Introducción a la computación cuántica y fundamentos de lenguajes de programación

Materia optativa de la
Licenciatura en Ciencias de la Computación
con créditos para el doctorado

Universidad Nacional de Rosario

Alejandro Díaz-Caro
Instituto de Investigación en Ciencias de la Computación
(CONICET / Universidad de Buenos Aires)
& Universidad Nacional de Quilmes

version 2018-06-25

Enfoque de este apunte

Estas notas están pensadas para estudiantes de grado y posgrado de computación, no de física. Es por ello que el enfoque que se da es casi puramente matemático, con algún comentario aquí y allá de la física que motiva el formalismo, pero todos los razonamientos se realizan exclusivamente desde el lado de la matemática.

Materia:
Introducción a la computación cuántica y fundamentos de lenguajes de programación

Departamento de Ciencias de la Computación
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario

Índice general

Capítulo 1

Introducción a la computación cuántica

> *I feel that a deep understanding of why quantum algorithms work is still lacking. Surely the power of quantum computers has something to do with entanglement, quantum parallelism, and the vastness of Hilbert space, but I think that it should be possible to pinpoint more precisely the true essence of the matter.*
>
> John Preskill [1998]

1.1. Introducción

La computación cuántica, una rama de las ciencias de la computación teórica, tiene su origen en la física, y más precisamente en el físico estadounidense Richard Feynman, quien en 1981 dedicó una charla en el Massachusetts Institute of Technology (MIT) al problema de la simulación de la física cuántica con computadoras clásicas. Sus ya célebres palabras finales resumen su frustración de ese entonces:

> *And I'm not happy with all the analyses that go with just the classical theory, because nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy. Thank you.*
>
> (ver, por ejemplo, [Brown, 2001, pp.100])

Esta provocación, lejos de plantear soluciones, abrió las puertas a interrogantes nunca antes concebidos. ¿Qué ganancia se lograría si las computadoras fuesen regidas por las leyes de la mecánica cuántica? Fueron los algoritmos de Grover [1996] y Shor [1997] los cuales despertaron el gran interés desde las ciencias de la computación en este nuevo paradigma. El primero es un algoritmo de búsqueda sobre registros desordenados, el cual provee una ganancia cuadrática de complejidad temporal frente a cualquier algoritmo clásico conocido. El segundo es un algoritmo para la factorización de números, con una ganancia exponencial.

Actualmente existen muchas áreas de investigación dentro de la computación cuántica. Por ejemplo, desde un punto de vista práctico se plantea el problema de construir el hardware de una computadora cuántica. Desde sus orígenes, en las palabras de Feinmann,

la idea es que un algoritmo cuántico sea una simulación cuántica en hardware que se comporta de acuerdo a las leyes de la física cuántica. Es decir que un experimento cuántico en un laboratorio, puede considerarse como un algoritmo. O dicho de otro modo: podemos describir el comportamiento de un sistema cuántico a través de un algoritmo. La pregunta es, ¿podemos realizar el experimento cuántico que describe un algoritmo dado? Allí es donde se manifiesta el desafío técnico.

Otra área es la de desarrollar algoritmos que obtengan una ganancia con respecto a su contraparte clásica. En general los algoritmos de Grover y Shor mencionados anteriormente se consideran como los ejemplos canónicos de aceleración obtenida gracias a la computación cuántica. Muchos otros algoritmos cuánticos son derivados de ellos. La pregunta aquí es ¿qué otros algoritmos podemos obtener que nos den una ganancia respecto a los algoritmos clásicos?

Otra rama de investigación es la del diseño de lenguajes de programación que permitan expresar los algoritmos cuánticos de una manera amigable, y quizá permitiendo descubrir nuevos algoritmos al tener una herramienta de alto nivel para pensarlos.

Desde un punto de vista más fundamental, y como lo expresara Preskill en la cita que abre este capítulo, los fundamentos lógicos detrás de la computación cuántica, siguen siendo un misterio. Si bien existe una lógica cuántica [Birkhoff y von Neumann, 1936], ésta fue propuesta muchos años antes de la computación cuántica, por lo que encontrar la correspondencia entre computación y lógica cuántica no es trivial. Esta área tiene muchas subáreas con metodologías diferentes. En particular, el estudio de semántica de lenguajes de programación sigue este objetivo. En este caso no se persigue el estudio del lenguaje en sí mismo, sino que el objetivo es el estudio de la lógica subyacente. Estudiar la lógica detrás de la computación cuántica implica estudiar la lógica detrás de la física cuántica, lo cual puede tener influencia en el desarrollo de nuevas teorías sobre el mundo que nos rodea.

En esta materia nos interesan los dos últimos aspectos: lenguajes de programación que permitan expresar el cómputo cuántico de una manera estructurada y amigable, y el estudio de propiedades de lenguajes que nos acerquen hacia una lógica computacional de la física cuántica.

Estructura de la materia y de estos apuntes En el primer encuentro, de dos días (10hs en total), y se plantea la primera introducción a la computación cuántica. Esta es una introducción para un sólo encuentro, para una introducción más extensa, se recomienda el libro de Nielsen y Chuang [2010]. En el resto de este capítulo desarrollaremos los rudimentos básicos de la computación cuántica, desde un enfoque puramente matemático (en contraposición con el enfoque físico). En el Capítulo 2 explicaremos algunos de los algoritmos más conocidos y una aplicación a la criptografía. En el segundo encuentro estudiaremos los postulados de la mecánica cuántica, desarrollados en el Capítulo 3, desde un punto de vista más físico, relacionándolo con el formalismo aprendido. En el tercer encuentro estudiaremos el cálculo lambda tipado y el isomorfismo de Curry-Howard. Finalmente, en el cuarto y último encuentro, estudiaremos extensiones cuánticas al cálculo lambda, en el paradigma de control clásico y datos cuánticos y en el paradigma de control y datos cuánticos.

1.2. Preliminares: un poco de álgebra

1.2.1. Espacio de Hilbert

TL;DR $\mathbb{C}^n$ con la suma $(+)$ y el producto $(\cdot)$ usuales, y el producto escalar definido por

$$\langle \vec{v}, \vec{w} \rangle = \langle (v_1, v_2, \ldots, v_n), (w_1, w_2, \ldots, w_n) \rangle = \sum_{i=1}^{n} v_i^* \cdot w_i$$

donde v^* es el complejo conjugado de v, es un *espacio de Hilbert*.

En el resto de la sección se define formalmente qué es un espacio de Hilbert.

Definición 1.1 (Producto escalar)**.** Sea E un espacio vectorial sobre el cuerpo $\mathbb{K}$ ($\mathbb{R}$ o $\mathbb{C}$). Un producto escalar (también llamado producto interno) definido sobre E es una función $\langle , \rangle : E \times E \to \mathbb{K}$ que verifica las siguientes propiedades.
Para todo $\vec{u}, \vec{v}, \vec{w} \in E$, $a, b \in \mathbb{K}$, se cumple:

$$\begin{cases} \langle \vec{u}, \vec{u} \rangle \geq 0 \\ \langle \vec{u}, \vec{u} \rangle = 0 \Leftrightarrow \vec{u} = \vec{0}_E \end{cases} \qquad \text{(Definida positiva)}$$

$$\langle \vec{w}, a\vec{u} + b\vec{v} \rangle = a\langle \vec{w}, \vec{u} \rangle + b\langle \vec{w}, \vec{v} \rangle \qquad \text{(Lineal por derecha)}$$

$$\langle a\vec{u} + b\vec{v}, \vec{w} \rangle = a^*\langle \vec{u}, \vec{w} \rangle + b^*\langle \vec{v}, \vec{w} \rangle \qquad \text{(Antilineal por izquierda)}$$

$$\langle \vec{u}, \vec{v} \rangle = \langle \vec{v}, \vec{u} \rangle^* \qquad \text{(Hermítica)}$$

Definición 1.2 (Espacio pre-Hilbert)**.** Un espacio pre-Hilbert es un espacio vectorial sobre $\mathbb{K}$ con producto escalar.

Observación. Todo espacio pre-Hilbert es un espacio vectorial normado con la norma

$$\|\vec{v}\| = \sqrt{\langle \vec{v}, \vec{v} \rangle}$$

Definición 1.3 (Sucesión de Cauchy)**.** Sea $\vec{v}_n$ una sucesión de vectores del espacio E. Si $\|\vec{v}_n - \vec{v}_m\| \to 0$ cuando $n, m \to \infty$, entonces la sucesión $\vec{v}_n$ es una sucesión de Cauchy. (Esto quiere decir que puedo hacer distar entre sí los términos tan poco como quiera).

Observación. Toda sucesión convergente es de Cauchy, pero no toda sucesión de Cauchy es convergente.

Definición 1.4 (Espacio completo)**.** E es completo para la norma $\|\cdot\|$, si y sólo si toda sucesión de Cauchy converge con esa norma.

Definición 1.5 (Espacio de Hilbert)**.** Un espacio pre-Hilbert completo en su norma se denomina espacio de Hilbert.

1.2.2. Productos tensoriales

En esta sección consideramos espacios vectoriales equipados con una base canónica.

Definición 1.6 (Producto tensorial)**.** Sean E y F dos espacios vectoriales con bases canónicas $B = \{\vec{b}_i \mid i \in I\}$ y $C = \{\vec{c}_j \mid j \in J\}$ respectivamente. El producto tensorial $E \otimes F$ de E y F es el espacio vectorial de base canónica $\{\vec{b}_i \otimes \vec{c}_j \mid i \in I \text{ y } j \in J\}$, donde $\vec{b}_i \otimes \vec{c}_j$ es el par ordenado formado por el vector $\vec{b}_i$ y el vector $\vec{c}_j$. La operación $\otimes$ se extiende a vectores de E y F bilinearmente:

$$(\sum_i \alpha_i \vec{b}_i) \otimes (\sum_j \beta_j \vec{c}_j) = \sum_{ij} \alpha_i \beta_j (\vec{b}_i \otimes \vec{c}_j)$$

Definición 1.7 (Producto cartesiano entre dos subconjuntos de espacios vectoriales)**.** Sean E y F dos espacios vectoriales equipados con bases B y C, y sean S y T dos subconjuntos de E y F respectivamente. Definimos el conjunto $S \times T$, subconjunto del espacio vectorial $E \otimes F$, de la siguiente manera:

$$S \times T = \{\vec{u} \otimes \vec{v} \mid \vec{u} \in S, \vec{v} \in T\}$$

Observación. $E \times F \neq E \otimes F$. Por ejemplo, si $E = F = \mathbb{C}^2$, con base canónica $\{\vec{i}, \vec{j}\}$, entonces $E \times F$ contiene a $\vec{i} \otimes \vec{i}$ y a $\vec{j} \otimes \vec{j}$, pero no a $\vec{i} \otimes \vec{i} + \vec{j} \otimes \vec{j}$, que no es producto de dos vectores de $\mathbb{C}^2$.

Definición 1.8 (Generador)**.** Sea E un espacio vectorial equipado con una base B, y $S \subseteq E$. Escribimos $\mathcal{G}(S)$ al espacio vectorial sobre $\mathbb{C}$ generado por S, es decir, que contiene todas las combinaciones lineales de elementos de S.

Observación. Si E y F son dos espacios vectoriales con bases B y C respectivamente, entonces

$$E \otimes F = \mathcal{G}(B \times C) = \mathcal{G}(E \times F)$$

La operación $\otimes$ introducida genéricamente en la Definición 1.6, puede ser definida más precisamente para matrices (y vectores, tomando matrices columna o fila) de la siguiente manera.

Definición 1.9 (Producto tensorial entre matrices)**.** El producto tensorial de dos matrices, P y Q se define como la matriz

$$P \otimes Q = \begin{pmatrix} p_{11}Q & \dots & p_{1m}Q \\ \vdots & & \vdots \\ p_{n1}Q & \dots & p_{nm}Q \end{pmatrix}$$

Ejemplos 1.10.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} & 2\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \\ 3\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} & 4\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 3 \\ 4 \end{pmatrix} \\ 2\begin{pmatrix} 3 \\ 4 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

Observación. El producto escalar, o producto interno, entre dos vectores nos da un número. El producto tensorial, o producto externo, entre dos vectores nos da un vector de mayor dimensión.

