([Díaz-Caro & Malherbe, arXiv:2205.02142, 2022])
- ▶ Seguimos estudiando esta lógica, y la relación con otras lógicas conocidas.