Como se dijo anteriormente, $E \times F \neq E \otimes F$, y por lo tanto:

Existen vectores de $E \otimes F$ que no son producto tensorial entre uno de E y uno de F.

Ejemplo 1.11. Consideremos el espacio $\mathbb{C}^2 \otimes \mathbb{C}^2$. Una base de $\mathbb{C}^2$ es $\{\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}\}$ Por lo tanto

$$\mathbb{C}^2 \otimes \mathbb{C}^2 = \text{Gen}(\left\{ \begin{pmatrix}0\\0\\0\\1\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \right\}) = \mathbb{C}^4$$

Tomemos $\vec{v} = (\alpha, 0, 0, \beta)^T$, con $\alpha, \beta \neq 0$. Es fácil verificar que $\vec{v} \in \mathbb{C}^4$. Sin embargo, no existen $\vec{v}_1, \vec{v}_2 \in \mathbb{C}^2$ tal que $\vec{v} = \vec{v}_1 \otimes \vec{v}_2$.

Demostración. Supongamos que existen $\vec{v}_1$ y $\vec{v}_2$ tales que $\vec{v}_1 \otimes \vec{v}_2 = \vec{v}$, entonces

$$\begin{pmatrix}a\\b\end{pmatrix} \otimes \begin{pmatrix}c\\d\end{pmatrix} = \begin{pmatrix}ac\\ad\\bc\\bd\end{pmatrix} = \begin{pmatrix}\alpha\\0\\0\\\beta\end{pmatrix} \Rightarrow \begin{cases} ac = \alpha \\ ad = 0 \\ bc = 0 \\ bd = \beta \end{cases}$$

pero este es un sistema que no tiene solución. □

1.2.3. Notación bra–ket

Notación introducida por Paul Dirac [1939] para describir estados cuánticos.

1.2.3.1. Notación bra y ket para vectores

En lugar de escribir los vectores como $\vec{v}$ la notación ket usa $|v\rangle$.
En particular definimos:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

Por lo tanto, cualquier vector de $\mathbb{C}^2$ puede escribirse como

$$\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \alpha \begin{pmatrix}1\\0\end{pmatrix} + \beta \begin{pmatrix}0\\1\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle$$

Podemos, por ejemplo, definir vectores como los siguientes

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \begin{pmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{pmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \begin{pmatrix}\frac{1}{\sqrt{2}}\\-\frac{1}{\sqrt{2}}\end{pmatrix}$$

y como estos son dos vectores ortogonales (por ende, forman una base), ahora es posible también escribir cualquier vector de $\mathbb{C}^2$ como combinación lineal de $|+\rangle$ y $|-\rangle$.
Por ejemplo:

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle = \frac{1}{\sqrt{2}}(\alpha+\beta)|+\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)|-\rangle$$

Observación. Al menos que se indique lo contrario, en el resto del apunte consideraremos el espacio complejo de dimensión $N = 2^n$, $\mathbb{C}^N = \mathbb{C}^{2^n}$.

Definición 1.12 (Bra y Ket)**.** Llamamos ket a un vector de la forma

$$|\psi\rangle = \begin{pmatrix}\alpha_1\\ \vdots\\ \alpha_N\end{pmatrix}$$

y bra a un vector de la forma

$$\langle\psi| = (\alpha_1^*, \ldots, \alpha_N^*)$$

donde $\alpha_i \in \mathbb{C}$ y α_i^* denota el conjugado de α_i.

Observaciones.

- Haciendo un abuso de notación, podemos escribir vectores como el siguiente:

$$|\alpha_1\psi_1 + \alpha_2\psi_2\rangle = \alpha_1|\psi_1\rangle + \alpha_2|\psi_2\rangle$$

- A partir la definición de bras y kets, llamamos "braket" al producto escalar:

$$\langle\psi|\phi\rangle = (\alpha_1^*, \ldots, \alpha_N^*)\begin{pmatrix}\beta_1\\ \vdots\\ \beta_N\end{pmatrix} = a \in \mathbb{C}$$

- Recordatorio de álgebra: Una *base ortonormal* de un espacio vectorial normado es una base donde todos los vectores tienen norma 1. Además, en una base, todos los vectores son ortogonales entre sí (es decir, el producto escalar entre ellos es 0). Por lo tanto:

  Dado un conjunto $B = \{|u_1\rangle, \ldots, |u_N\rangle\}$, B es una base ortonormal de $\mathbb{C}^N$ si y sólo si para todo i, j tenemos $\langle u_i|u_j\rangle = \delta_{ij}$, donde δ_{ij} es la delta de Kronecker (igual a 1 si $i = j$, y 0 en otro caso).

- Entonces, todo Ket $|\psi\rangle$ se puede expresar como $|\psi\rangle = \sum_{i=1}^N a_i|u_i\rangle$.

- Si tomamos la base canónica de $\mathbb{C}^N$, con $|u_i\rangle$ el vector i-ésimo de dicha base, podemos calcular la componente i-ésima de un vector cualquiera de la siguiente manera:

$$\langle u_i|\psi\rangle = \langle u_i|\sum_{j=1}^N a_j|u_j\rangle = \sum_{j=1}^N a_j \underbrace{\langle u_i|u_j\rangle}_{\delta_{ij}} = a_i$$

Teorema 1.13. Sea $B = \{|u_1\rangle, \ldots, |u_N\rangle\}$ una base ortonormal, entonces $\sum_{i=1}^{N} |u_i\rangle\langle u_i| = I$.

Demostración.

$$\left(\sum_{i=1}^{N} |u_i\rangle\langle u_i|\right)|\psi\rangle = \left(\sum_{i=1}^{N} |u_i\rangle\langle u_i|\right)\left(\sum_{j=1}^{N} a_j|u_i\rangle\right)$$
$$= \sum_{i=1}^{N}\sum_{j=1}^{N} a_j|u_i\rangle \underbrace{\langle u_i|u_j\rangle}_{\delta_{ij}} = \sum_{i=1}^{N} a_i|u_i\rangle = |\psi\rangle \qquad \square$$

Observaciones.

- Análogamente a los kets, todo bra $\langle\phi|$ puede ser descompuesto como $\langle\phi| = \sum_{i=1}^{N} b_i^*\langle u_i|$.
- Podemos ver que $b_i^* = \langle\phi|u_i\rangle \in \mathbb{C}$ ya que

$$\langle\phi| = \langle\phi|\underbrace{\left[\sum_{i=1}^{N} |u_i\rangle\langle u_i|\right]}_{I} = \sum_{i=1}^{N} \langle\phi|u_i\rangle\langle u_i| \quad \Rightarrow \quad b_i^* = \langle\phi|u_i\rangle$$

Observación. De aquí en más, trabajaremos sólo con los vectores normalizados de $\mathbb{C}^N$ (es decir, vectores cuya norma es 1). Esto es

$$1 = \|\psi\|^2 = \langle\psi|\psi\rangle = \left(\sum_{j=1}^{N} a_j^*\langle u_j|\right)\left(\sum_{i=1}^{N} a_i|u_i\rangle\right) = \sum_{i,j=1}^{N} a_j^* a_i \underbrace{\langle u_j|u_i\rangle}_{\delta_{ij}} = \sum_{i=1}^{N} |a_i|^2 = 1$$

Es decir, trabajamos con vectores cuya suma de los módulos al cuadrado de sus componentes es 1.

1.2.3.2. Notación bra y ket para matrices

Para toda matriz cuadrada de dimensión N a coeficientes complejos A, tenemos la siguiente representación:

$$A = \left(\underbrace{\sum_{i=1}^{N} |u_i\rangle\langle u_i|}_{I}\right) A \left(\underbrace{\sum_{j=1}^{N} |u_j\rangle\langle u_j|}_{I}\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} |u_i\rangle \underbrace{\langle u_i|\overbrace{A|u_j\rangle}^{|vector\rangle}}_{\alpha_{ij}}\langle u_j| = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_{ij}|u_i\rangle\langle u_j|$$

donde α_{ij} es la componente ij de la matriz.
Con esta representación, podemos representar el producto de una matriz por un vector de la siguiente manera:

$$A|\psi\rangle = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_{ij}|u_i\rangle\langle u_j|\right)\left(\sum_{k=1}^{N} a_k|u_k\rangle\right)$$
$$= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} \alpha_{ij}a_k|u_i\rangle \underbrace{\langle u_j|u_k\rangle}_{\delta_{jk}} = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_{ij}a_j|u_i\rangle$$

Es decir, las componentes del vector $A|\psi\rangle$ son $b_i = \sum_{j=1}^{N} \alpha_{ij} a_j$.

1.3. Bits cuánticos y operadores

1.3.1. Primera intuición

En computación clásica la unidad mínima de información es el bit, el cual puede estar en un estado 0 o 1. Leer un bit es una operación que no conlleva ninguna particularidad. En contraposición, un bit cuántico o qubit puede estar en un estado que sea una superposición de los estados 0 y 1. Un qubit es un vector de $\mathbb{C}^2$, por lo tanto lo podemos representar como $\alpha|0\rangle + \beta|1\rangle$, lo cual representa el estado que es 0 y en 1 a la vez. Leer un qubit en cambio se produce a través de una operación llamada medición, y al medir un qubit, éste colapsa, cambia su estado (dependiendo de la medición puede cambiar por ejemplo a $|0\rangle$ o $|1\rangle$, pero también podría usarse otro operador de medición que lo colapse a otra base).

1.3.2. Bits cuánticos

Definición 1.14 (Qubit)**.** Un qubit o bit cuántico es un vector normalizado (es decir, con norma 1) del espacio de Hilbert $\mathbb{C}^2$.

Observación. Considerando la base $\{|0\rangle, |1\rangle\}$ de $\mathbb{C}^2$, cualquier qubit puede escribirse como $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, con $|\alpha|^2 + |\beta|^2 = 1$.

Definición 1.15 (n-qubits)**.** Un sistema de n-qubits es un vector normalizado del espacio $\mathbb{C}^{2^n} = \bigotimes_{i=1}^{n} \mathbb{C}^2$.

Observaciones.

- En lugar de escribir $|0\rangle \otimes |1\rangle \otimes \cdots \otimes |0\rangle$ escribimos $|01\ldots0\rangle$.
- De la misma manera, en ocasiones, en lugar de $|0\rangle \otimes (\alpha|0\rangle + \beta|1\rangle)$ escribimos simplemente $|0\rangle(\alpha|0\rangle + \beta|1\rangle)$.
- La base canónica del espacio $\mathbb{C}^{2^n}$ es $\{|0\ldots00\rangle, |0\ldots01\rangle, \ldots, |1\ldots11\rangle\}$.

1.3.3. Operadores

Definición 1.16 (Operador)**.** Un operador de $\mathbb{C}^N$ es una matriz cuadrada de dimensión N a coeficientes complejos.

Definición 1.17 (Adjunto)**.** El adjunto de un operador A se nota por $A^\dagger$ y se define como el operador transpuesto y conjugado de A. Es decir, si $\alpha_{ij} = \langle u_i|A|u_j\rangle$ son las componentes de A, las componentes de $A^\dagger$ son $\alpha^*_{ji} = \langle u_j|A|u_i\rangle^* = \langle u_i|A^\dagger|u_j\rangle$.

Propiedades. Sean A y B operadores de $\mathbb{C}^N$, $a \in \mathbb{C}$ y $|\psi\rangle \in \mathbb{C}^N$

- $(A^\dagger)^\dagger = A$
- $(aA)^\dagger = a^* A^\dagger$
- $\langle A\psi| = \langle\psi|A^\dagger$
- $(A+B)^\dagger = A^\dagger + B^\dagger$
- $(AB)^\dagger = B^\dagger A^\dagger$

Definición 1.18 (Proyector)**.** A los operadores de la forma $P = |\phi\rangle\langle\phi|$ se les llama proyectores, ya que proyecta ortogonalmente un ket $|\psi\rangle$ cualquiera sobre el ket $|\phi\rangle$:

$$P|\psi\rangle = |\phi\rangle \underbrace{\langle\phi|\psi\rangle}_{a\in\mathbb{C}} = a|\phi\rangle$$

Ejemplo 1.19. Tomemos la base $\{|0\rangle, |1\rangle\}$, con $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ y $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$. Un vector $|\psi\rangle$ cualquiera puede escribirse como $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Por lo tanto

$$|0\rangle\langle 0|\psi\rangle = |0\rangle\langle 0|(\alpha|0\rangle + \beta|1\rangle) = |0\rangle(\alpha \underbrace{\langle 0|0\rangle}_{1} + \beta \underbrace{\langle 1|0\rangle}_{0}) = \alpha|0\rangle$$

Definición 1.20 (Operador hermítico)**.** Un operador A es hermítico si $A = A^\dagger$.

Definición 1.21 (Operador unitario)**.** Un operador U es unitario si $U^\dagger U = UU^\dagger = I$, o lo que es lo mismo $U^\dagger = U^{-1}$.

Propiedades. Para cualquier operador U unitario vale:

- U preserva el producto interno: $\langle U\phi|U\psi\rangle = \langle\phi|U^\dagger U|\psi\rangle = \langle\phi|\psi\rangle$
- U^{-1} es unitario.
- Si $\{|\psi_1\rangle, \ldots, |\psi_N\rangle\}$ es base ortonormal, entonces $\{U|\psi_1\rangle, \ldots, U|\psi_N\rangle\}$ también lo es.

Definición 1.22 (Operador de medición)**.** Un conjunto de proyectores $\{M_1, \ldots, M_k\}$ se dice que es un operador de medición si satisface

$$\sum_{i=1}^{k} M_i M_i^\dagger = I$$

Definición 1.23 (Compuertas cuánticas)**.** A los operadores unitarios se les llama compuertas cuánticas, como analogía a las compuertas lógicas de la computación clásica, ya que serán esos los que se utilizan para realizar el cómputo.

Observación. La mayoría de las compuertas cuánticas que usaremos a lo largo del curso serán además de operadores unitarios, hermíticos, por lo que coinciden con su inversa.

Definición 1.24 (Evolución)**.** Se dice que un sistema representado por un ket $|\psi\rangle$ evoluciona al sistema $|\phi\rangle$, cuando se realiza una de las siguientes operaciones:

- Se premultiplica por una compuerta cuántica U:

$$|\phi\rangle = U|\psi\rangle$$

- Se aplica un operador de medición $M = \{M_1, \ldots, M_k\}$ de la siguiente manera:

$$|\phi\rangle = \frac{M_i|\psi\rangle}{\sqrt{\langle\psi|M_i^\dagger M_i|\psi\rangle}} \text{ para algún } 1 \leq i \leq k$$

  La elección del M_i no se conoce de antemano, sólo se conoce la probabilidad para cada i, la cual viene dada por la siguiente ley:

$$p(i) = \langle\psi|M_i^\dagger M_i|\psi\rangle$$

Observaciones.

- Usaremos también la notación $|\psi\rangle \xrightarrow{U} |\phi\rangle$ o $|\psi\rangle \xrightarrow{M} |\phi\rangle$ para indicar que el ket $|\psi\rangle$ evoluciona al ket $|\phi\rangle$.
- Cuando se quiera hacer evolucionar sólo un qubit de un sistema de n-qubits, digamos el qubit i, se premultiplica tensorialmente $i-1$ veces y se postmultiplica $n-i-1$ veces la compuerta a aplicar por la matriz identidad. Ejemplo: U aplicada al segundo qubit de un sistema de 2–qubits, será la compuerta $I \otimes U$.

Ejemplo 1.25. Consideramos el operador medición de $\{M_0, M_1\}$ con

$$M_0 = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad M_1 = |1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Podemos verificar que $M_0M_0^\dagger + M_1M_1^\dagger = M_0 + M_1 = I$, y por lo tanto es un operador de medición.
Sea $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, entonces, la probabilidad de que el proyector que se aplique sea M_0 es

$$\begin{aligned} p(0) &= \langle\psi|M_0^\dagger M_0|\psi\rangle \\ &= (\alpha^*\langle 0| + \beta^*\langle 1|)M_0(\alpha|0\rangle + \beta|1\rangle) \\ &= |\alpha|^2\langle 0|M_0|0\rangle + \alpha^*\beta\langle 0|M_0|1\rangle + \alpha\beta^*\langle 1|M_0|0\rangle + |\beta|^2\langle 1|M_0|1\rangle \\ &= |\alpha|^2\underbrace{\langle 0|0\rangle}_{1}\underbrace{\langle 0|0\rangle}_{1} + \alpha^*\beta\underbrace{\langle 0|0\rangle}_{1}\underbrace{\langle 0|1\rangle}_{0} + \alpha\beta^*\underbrace{\langle 1|0\rangle}_{0}\underbrace{\langle 0|0\rangle}_{1} + |\beta|^2\underbrace{\langle 1|0\rangle}_{0}\underbrace{\langle 0|1\rangle}_{0} \\ &= |\alpha|^2 \end{aligned}$$

Análogamente, $p(1) = \langle\psi|M_1^\dagger M_1|\psi\rangle = \cdots = |\beta|^2$.
Dado que el vector está normalizado, $p(0) + p(1) = |\alpha|^2 + |\beta|^2 = \|\psi\| = 1$.
Luego de aplicar este operador de medición, la evolución es la siguiente. Si se aplicó el proyector M_0, el sistema queda en el siguiente estado:

$$\frac{M_0|\psi\rangle}{\sqrt{\langle\psi|M_0^\dagger M_0|\psi\rangle}} = \frac{M_0|\psi\rangle}{\sqrt{p(0)}} = \frac{\alpha}{|\alpha|}|0\rangle$$

Este estado está normalizado ya que $\left|\frac{\alpha}{|\alpha|}\right|^2 = \frac{|\alpha|^2}{|\alpha|^2} = 1$.

Análogamente si se aplicó M_1 se obtiene $\frac{M_1|\psi\rangle}{\sqrt{p(1)}} = \frac{\beta}{|\beta|}|1\rangle$.

Definición 1.26 (Compuertas más comunes y operadores de Pauli)**.** Las compuertas cuánticas más importantes, por su utilidad en el diseño de algoritmos, son las siguientes:

- La compuerta H de Hadamard:

$$\begin{matrix} H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{matrix} \quad \text{donde:} \quad H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

- La identidad I:

$$\begin{matrix} I|0\rangle = |0\rangle \\ I|1\rangle = |1\rangle \end{matrix} \quad \text{donde:} \quad I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- La negación X:

$$\begin{matrix} X|0\rangle = |1\rangle \\ X|1\rangle = |0\rangle \end{matrix} \quad \text{donde:} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- El cambio de fase Z:

$$\begin{matrix} Z|0\rangle = |0\rangle \\ Z|1\rangle = -|1\rangle \end{matrix} \quad \text{donde:} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- La No-controlada $CNOT$:

$$\begin{matrix} CNOT|0x\rangle = |0x\rangle \\ CNOT|1x\rangle = |1\rangle \otimes X|x\rangle \end{matrix} \quad \text{donde:} \quad CNOT = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix}$$

En particular, las matrices I, X, iXZ y Z son las llamadas *matrices de Pauli* en honor a Wolfgang Pauli

1.4. Teorema del no-clonado

El teorema de no-clonado [Wootters y Zurek, 1982] dice que es imposible construir una máquina universal de clonado. Es decir, no podemos copiar un qubit arbitrario, ya que no existe ningún método que pueda copiarlo sin saber su estado preciso, y como la medición cambia el qubit, no podemos saber su estado preciso. En consecuencia, no podemos copiar un qubit arbitrario.

Teorema 1.27 (No-cloning)**.** No existe ninguna compuerta cuántica U tal que para algún $|\phi\rangle \in \mathbb{C}^N$ y $\forall|\psi\rangle \in \mathbb{C}^N$ se cumpla $U|\psi\phi\rangle = |\psi\psi\rangle$.

Demostración. Supongamos que existe la operación U de la cual se habla en el teorema, entonces, dados cualesquiera $|\psi\rangle, |\phi\rangle \in \mathbb{C}^N$, se cumple

$$\begin{aligned} U|\psi\phi\rangle &= |\psi\psi\rangle \\ U|\varphi\phi\rangle &= |\varphi\varphi\rangle \end{aligned}$$

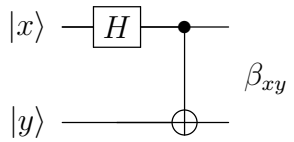
Por lo tanto, $\langle U\psi\phi|U\varphi\phi\rangle = \langle\psi\psi|\varphi\varphi\rangle$. Sin embargo, por un lado

$$\langle U\psi\phi|U\phi\varphi\rangle = \langle\psi\phi|U^\dagger U|\varphi\phi\rangle = \langle\psi\phi|\varphi\phi\rangle = \langle\psi|\varphi\rangle\langle\phi|\phi\rangle = \langle\psi|\varphi\rangle$$

Mientras por el otro $\langle\psi\psi|\varphi\varphi\rangle = \langle\psi|\varphi\rangle\langle\psi|\varphi\rangle = \langle\psi|\varphi\rangle^2$
Pero si $\langle\psi|\phi\rangle = \langle\psi|\phi\rangle^2$, entonces $\langle\psi|\phi\rangle = 0$ o $\langle\psi|\phi\rangle = 1$, lo cual es imposible: 0 implica que los dos vectores tomados al azar son ortogonales, y 1 que son iguales. □

1.5. Estados de Bell

Consideremos el siguiente *circuito* cuántico



Es decir, partiendo del estado inicial $|xy\rangle$, se aplica H al primer qubit. Luego se aplica $CNOT$ a ambos, donde el primero es el de control (marcado con el punto negro). En otras palabras, este circuito representa la siguiente ecuación:

$$\beta_{xy} = CNOT(H \otimes I)|xy\rangle$$

Las posibles salidas de este circuito, cuando x e y varían entre 0 y 1 son las siguientes:

$$|00\rangle \xrightarrow{H(1)} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \xrightarrow{CNOT(1,2)} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \beta_{00}$$
$$|01\rangle \xrightarrow{H(1)} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|1\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |11\rangle) \xrightarrow{CNOT(1,2)} \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) = \beta_{01}$$
$$|10\rangle \xrightarrow{H(1)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle) \xrightarrow{CNOT(1,2)} \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = \beta_{10}$$
$$|11\rangle \xrightarrow{H(1)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|1\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |11\rangle) \xrightarrow{CNOT(1,2)} \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) = \beta_{11}$$

Observación. $\beta_{00} = (X \otimes I)\beta_{01} = (Z \otimes I)\beta_{10} = (XZ \otimes I)\beta_{11}$.

A estos cuatro estados se les llama *Estados de Bell*, en honor a John S. Bell. Estos son estados *entrelazados*, es decir, estados que no pueden representarse como el producto tensorial de dos estados individuales.

A los estados entrelazados también se les llama estados EPR por Einstein, Podolsky, y Rosen [1935] quienes detectaron, en pleno auge de las formulaciones de la teoría cuántica, que existía una acción a distancia que parecía no razonable. Por muchos años se llamó la "paradoja EPR". Lo que determinaron es que cuando se tiene un par entrelazado (físicamente el estado representa por ejemplo el *spin* en un par de electrones, o la polarización de un par de fotones), sucede que cuando se colapsa (por acción de la medición) un estado del par, el segundo también colapsará, incluso cuando físicamente se encuentren a años luz de distancia. Con el tiempo se demostró experimentalmente que esto es exactamente lo que sucede, y por lo tanto no hay paradoja. También se demuestra que esto no contradice la teoría de la relatividad (que entre otras cosas determina que nada puede viajar a mayor velocidad que la luz, ni siquiera la información), ya que no hay trasmisión de información en este colapso a distancia.

Matemáticamente la acción de medir un estado de un par se ve con el siguiente ejemplo:

Ejemplo 1.28. Consideremos el siguiente operador de medición: $M = \{M_0, M_1\}$ donde $M_0 = |0\rangle\langle 0|$ y $M_1 = |1\rangle\langle 1|$.

Aplicando este operador al primer qubit del estado β_{00}, se obtiene uno de los siguientes resultados:

- Si se aplica el proyector M_0 (el cual lo expresamos como $M_0 \otimes I$ para que se aplique M_0 al primer qubit y la identidad al segundo), el estado resultante será

$$\frac{(M_0 \otimes I)\beta_{00}}{\sqrt{p(0)}} = \frac{(|00\rangle\langle 00| + |01\rangle\langle 01|)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)}{\sqrt{\frac{1}{\sqrt{2}}(\langle 00| + \langle 11|)(|00\rangle\langle 00| + |01\rangle\langle 01|)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)}}$$
$$= \frac{\frac{1}{\sqrt{2}}(|00\rangle\langle 00|00\rangle)}{\sqrt{\frac{1}{2}\langle 00|00\rangle\langle 00|00\rangle}} = |00\rangle$$

  Análogamente, si se aplica M_1 se obtiene $|11\rangle$

Es decir, al medir el primer qubit del estado entrelazado β_{00}, se obntiene $|00\rangle$ o $|11\rangle$, es decir que ambos qubits colapsan.

1.6. Usando los estados de Bell

Como se mencionó en la sección anterior, el colapso de un par entrelazado no transmite información (y por eso no viola la teoría de la relatividad), sin embargo, es posible utilizar dicho colapso como canal de comunicación, el cual necesita también de un canal clásico para terminar la transmisión (y por ende, el canal clásico implica todas las limitaciones impuestas por la relatividad).
El algoritmo cuántico descripto en la sección 1.6.1, descripto por primera vez por Bennett y Wiesner [1992], permite transmitir dos bit clásicos, enviando sólo un bit cuántico, utilizando un par entrelazado como canal de comunicación. Es llamado "codificación superdensa" ya que se trata de codificar dos bits de información en un bit cuántico, o dicho de otro modo: dos bits de información en el estado de una partícula cuántica.
El algoritmo descripto en la sección 1.6.2, descripto por primera vez por Bennett, Brassard, Crépeau, Jozsa, Peres, y Wootters [1993], permite enviar un bit cuántico enviando dos bit clásicos y utilizando un par entrelazado como canal de comunicación. Es llamado "teleportación cuántica" ya que se trata de mover el valor de un bit cuántico (recordemos que un bit cuántico no puede ser copiado (ver Teorema 1.27)) a otro bit cuántico, o dicho de otro modo: se trata de teletransportar el estado de una partícula a una nueva partícula, destruyendo la primera.

1.6.1. Codificación superdensa

El objetivo de esta técnica es transmitir 2 bits clásicos enviando tan sólo 1 qubit.
Los pasos a seguir por el emisor (a quien llamaremos "Alice") y el receptor (a quien llamaremos "Bob") son los siguientes.

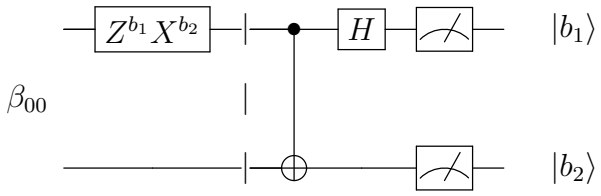
1. Alice y Bob preparan un estado β_{00}.

2. Alice se queda con el primer qubit del par y Bob se lleva el segundo. Podemos considerar que estos dos pasos son la preparación del canal cuántico.

Observación. El estado entrelazado no se puede separar en el sentido de que no puede considerarse matemáticamente como un qubit multiplicado tensorialmente por otro qubit. Debemos considerarlos como un vector del espacio $\mathbb{C}^2 \otimes \mathbb{C}^2$, es decir, un vector de dimensión 4. Pero físicamente son un par de electrones, o fotones (u otra partícula elemental), las cuales sí pueden ser separadas físicamente (más allá de que no es trivial el problema experimental que representa manipular dichas partículas sin que interaccionen con el ambiente).

3. Alice aplica una transformación a su qubit, de acuerdo a los bits que quiere enviar: $Z^{b_1}X^{b_2}$, donde $C^0 = I$ y $C^1 = C$.
4. Alice envía su qubit a Bob.
5. Bob aplica CNOT a los dos elementos del par y luego Hadamard al primero.
6. Bob realiza una medición.

El circuito completo queda de la siguiente manera



donde la línea punteada determina el paso 4, en el que Alice envía su qubit a Bob.

Ejemplo 1.29. Se quiere enviar los bits 11. Por lo tanto se aplica $(ZX \otimes I)$ a β_{00}, con lo que se obtiene β_{11} (en general, la aplicación de la compuerta $Z^{b_1}X^{b_2}$ cambia el estado β_{00} a $\beta_{b_1 b_2}$):

$$\begin{aligned}(ZX \otimes I)\beta_{00} &= (Z \otimes I)\left((X \otimes I)\beta_{00}\right) \\ &= (Z \otimes I)\left((X \otimes I)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\right) \\ &= (Z \otimes I)\left(\frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)\right) \\ &= \frac{1}{\sqrt{2}}(-|10\rangle + |01\rangle) = \beta_{11}\end{aligned}$$

El resto del circuito (a partir de la línea punteada vertical) es el circuito inverso al de Bell, y como toda compuerta unitaria es tal que $U = U^{-1}$, aplicando el circuito inverso al de Bell se obtiene los estados iniciales. En este caso, $|11\rangle$.

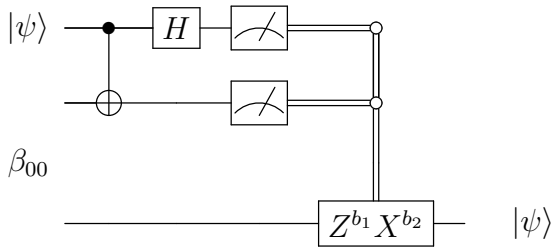
1.6.2. Teleportación cuántica

El objetivo de esta técnica es transmitir un qubit mediante el envío de dos bits clásicos. Los pasos a seguir por Alice y Bob son los siguientes.

1. Alice y Bob preparan un estado β_{00}.

2. Alice se queda con el primer qubit del par y Bob se lleva el segundo.
3. Alice aplica CNOT entre el qubit a transmitir y el primero del par β_{00}, y luego Hadamard al primero.
4. Alice realiza una medición sobre los dos qubits en su posesión y envía el resultado de la medición (2 bits clásicos) a Bob.
5. Bob aplica una transformación sobre su qubit, de acuerdo a los bits recibidos: $Z^{b_1}X^{b_2}$.

El circuito completo queda de la siguiente manera



donde $|\psi\rangle$ es el qubit a transmitir (o "teleportar").

Ejemplo 1.30. Se quiere transmitir el qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, entonces

$$\begin{aligned}
|\psi\rangle \otimes \beta_{00} &= (\alpha|0\rangle + \beta|1\rangle)\left(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\right) \\
&= \frac{1}{\sqrt{2}}(\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle)) \\
&\xrightarrow{CNOT(1,2)} \frac{1}{\sqrt{2}}(\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle)) \\
&\xrightarrow{H(1)} \frac{1}{\sqrt{2}}\left(\alpha\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)(|10\rangle + |01\rangle)\right) \\
&= \frac{1}{2}[|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)] \\
&= \frac{1}{2}\sum_{b_1=0}^{1}\sum_{b_2=0}^{1}|b_1 b_2\rangle(X^{b_2}Z^{b_1})|\psi\rangle
\end{aligned}$$

Por lo tanto, aplicando $Z^{b_1}X^{b_2}$, Bob obtendrá el estado original $|\psi\rangle$. (Nótese que para toda compuerta U, $U = U^{-1}$).

Observación. Si se quiere escribir la compuerta —$\boxed{Z^{b_1}X^{b_2}}$— como dos dos compuertas, debe escribirse —$\boxed{X^{b_2}}$—$\boxed{Z^{b_1}}$—, ya que en $Z^{b_1}X^{b_2}$ primero se aplica la compuerta X^{b_2} y luego Z^{b_1}.

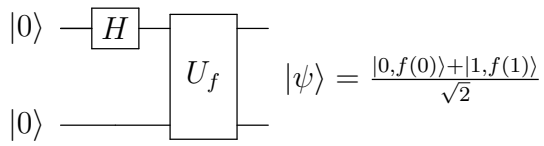
1.7. Paralelismo Cuántico

Consideremos una función $f : \{0,1\} \to \{0,1\}$. Clásicamente para obtener todos los resultados posibles de esta función, es necesario evaluarla tantas veces como sea el cardinal del dominio (2 en este caso, una evaluación para la entrada 0, y otra para la entrada 1).

Esta es una función que toma un bit y devuelve un bit. Si fuese un bit cuántico, sería posible evaluar la función en una superposición de 0 y 1 (por ejemplo $\frac{1}{2}(|0\rangle + |1\rangle)$), lo cual nos daría como resultado una superposición de f aplicada a 0 y a 1.
El método es el siguiente. Primero se debe construir una matriz unitaria U_f de $\mathbb{C}^4$ que calcule la función, de la siguiente manera:

$$U_f|x, 0\rangle = |x, f(x)\rangle$$

En realidad, aunque vamos a usar la definición que acabamos de dar, se debe definir también qué sucede cuando el segundo qubit es $|1\rangle$, por lo que esta compuerta se define más generalmente como $U_f|x, y\rangle = |x, y \oplus f(x)\rangle$, donde $\oplus$ es la suma módulo 2.
Lo que se pretende es aplicar f a todas las entradas posibles, por lo que primero se aplicará Hadamard al $|0\rangle$, a fin de obtener una superposición, y luego se aplicará la compuerta U_f. El circuito es el siguiente:



Es decir:

$$|00\rangle \xrightarrow{H(1)} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \xrightarrow{U_f} \frac{1}{\sqrt{2}}(|0, f(0)\rangle + |1, f(1)\rangle)$$

La salida de este circuito es un estado que es superposición de todos los resultados posibles de la aplicación de la función f. Y la compuerta U_f fue utilizada una sola vez. El problema ahora pasa porque el resultado es una superposición de todos los resultados posibles, y al querer leerlo (es decir, al medirlo), éste colapsará a uno de los dos. El problema de los algoritmos cuánticos pasa por utilizar la superposición de manera inteligente para aprovechar el paralelismo, pero obteniendo el resultado buscado y no una superposición de resultados sin utilidad.
En el siguiente capítulo mostraremos algunos de los algoritmos que, haciendo uso del paralelismo, consiguen ganancias en complejidad respecto a su contrapartida clásica.

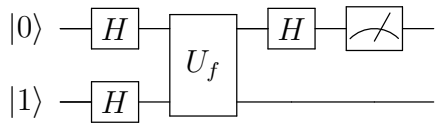
Capítulo 2

Algoritmos cuánticos y aplicación a criptografía

En este capítulo veremos algunos de los algoritmos cuánticos más conocidos. En particular, los algoritmos de Deutsch [1985] y de Deutsch y Jozsa [1992], que pueden considerarse como los primeros algoritmos cuánticos que hacen uso del paralelismo (ver Sección 1.7). El algoritmo de Grover [1996], que es uno de los que motivó que los investigadores en computación se interesaran en el área. No se incluye el algoritmo de Shor [1997], el otro importante algoritmo que motivó a investigadores en computación a adentrarse en el área. Finalmente, el último ejemplo es una aplicación directa de la física cuántica en criptografía, diseñado por Bennett y Brassard [1984], la cual no sigue el esquema de los otros algoritmos cuánticos presentados, pero es también el puntapié de un área de investigación activa dentro de la computación cuántica.

2.1. Algoritmo de Deutsch

El objetivo de este algoritmo es saber si una función que toma un bit y devuelve un bit, es constante o no.
El algoritmo se resume en el siguiente circuito



Observación. U_f es la compuerta definida en la Sección 1.7, la cual consideraremos que existe sin dar más detalles de su construcción.

$$U_f|x, y\rangle = |x, y \oplus f(x)\rangle$$

Las primeras dos compuertas Hadamard, aplicadas a $|0\rangle$ y $|1\rangle$, producen lo siguiente:

$$|01\rangle \xrightarrow{H(1,2)} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}(|x, 0\rangle - |x, 1\rangle) \quad (2.1)$$

donde $|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ es una abreviación introducida por comodidad.

La aplicación de U_f sobre el estado (2.1) produce el siguiente estado:

$$\begin{aligned} & U_f(\frac{1}{\sqrt{2}}(|x,0\rangle - |x,1\rangle)) \\ & = \frac{1}{\sqrt{2}}\left(U_f|x,0\rangle - U_f|x,1\rangle\right) \\ & = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0,f(0)\rangle + |1,f(1)\rangle) - \frac{1}{\sqrt{2}}(|0,1\oplus f(0)\rangle + |1,1\oplus f(1)\rangle)\right) \\ & = \frac{1}{2}\left(|0,f(0)\rangle + |1,f(1)\rangle - |0,1\oplus f(0)\rangle - |1,1\oplus f(1)\rangle\right) \end{aligned} \tag{2.2}$$

Si $f(0) \neq f(1)$, (2.2) es igual a

$$\pm\frac{1}{2}\left(|00\rangle + |11\rangle - |01\rangle - |10\rangle\right) = \pm\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

en cambio si $f(0) = f(1)$,

$$\pm\frac{1}{2}\left(|00\rangle + |10\rangle - |01\rangle - |11\rangle\right) = \pm\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Es decir, el primer qubit es $\pm|-\rangle$, si $f(0) \neq f(1)$ y $\pm|+\rangle$ si $f(0) = f(1)$. Aplicando Hadamard al primer qubit, obtenemos $|1\rangle$ si éste era $|-\rangle$ y $|0\rangle$ si éste era $|+\rangle$.

$$\text{Si } f(0) \neq f(1), \text{ aplicando Hadamard se obtiene } \pm|1\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$
$$\text{Si } f(0) = f(1), \text{ aplicando Hadamard se obtiene } \pm|0\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

es decir, aplicando Hadamard, se obtiene

$$\pm|f(0)\oplus f(1)\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$$

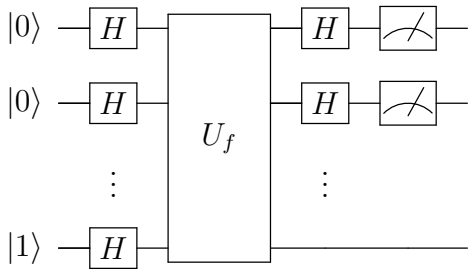
Dado que el primer qubit es $|0\rangle$ o $|1\rangle$, podemos medirlo y nos dará con probabilidad 1 el valor 0 si f es constante y con probabilidad 1 el valor 1 si f no lo es.

Observación. Este algoritmo hace uso del paralelismo, ya que la evaluación de la función se realiza una vez sobre el estado en superposición de 0 y 1. El algoritmo clásico equivalente haría dos evaluaciones de la función y una comparación.

2.2. Algoritmo de Deutsch-Jotza

Este algoritmo es una generalización del anterior. Dada una función que toma n bits y devuelve uno, el algoritmo permite distinguir si la función es constante o balanceada (o sea, con la mitad de las entradas devuelve 0 y con la otra mitad 1). Sólo se distinguen esos dos casos, el algoritmo no es útil para otro tipo de funciones.

El circuito es el siguiente:



La entrada de este algoritmo son $n+1$ qubits: $|0\rangle^{\otimes n}|1\rangle = |0\ldots01\rangle$.
Aplicando las $n+1$ compuertas Hadamard sobre la entrada, se obtiene

$$\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)^{\otimes n}\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) = \sum_{\overline{x}\in\{0,1\}^n}\frac{|\overline{x}\rangle}{\sqrt{2^n}}\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \tag{2.3}$$

La compuerta U_f que se utiliza es una generalización del caso anterior definida por

$$U_f|\overline{x},y\rangle = |\overline{x},y\oplus f(\overline{x})\rangle$$

donde $\overline{x}$ son cadenas de n bits.
Es decir

$$U_f|\overline{x},0\rangle = |\overline{x},f(\overline{x})\rangle \qquad\qquad U_f|\overline{x},1\rangle = |\overline{x},1\oplus f(\overline{x})\rangle$$

Por lo tanto, aplicando U_f sobre el estado (2.3) se obtiene

$$\begin{aligned}
U_f\left(\sum_{\overline{x}\in\{0,1\}^n}\frac{|\overline{x}\rangle}{\sqrt{2^n}}\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]\right) &= \sum_{\overline{x}\in\{0,1\}^n}\frac{1}{\sqrt{2^n}}U_f|\overline{x}\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \\
&= \sum_{\overline{x}\in\{0,1\}^n}\frac{1}{\sqrt{2^{n+1}}}\left(U_f|\overline{x},0\rangle - U_f|\overline{x},1\rangle\right) \\
&= \sum_{\overline{x}\in\{0,1\}^n}\frac{1}{\sqrt{2^{n+1}}}\left(|\overline{x},f(\overline{x})\rangle - |\overline{x},1\oplus f(\overline{x})\rangle\right) \\
&= \sum_{\overline{x}\in\{0,1\}^n}\frac{1}{\sqrt{2^n}}|\overline{x}\rangle\left(\frac{|f(\overline{x})\rangle - |1\oplus f(\overline{x})\rangle}{\sqrt{2}}\right)
\end{aligned} \tag{2.4}$$

Para simplificar la notación, la compuerta Hadamard puede expresarse como sigue

$$\left.\begin{array}{l} H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle) \\ H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle-|1\rangle) \end{array}\right\} \Rightarrow H|x\rangle = \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|y\rangle$$

De la misma manera, es posible generalizar la aplicación de H a n qubits como sigue:

$$\begin{aligned}
H^{\otimes n}|x_1\ldots x_n\rangle &= \left(\frac{1}{\sqrt{2}}\sum_{z_1\in\{0,1\}}(-1)^{x_1z_1}|z_1\rangle\right)\cdots\left(\frac{1}{\sqrt{2}}\sum_{z_n\in\{0,1\}}(-1)^{x_nz_n}|z_n\rangle\right) \\
&= \frac{1}{\sqrt{2^n}}\sum_{\overline{z}\in\{0,1\}^n}(-1)^{\overline{x}\cdot\overline{z}}|\overline{z}\rangle
\end{aligned}$$

donde $\overline{x}\cdot\overline{z} = x_1z_1+\ldots+x_nz_n$.

Con esta notación, se aplica Hadamard a los primeros n qubits del estado (2.4) (es decir, al ket $|\overline{x}\rangle$), obteniendo

$$\begin{aligned}\sum_{\overline{x}\in\{0,1\}^n} \frac{1}{\sqrt{2^n}} \left(\frac{1}{\sqrt{2^n}} \sum_{\overline{z}\in\{0,1\}^n} (-1)^{\overline{x}\cdot\overline{z}} |\overline{z}\rangle \right) \left(\frac{|f(\overline{x})\rangle - |1 \oplus f(\overline{x})\rangle}{\sqrt{2}} \right) \\ = \sum_{\overline{x}\in\{0,1\}^n} \sum_{\overline{z}\in\{0,1\}^n} \frac{(-1)^{\overline{x}\cdot\overline{z}} |\overline{z}\rangle}{2^n} \left(\frac{|f(\overline{x})\rangle - |1 \oplus f(\overline{x})\rangle}{\sqrt{2}} \right)\end{aligned} \tag{2.5}$$

Casos:

- Si f es constante, el estado (2.5) es

  $$\pm \sum_{\overline{x}\in\{0,1\}^n} \sum_{\overline{z}\in\{0,1\}^n} \frac{(-1)^{\overline{x}\cdot\overline{z}} |\overline{z}\rangle}{2^n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

  Cuando $\overline{z} = 0$, los primeros n qubits son

  $$\pm \sum_{\overline{x}\in\{0,1\}^n} \frac{|0\rangle^{\otimes n}}{2^n} = \pm \frac{2^n}{2^n} |0\rangle^{\otimes n} = \pm |0\rangle^{\otimes n}$$

  Por lo tanto, dado que este vector tiene norma 1, el resto de los términos de la suma deben anularse, debido a que el resultado tiene que ser forzosamente un vector normalizado. Por lo tanto, cuando f es constante, el estado (2.5) es

  $$\pm |0\rangle^{\otimes n} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$$

  Es decir, midiendo los primeros n qubits se obtiene $0 \ldots 0$ en este caso.

- Si f es balanceada (50 % de las veces devuelve 0 y 50 % devuelve 1), entonces para $\overline{z} = 0$

  $$\sum_{\overline{x}\in\{0,1\}^n} \frac{|0\rangle^{\otimes n}}{2^n} \left(\frac{|f(\overline{x})\rangle - |1 \oplus f(\overline{x})\rangle}{\sqrt{2}} \right) = \sum_{\overline{x}\in\{0,1\}^n} (-1)^{\overline{x}} \frac{|0\rangle^{\otimes n}}{2^n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = 0$$

  Es decir que los primeros n qubits no incluyen al qubit $|0\rangle^{\otimes n}$, y por lo tanto, al medir los primeros n qubits no se puede obtener $0 \ldots 0$ en este caso.

Conclusión: Si se obtiene $|0\rangle^{\otimes n}$ a la salida de la medición, la función es constante, en otro caso la función es balanceada.

2.3. Algoritmo de Búsqueda de Grover

Antes de analizar este algoritmo, son necesarias algunas compuertas extras: la compuerta *Oráculo* (Sección 2.3.1), y la compuerta de *inversión sobre el promedio* (Sección 2.3.2).

2.3.1. Oráculo

Dada una función de un bit en un bit f, la compuerta U_f definida en la Sección 1.7, es $U_f|x,y\rangle = |x, y \oplus f(x)\rangle$.
Si se elije $y = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, entonces

$$\begin{aligned} U_f|x,y\rangle &= U_f\left(|x\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) \\ &= \frac{1}{\sqrt{2}}\left(U_f|x,0\rangle - U_f|x,1\rangle\right) \\ &= \frac{1}{\sqrt{2}}(|x,f(x)\rangle - |x,1\oplus f(x)\rangle) \\ &= |x\rangle\frac{1}{\sqrt{2}}(|f(x)\rangle - |1\oplus f(x)\rangle) \\ &= (-1)^{f(x)}|x,y\rangle \end{aligned}$$

Dado que U_f no modifica el estado y, es posible omitirlo y tomarlo como parte de la definición de la compuerta. Entonces, definimos la compuerta

$$U|x\rangle = (-1)^{f(x)}|x\rangle$$

a la cual se le llama *Oráculo*.

2.3.2. Inversión sobre el promedio

Sea el estado $|\phi\rangle = {}^1/\sqrt{2^n}\sum_{\overline{x}\in\{0,1\}^n}|\overline{x}\rangle$. Definimos la compuerta de *Inversión sobre el promedio* como $G = 2|\phi\rangle\langle\phi| - I$. Es decir

$$\begin{aligned} G &= 2|\phi\rangle\langle\phi| - I \\ &= 2\begin{pmatrix} \frac{1}{\sqrt{2^n}} \\ \vdots \\ \frac{1}{\sqrt{2^n}} \end{pmatrix}_{2^n}\left(\frac{1}{\sqrt{2^n}}\cdots\frac{1}{\sqrt{2^n}}\right)_{2^n} - I \\ &= \begin{pmatrix} \frac{2}{2^n}-1 & \frac{2}{2^n} & \cdots & \frac{2}{2^n} \\ \frac{2}{2^n} & \frac{2}{2^n}-1 & \cdots & \frac{2}{2^n} \\ \vdots & \vdots & & \vdots \\ \frac{2}{2^n} & \frac{2}{2^n} & \cdots & \frac{2}{2^n}-1 \end{pmatrix}_{2^n\times 2^n} \end{aligned}$$

La aplicación de G sobre un estado cualquiera $|\psi\rangle = \sum_{\overline{x}\in\{0,1\}^n} a_{\overline{x}}|\overline{x}\rangle$ es la siguiente

$$\begin{array}{c|c} G|\psi\rangle & \begin{pmatrix} a_0 \\ \vdots \\ a_{2^n-1} \end{pmatrix} \\ \hline \begin{pmatrix} \frac{2}{2^n}-1 & \cdots & \frac{2}{2^n} \\ \vdots & & \vdots \\ \frac{2}{2^n} & \cdots & \frac{2}{2^n}-1 \end{pmatrix} & \begin{pmatrix} \left(\sum_{\overline{x}\in\{0,1\}^n} \frac{2a_{\overline{x}}}{2^n}\right) - a_0 \\ \vdots \\ \left(\sum_{\overline{x}\in\{0,1\}^n} \frac{2a_{\overline{x}}}{2^n}\right) - a_{2^n-1} \end{pmatrix} \end{array}$$

Es decir:

$$G\left(\sum_{\overline{x}\in\{0,1\}^n} a_{\overline{x}}|\overline{x}\rangle\right) = \sum_{\overline{x}\in\{0,1\}^n}\left[\left(\sum_{\overline{y}\in\{0,1\}^n}\frac{2a_{\overline{y}}}{2^n}\right) - a_{\overline{x}}\right]|\overline{x}\rangle = \sum_{\overline{x}\in\{0,1\}^n}(2A - a_{\overline{x}})|\overline{x}\rangle$$

donde A es el promedio de los $a_{\overline{x}}$.

2.3.3. El algoritmo

El algoritmo de Grover es un algoritmo de búsqueda sobre una lista desordenada. Suponemos una lista de tamaño N, con $N = 2^n$ (observar que siempre es posible aumentar la lista con datos irrelevantes para cumplir la condición sobre N). Los índices de la lista son $\overline{x} \in 0, 1^n$, es decir $\overline{x} = 0 \ldots 2^n - 1$.
El objetivo del algoritmo es localizar el $\overline{x}_0$ tal que $f(\overline{x}_0) = 1$, para una función booleana f dada.
El input del circuito es $|0\rangle^{\otimes n}$.

2.3.3.1. Paso 1: Se aplica Hadamard ($H^{\otimes n}$)

El primer paso es generar una superposición en todos los qubits.

$$|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}}\sum_{\overline{x}\in\{0,1\}^n}|\overline{x}\rangle \tag{2.6}$$

Este estado es una superposición de todos los elementos de la lista. La idea del algoritmo es subir la probabilidad de que al medir este estado obtengamos el elemento $\overline{x}_0$.

2.3.3.2. Paso 2: Se aplica el oráculo (U)

Aplicar el oráculo es el equivalente a aplicar la función booleana f sobre la superposición.

$$(2.6) \xrightarrow{U} \frac{1}{\sqrt{2^n}}\sum_{\overline{x}\in\{0,1\}^n}(-1)^{f(\overline{x})}|\overline{x}\rangle \tag{2.7}$$

2.3.3.3. Paso 3: Se aplica la inversión sobre el promedio (G)

$$
\begin{aligned}
(2.7) &= \sum_{\overline{x}\in\{0,1\}^n} \underbrace{\left[\frac{(-1)^{f(\overline{x})}}{\sqrt{2^n}}\right]}_{a_{\overline{x}}} |\overline{x}\rangle \\
&\xrightarrow{G} \sum_{\overline{x}\in\{0,1\}^n} (2A - a_{\overline{x}})|\overline{x}\rangle \\
&= \sum_{\overline{x}\in\{0,1\}^n} \left[\left(2\sum_{\overline{y}\in\{0,1\}^n} \frac{(-1)^{f(\overline{y})}}{2^n\sqrt{2^n}}\right) - \frac{(-1)^{f(\overline{x})}}{\sqrt{2^n}}\right] |\overline{x}\rangle \\
&= \sum_{\overline{x}\in\{0,1\}^n} \left[\left(2\sum_{\substack{\overline{y}\in\{0,1\}^n \\ \overline{y}\neq\overline{x}}} \frac{(-1)^{f(\overline{y})}}{2^n\sqrt{2^n}}\right) + \frac{2(-1)^{f(\overline{x})}}{2^n\sqrt{2^n}} - \frac{(-1)^{f(\overline{x})}}{\sqrt{2^n}}\right] |\overline{x}\rangle \\
&= \sum_{\overline{x}\in\{0,1\}^n} \left[\left(2\sum_{\substack{\overline{y}\in\{0,1\}^n \\ \overline{y}\neq\overline{x}}} \frac{(-1)^{f(\overline{y})}}{2^n\sqrt{2^n}}\right) + \frac{2-2^n}{2^n\sqrt{2^n}}(-1)^{f(\overline{x})}\right] |\overline{x}\rangle
\end{aligned}
\tag{2.8}
$$

En el estado (2.8), el término $\overline{x} = \overline{x}_0$, con $f(\overline{x}_0) = 1$, el cual estamos buscando es el siguiente:

$$
\begin{aligned}
\left[\left(2\sum_{\substack{\overline{y}\in\{0,1\}^n \\ \overline{y}\neq\overline{x}_0}} \frac{1}{2^n\sqrt{2^n}}\right) + \frac{2^n-2}{2^n\sqrt{2^n}}\right] |\overline{x}_0\rangle &= \left[\frac{2}{2^n\sqrt{2^n}}(2^n-1) + \frac{2^n-2}{2^n\sqrt{2^n}}\right] |\overline{x}_0\rangle \\
&= \left[\frac{2^{n+1}+2^n-4}{2^n\sqrt{2^n}}\right] |\overline{x}_0\rangle
\end{aligned}
$$

mientras que los otros términos, donde $\overline{x} \neq \overline{x}_0$, son

$$
\left[\left(2\sum_{\substack{\overline{y}\in\{0,1\}^n \\ \overline{y}\neq\overline{x}_0 \\ \overline{y}\neq\overline{x}}} \frac{1}{2^n\sqrt{2^n}}\right) + \frac{2(-1)}{2^n\sqrt{2^n}} + \frac{2-2^n}{2^n\sqrt{2^n}}\right] |\overline{x}\rangle = \left[\frac{2^{n+1}-2^n-4}{2^n\sqrt{2^n}}\right] |\overline{x}\rangle
$$

El algoritmo ha cambiado las amplitudes del estado, aumentando la amplitud del estado $\overline{x}_0$ y disminuyendo las otras.
Repitiendo este proceso (pasos 2 y 3) se va subiendo la amplitud del estado que se quiere encontrar y disminuyendo las otras. Sin embargo es cíclico: pasado cierto número de repeticiones, esa amplitud vuelve a decrecer. En la Sección 2.3.4 se calcula el número optimo de repeticiones para obtener la amplitud máxima. Cuando la amplitud es máxima, se realiza una medición, obteniendo el estado $\overline{x}_0$ con la máxima probabilidad. En la Sección 2.3.4 se muestra que la probabilidad de error tiene cota máxima en $1/2^n$.

Ejemplo

Sea una lista de $2^4 = 16$ elementos, de los que sólo uno, $\overline{x}_0$, verifica la propiedad $f(\overline{x}_0) = 1$. El algoritmo comienza por tomar el estado $|0\rangle^{\otimes 4}$ y aplicar $H^{\otimes 4}$ obteniendo,

$$\frac{1}{4}\sum_{\overline{x}\in\{0,1\}^4} |\overline{x}\rangle$$

Inicialmente todas las amplitudes son iguales a $1/4$. Se aplica el oráculo y se obtiene

$$\frac{1}{4}\sum_{\overline{x}\in\{0,1\}^4} (-1)^{f(\overline{x})}|\overline{x}\rangle$$

Luego se aplica la inversión sobre el promedio, y la nueva amplitud del estado $\overline{x}_0$ será

$$\frac{2^5 + 2^4 - 4}{2^4\sqrt{2^4}} = \frac{11}{16} = 0{,}6875$$

y para el resto de los $\overline{x}$ la amplitud será

$$\frac{2^5 - 2^4 - 4}{2^4\sqrt{2^4}} = \frac{3}{16} = 0{,}1875$$

Con las sucesivas repeticiones de la aplicación del oráculo y la inversión sobre el promedio, se obtienen las siguientes amplitudes:

Repetición	Amplitud de $\overline{x}_0$	Amplitud de $\overline{x} \neq \overline{x}_0$	Probabilidad de error
1	0.6875	0.1875	0.527
2	0.953125	0.078125	0.092
3	0.98046875	-0.05078125	0.039

A partir de la iteración 4 la probabilidad de error comienza a subir, por lo tanto el número óptimo de iteraciones es 3, con una probabilidad de error de 0,039.

2.3.4. Cálculo del número óptimo de iteraciones

Luego de k iteraciones $\overline{x}_0$ tendrá una amplitud b_k y el resto tendrán todos una amplitud m_k. Es decir, el estado será

$$b_k|\overline{x}_0\rangle + m_k \sum_{\substack{\overline{x}\in\{0,1\}^n \\ \overline{x}\neq\overline{x}_0}} |\overline{x}\rangle$$

En cada iteración se aplica el oráculo U, el cual cambia el signo de b_k, y luego G. Es posible definir recursivamente las amplitudes en la repetición k:

$$\begin{aligned} m_0 = b_0 &= \frac{1}{\sqrt{2^n}} \\ m_{k+1} &= 2A_k - m_k \\ b_{k+1} &= 2A_k + b_k \end{aligned} \qquad \text{donde} \qquad A_k = \frac{(2^n - 1)m_k - b_k}{2^n}$$

Las fórmulas cerradas para estas recursiones son

$$m_k = \frac{1}{\sqrt{2^n - 1}} \cos((2k+1)\gamma)$$
$$b_k = \text{sen}((2k+1)\gamma)$$

donde $\cos(\gamma) = \sqrt{\frac{2^n - 1}{2^n}}$ y $\text{sen}(\gamma) = \sqrt{\frac{1}{2^n}}$.
Para conseguir la mínima probabilidad de error, se debe minimizar $|m_k|$. Notar que $m_k = 0$ si y sólo si $(2k+1)\gamma = \frac{\pi}{2}$, es decir, si $k = \frac{\pi}{4\gamma} - \frac{1}{2}$.
Sin embargo, dado que k es el número de repeticiones, debe ser entero, por lo tanto, el número óptimo de iteraciones es

$$\tilde{k} = \left\lfloor \frac{\pi}{4\gamma} \right\rfloor$$

Para calcular una cota de la probabilidad de error, observar primero que que $|k - \tilde{k}| \leq \frac{1}{2}$, entonces

$$|\frac{\pi}{2} - (2\tilde{k}+1)\gamma| = |(2k+1)\gamma - (2\tilde{k}+1)\gamma| = |2\gamma(k - \tilde{k})| \leq \gamma$$

Con esto, podemos determinar que la probabilidad de error luego de $\tilde{k}$ iteraciones es

$$(2^n - 1)(m_k)^2 = cos^2((2\tilde{k}+1)\gamma) = sen^2(\frac{\pi}{2} - (2\tilde{k}+1)\gamma) \leq sen^2(\gamma) = \frac{1}{2^n}$$

En el ejemplo anterior

$$\tilde{k} = \left\lfloor \frac{\pi}{4asen(\sqrt{\frac{1}{16}})} \right\rfloor = 3$$

y la probabilidad de error es $0{,}039 \leq \frac{1}{2^4} = 0{,}0625$.

2.4. Aplicación criptográfica

2.4.1. One-time pad

Este es un método de criptografía clásica [Vernam, 1926] que consiste en compartir una secuencia de bits (clave) del largo del mensaje a transmitir y aplicar la operación reversible XOR para cifrar y descifrar. (Ver Figura 2.1). Las claves deben ser secretas y no deben ser reutilizadas.
Este método es 100 % seguro: un 0 en el mensaje encriptado puede significar un 0 en el mensaje original y un 0 en la clave, o un 1 en el mensaje y un 1 en la clave. Lo mismo sucede con un 1 en el mensaje encriptado. Es decir que adivinar la clave tiene la misma probabilidad que adivinar el mensaje original. La única debilidad de este método es la predistribución de claves, ya que el canal que se use para distribuirla podría ser vulnerado.
El método cuántico que se describe a continuación, QKD-BB84 (por *Quantum Key Distribution* de Bennett y Brassard [1984]), es justamente un método para la distribución segura de claves.

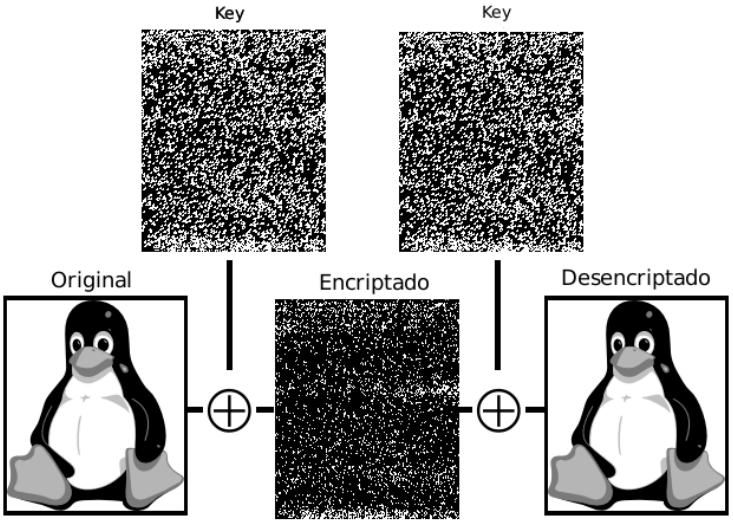


Figura 2.1: One-Time pad

2.4.2. Criptosistema Cuántico QKD-BB84

La idea es transmitir una clave binaria por un canal inseguro.
Para transmitir el bit 0, Alice (el emisor) puede elegir, al azar, la base $\{|0\rangle, |1\rangle\}$ (a la que llamaremos esquema $+$) y considerar $0 \equiv |0\rangle$, o la base $\{|-\rangle, |+\rangle\}$ (a la que llamaremos esquema $\times$) y considerar $0 \equiv |-\rangle$. Análogamente al bit 1 lo codificamos como $|1\rangle$ en el esquema $+$ o como $|+\rangle$ en el esquema $\times$.
Bob realizará una medición sobre el estado recibido eligiendo al azar entre el esquema $+$ y el esquema $\times$. Ver ejemplo en Figura 2.2. El paso final es intercambiar información (por un canal abierto) de los esquemas utilizados, y sólo conservar los bits producidos usando el mismo esquema.

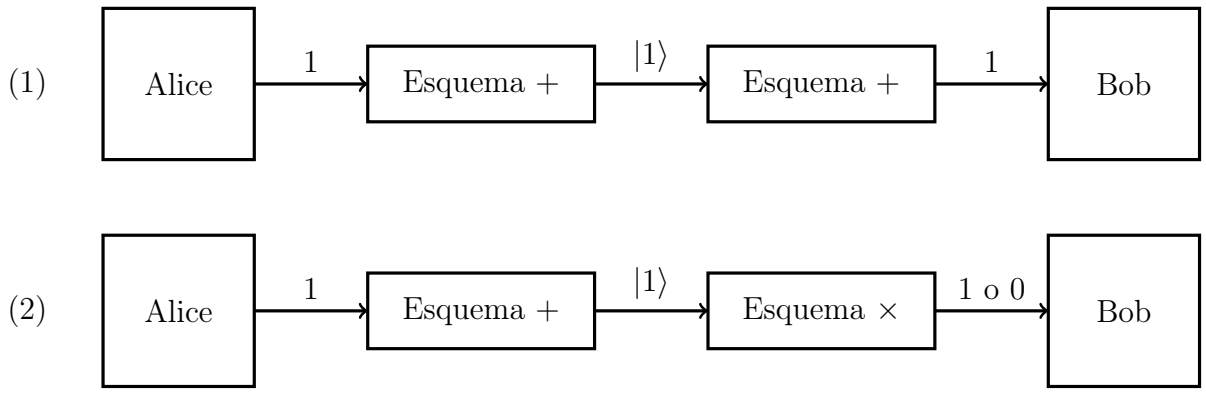


Figura 2.2: Ejemplo: (1) Alice transmite un 1 codificado mediante el esquema $+$ y Bob elije al azar el esquema $+$ obteniendo un 1 (2) si Bob elige el esquema $\times$ obtiene 0 ó 1 con probabilidad $1/2$.

El algoritmo paso a paso:

1. Alice comienza a transmitir una secuencia de 0 y 1, elegidos aleatoriamente, alternando los esquemas $+$ y $\times$ también aleatoriamente.

2. Bob recibe la secuencia y va alternando las mediciones entre los esquemas + y × aleatoriamente.

3. Alice le transmite a Bob la sucesión de esquemas empleada.

4. Bob le informa a Alice en qué casos utilizó el mismo esquema.

5. Usando solamente los bits de los esquemas idénticos a dos puntas, ambos han definido una sucesión aleatoria de bits que servirá como one-time pad de encriptación para transmisiones futuras por cualquier canal.

Esquemas de Alice	×	+	+	×	×	+	×	+
Valores de Alice	$\|-\rangle$	$\|0\rangle$	$\|0\rangle$	$\|+\rangle$	$\|-\rangle$	$\|0\rangle$	$\|-\rangle$	$\|1\rangle$
Esquemas de Bob	+	×	+	×	+	+	×	×
Valores de Bob	$\|0\rangle$	$\|+\rangle$	$\|0\rangle$	$\|+\rangle$	$\|1\rangle$	$\|0\rangle$	$\|-\rangle$	$\|-\rangle$
Coincidencias			√	√		√	√	
Clave			0	1		0	0	

6. Alice y Bob intercambian hashes de las claves (en bloques) para aceptarla o descartarla.

Inviolabilidad Este protocolo es, en teoría, inviolable. Supongamos que Cliff espía el canal de comunicación entre Alice y Bob e intenta recuperar la clave. Cliff está en la misma situación que Bob y no conoce cuál esquema es el correcto, + o ×. Por lo tanto elige al azar y se equivocará, en promedio, la mitad de las veces.
En el paso 5 Alice y Bob se ponen de acuerdo en cuáles valores tomar en cuenta (las coincidencias de la secuencia de esquemas). Esta información no le es útil a Cliff porque sólo en la mitad de las veces habrá usado el detector correcto, de manera que mal interpretará sus valores finales.
Además el QKD brinda el método para que Alice y Bob puedan detectar el potencial espionaje de Cliff:
Imaginemos que Alice envió un 0 con el esquema × (es decir, el qubit $|-\rangle$). Si Cliff usa el esquema +, colapsará el qubit a $|0\rangle$ o $|1\rangle$. Si Bob usa el esquema × y mide $|-\rangle$ coincide con lo enviado por Alice, pero si mide $|+\rangle$ Alice y Bob descubrirán esa discrepancia durante el intercambio de hashes, por lo tanto descartarán el bloque.

Capítulo 3

Introducción a la mecánica cuántica

3.1. Postulados de la mecánica cuántica

En el Capítulo 1 vimos los postulados de la mecánica cuántica sin mencionarlo, desde un punto de vista matemático formal. Revisitaremos los mismos postulados, nombrándolos como tales. Estos cuatro postulados definen el entorno matemático conocido como mecánica cuántica.

Postulado 1 (Espacio de estados)**.** Todo sistema físico cuántico aislado tiene asociado un espacio vectorial complejo con producto escalar conocido como el *espacio de estados* del sistema. El sistema se describe completamente por un *vector de estado*, el cual es un vector unidad en el espacio de estados.

Postulado 2 (Evolución)**.** La evolución de un sistema físico cuántico aislado se describe por una *transformación unitaria*. Es decir, el estado $|\psi\rangle$ del sistema en el tiempo t_1 se relaciona con el estado $|\psi'\rangle$ del sistema en el tiempo t_2 a través del operador unitario U, el cual sólo depende de los tiempos t_1 y t_2.

$$|\psi'\rangle = U|\psi\rangle$$

El postulado anterior se puede tomar con tiempo continuo, para lo cual hace falta una ecuación diferencial, y el postulado se transforma en el siguiente:

Postulado 2′. La evolución del estado de un sistema físico cuántico aislado es descripta por la *ecuación de Shcrödinger*,

$$i\hbar\frac{d|\psi\rangle}{dt} = H|\psi\rangle$$

En esta ecuación, $\hbar$ es una constante física conocida como *constante de Planck* cuyo valor debe ser determinado experimentalmente. El valor exacto no es importante, en la práctica es común absorber el valor $\hbar$ en H tomando $\hbar = 1$. El operador H no es la compuerta Hadamard vista anteriormente sino un operador hermítico fijo conocido como el *Hamiltoniano* del sistema.

Postulado 3 (Medición cuántica)**.** La medición cuántica se describe por una colección $\{M_m\}$ de *matrices de medición*. Dichas matrices actúan en el espacio de estados del sistema que se mide. El índice m refiere a los resultados posibles de la medición. Si el estado del sistema es $|\psi\rangle$, inmediatamente antes de la medición, entonces la probabilidad de que el resultado m ocurra viene dado por

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle$$

y el estado del sistema luego de la medición es

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}$$

Las matrices satisfacen la ecuación de completitud,

$$\sum_m M_m^\dagger M_m = I$$

La ecuación de completitud expresa el hecho de que las probabilidades suman uno:

$$1 = \sum_m p(m) = \sum_m \langle\psi|M_m^\dagger M_m|\psi\rangle = \langle\psi| \left(\sum_m M_m^\dagger M_m\right) |\psi\rangle$$

Postulado 4 (Sistema compuesto)**.** El espacio de estados de un sistema físico compuesto es el producto tensorial de los espacios de estados de los componentes. Más aún, si tenemos sistemas enumerados de 1 a n, donde el sistema i está en el estado $|\psi_i\rangle$, el estado conjunto del sistema total es

$$|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$$

3.1.1. Medición proyectiva

Un caso particular del postulado 3 es el conocido como *medición proyectiva*. De hecho, la medición general es equivalente a las mediciones proyectivas más operaciones unitarias. Por lo tanto, en general, usaremos sólo mediciones proyectivas.

3.1.1.1. Preliminares

Definición 3.1 (Autovectores y autovalores)**.** Un *autovector* de un operador lineal A en un espacio vectorial dado es un vector no-nulo $|v\rangle$ tal que $A|v\rangle = v|v\rangle$, donde v es un número complejo conocido como *autovalor* de A correspondiente a $|v\rangle$.

Observación. En la definición anterior, notar que $v \neq |v\rangle$. De hecho, $v \in \mathbb{C}$ y $|v\rangle \in \mathbb{C}^2$. La "$v$" que aparece en $|v\rangle$ es simplemente una etiqueta, un nombre, para el vector.

Ejemplo 3.2. Consideremos la matriz de Pauli iXZ

$$\begin{aligned} iXZ &= i(|0\rangle\langle 1| + |1\rangle\langle 0|)(|0\rangle\langle 0| - |1\rangle\langle 1|) \\ &= i(|0\rangle\langle 1|0\rangle\langle 0| - |0\rangle\langle 1|1\rangle\langle 1| + |1\rangle\langle 0|0\rangle\langle 0| - |1\rangle\langle 0|1\rangle\langle 1|) \\ &= i(|1\rangle\langle 0| - |0\rangle\langle 1|) \end{aligned}$$

O, en su notación matricial, $iXZ = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$

Queremos buscar un vector $|v\rangle = \alpha|0\rangle + \beta|1\rangle$ tal que $iXZ|v\rangle = v|v\rangle$, para algún v, y con $\||v\rangle\| = 1$. Es decir:

$$i(|1\rangle\langle 0| - |0\rangle\langle 1|)(\alpha|0\rangle + \beta|1\rangle) = i(\alpha|1\rangle - \beta|0\rangle) = -\beta i|0\rangle + \alpha i|1\rangle$$

debe ser igual a $v(\alpha|0\rangle + \beta|1\rangle)$, con $|\alpha|^2 + |\beta|^2 = 1$.
Por lo tanto,

$$\begin{cases} |\alpha|^2 + |\beta|^2 = 1 \\ v\alpha = -\beta i \\ v\beta = \alpha i \end{cases} \Rightarrow \begin{cases} \left|\dfrac{\beta i}{v}\right|^2 + |\beta|^2 = 1 \\ \alpha = -\dfrac{\beta i}{v} \\ v\beta = \left(-\dfrac{\beta i}{v}\right) i \end{cases} \Rightarrow \begin{cases} |\beta| = \sqrt{\dfrac{|v|}{|v|+1}} \\ \alpha = -\beta i \\ v^2 = 1 \end{cases}$$

$$\Rightarrow \begin{cases} |\beta| = \dfrac{1}{\sqrt{2}} \\ \alpha = -\beta i \\ v = 1 \end{cases} \text{ o } \begin{cases} |\beta| = \dfrac{1}{\sqrt{2}} \\ \alpha = -\beta i \\ v = -1 \end{cases}$$

Tomando, por ejemplo, $v = 1$ y $\beta = 1/\sqrt{2}$ tenemos $\alpha = -i/\sqrt{2}$, y por lo tanto 1 es un autovalor de iXZ con autovector $1/\sqrt{2}(|1\rangle - i|0\rangle)$.

Definición 3.3 (Función característica)**.** La *función característica* de un operador lineal A es $c(x) = \det|A - xI|$.

Teorema 3.4. Las soluciones a la ecuación $c(x) = 0$ son los autovalores del operador A. □

Ejemplo 3.5. En el ejemplo anterior, podemos ver que

$$\begin{aligned} c(x) &= \det|iXZ - xI| \\ &= \det|i(|1\rangle\langle 0| - |0\rangle\langle 1|) - x(|0\rangle\langle 0| + |1\rangle\langle 1|)| \\ &= \det|-x|0\rangle\langle 0| - i|0\rangle\langle 1| + i|1\rangle\langle 0| - x|1\rangle\langle 1|| \\ &= (-x)^2 - (-i^2) \\ &= x^2 - 1 \end{aligned}$$

Y tenemos $c(x) = 0 \Rightarrow x^2 - 1 = 0 \Rightarrow x = \pm 1$.

Definición 3.6 (Autoespacio)**.** El *autoespacio* correspondiente un autovalor v de un operador linear A es el conjunto de vectores que tienen a v como autovalor.

Ejemplo 3.7. Siguiendo con el ejemplo anterior, el autoespacio correspondiente al autovalor 1 del operador iXZ es $\{\beta|1\rangle - \beta i|0\rangle \mid \beta \in \mathbb{C}\}$.

Teorema 3.8. El autoespacio de un autovalor v de un operador lineal A en un espacio vectorial V es un subespacio vectorial de V. □

3.1.1.2. Medición proyectiva

Definición 3.9. Una medición proyectiva es descripta por un *observable*, M, el cual es un operador hermítico en el espacio de estados del sistema que es objeto de la observación. El observable tiene descomposición espectral (es decir, factorización a forma canónica) dada por:

$$M = \sum_m m P_m$$

donde P_m es el proyector al autoespacio de M con autovalor m. Los posibles resultados de la medición corresponden con los autovalores m del observable. Luego de medir $|\psi\rangle$, la probabilidad de obtener el resultado m viene dada por

$$p(m) = \langle\psi|P_m|\psi\rangle$$

Al obtener el resultado m, el estado del sistema inmediatamente luego de la medición es

$$\frac{P_m|\psi\rangle}{\sqrt{p(m)}}$$

Observación. La medición proyectiva se puede ver como un caso particular del Postulado 3. Si a las matrices que forman el operador medición del Postulado 3 le agregamos la condición que los M_m son hermíticos y ortogonales, es decir $M_m M_{m'} = \delta_{m,m'} M_m$, entonces el Postulado 3 reduce a las mediciones proyectivas que acabamos de definir.

Ejemplo 3.10. Consideremos la medición del observable Z.

$$c(x) = \det|Z - xI| = \det|(1-x)|0\rangle\langle 0| - (1+x)|1\rangle\langle 1|| = (1-x)(-1-x) = -1 + x^2$$

Por lo tanto, las soluciones a $c(x) = 0$ son 1 y -1, y esos son los autovalores de Z. Dichos autovalores corresponden a los autovectores $|0\rangle$ y $|1\rangle$ respectivamente.
Los proyectores P_0 y P_1 sobre los autoespacios $\{\alpha|0\rangle \mid \alpha \in \mathbb{C}\}$ y $\{\beta|1\rangle \mid \beta \in \mathbb{C}\}$ respectivamente son $P_0 = |0\rangle\langle 0|$ y $P_1 = |1\rangle\langle 1|$. Notar que

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1|$$

Entonces, la medición de Z sobre el estado $|-\rangle = {}^{(|0\rangle+|1\rangle)}/\sqrt{2}$ da como resultado 1 con probabilidad $\langle -|0\rangle\langle 0|-\rangle = 1/2$. Similarmente, se obtiene resultado -1 con probabilidad $1/2$.

3.1.2. Fase

Consideremos por ejemplo el estado $e^{i\theta}|\psi\rangle$[①], donde $|\psi\rangle$ es un vector de estado, y θ es un número real. Decimos que el estado $e^{i\theta}|\psi\rangle$ es igual a $|\psi\rangle$, excepto por la fase global $e^{i\theta}$. La medición sobre ambos estados es la misma: Supongamos que M_m es una matriz de un operador de medición. Entonces las probabilidades de aplicar esa matriz vienen dadas por $\langle\psi|M_m^\dagger M_m|\psi\rangle$ y por $\langle\psi|e^{-i\theta}M_m^\dagger M_m e^{i\theta}|\psi\rangle = \langle\psi|M_m^\dagger M_m|\psi\rangle$[②]. Por lo tanto, desde un punto de vista observacional, ambos estados son idénticos.
Por esta razón solemos ignorar las fases globales ya que son irrelevantes a las propiedades observacionales de sistemas físicos.

[①] $e^{i\theta} = \cos\theta + i\sin\theta$ ($ae^{i\theta}$ es la llamada *notación exponencial* de un número complejo, donde a su módulo y θ su argumento).

[②] $e^{-i\theta}e^{i\theta} = e^{-i\theta+i\theta} = e^0 = 1$.

3.2. Operador densidad

Hasta ahora hemos visto la mecánica cuántica en términos de vectores de estados. Una formulación alternativa es usando el operador densidad (o matriz densidad). Esta presentación es equivalente matemáticamente, pero provee un lenguaje más conveniente para razonar en algunos escenarios comunes que se encuentran en la mecánica cuántica.

3.2.1. Preliminares

Definición 3.11 (Traza)**.** La *traza* de una matriz es la suma de sus elementos diagonales. Así, si $A = \sum_i \sum_j \alpha_{ij} |u_i\rangle\langle u_j|$, la traza se define por

$$\mathsf{tr}(A) = \sum_i \alpha_{ii}$$

Teorema 3.12. Sea $A = |\psi\rangle\langle\varphi|$. Entonces, $\mathsf{tr}(A) = \langle\varphi|\psi\rangle$.

Demostración. Sean $|\psi\rangle = \sum_i a_i|u_i\rangle$ y $|\varphi\rangle = \sum_j b_j|u_j\rangle$. Entonces,

$$\mathsf{tr}(A) = \mathsf{tr}(|\psi\rangle\langle\varphi|) = \mathsf{tr}(\sum_i a_i|u_i\rangle \sum_j b_j^*\langle u_j|) \ = \mathsf{tr}(\sum_{ij} a_i b_j^* |u_i\rangle\langle u_j|) = \sum_i a_i b_i^* = \langle\varphi|\psi\rangle$$

□

Ejemplos 3.13.

1. Sea $A = |0\rangle\langle -|$. Entonces, $A = {}^1\!/\!_{\sqrt{2}}(|0\rangle\langle 0| - |0\rangle\langle 1|)$ y $\mathsf{tr}(A) = {}^1\!/\!_{\sqrt{2}}$.
   Por otro lado, siguiendo el teorema, $\mathsf{tr}(A) = \langle 0|-\rangle = {}^1\!/\!_{\sqrt{2}}(\langle 0|0\rangle - \langle 0|1\rangle) = {}^1\!/\!_{\sqrt{2}}$.
2. Sea $A = |+\rangle\langle -| = {}^1\!/\!_2(|0\rangle\langle 0| - |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|)$.
   Entonces, $\mathsf{tr}(A) = {}^1\!/\!_2 - {}^1\!/\!_2 = 0 = \langle +|-\rangle$.

El siguiente corollario es muy útil para evaluar la traza de un operador.

Corolario 3.14. Sea $|\psi\rangle$ un vector normalizado y A un operador cuántico. Entonces

$$\mathsf{tr}(A|\psi\rangle\langle\psi|) = \langle\psi|A|\psi\rangle$$

Demostración. Ejercicio. □

Ejemplo 3.15. $\mathsf{tr}(X|0\rangle\langle 0|) = \langle 0|X|0\rangle = \langle 0|0\rangle\langle 1|0\rangle + \langle 0|1\rangle\langle 0|0\rangle = 0$.

Propiedades (de la traza de una matriz)**.** Sean A y B matrices de la misma dimensión, U un operador unitario y $\lambda \in \mathbb{C}$. Entonces

1. $\mathsf{tr}(AB) = \mathsf{tr}(BA)$
2. $\mathsf{tr}(A + B) = \mathsf{tr}(A) + \mathsf{tr}(B)$
3. $\mathsf{tr}(\lambda A) = \lambda\mathsf{tr}(A)$
4. $\mathsf{tr}(UAU^\dagger) = \mathsf{tr}(A)$

Demostración. Sólo mostramos la propiedad 4, las otras se dejan como ejercicio. De la propiedad 1 se tiene $\mathsf{tr}(UAU^\dagger) = \mathsf{tr}(U^\dagger UA)$, y como U es unitaria, $\mathsf{tr}(U^\dagger UA) = \mathsf{tr}(A)$. □

3.2.2. Conjuntos de estados cuánticos

El operador densidad provee una manera conveniente de describir un sistema cuántico en el cual el estado no se conoce del todo.

Definición 3.16 (Operador o matriz densidad)**.** Supongamos que un sistema cuántico está en uno de un número de estados $|\psi_i\rangle$, donde la probabilidad de que el estado sea $|\psi_i\rangle$ viene dada por p_i.
Decimos que el conjunto $\{p_i, |\psi_i\rangle\}$ es el *conjunto de estados puros*. El *operador densidad* o *matriz densidad* para este estado viene dado por la ecuación

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

Ejemplo 3.17. El operador densidad del conjunto de estados puros $\{(1/4, |+\rangle); (3/4, |1\rangle)\}$ tiene operador densidad

$$\rho = 1/4|+\rangle\langle+| + 3/4|1\rangle\langle 1| = 1/8|0\rangle\langle 0| + 1/8|0\rangle\langle 1| + 1/8|1\rangle\langle 0| + 7/8|1\rangle\langle 1|$$

Es decir

$$\rho = \begin{pmatrix} 1/8 & 1/8 \\ 1/8 & 7/8 \end{pmatrix}$$

Observación. Todos los postulados de la mecánica cuántica se pueden reformular en términos del operador densidad, y haremos eso más adelante en esta sección.

Evolución Supongamos que la evolución de un sistema cuántico cerrado se describe por el operador unitario U. Si el sistema estaba inicialmente en el estado $|\psi_i\rangle$ con probabilidad p_i, entonces, luego de la evolución el sistema estará en estado $U|\psi_i\rangle$ con probabilidad p_i. Por lo tanto, la evolución del operador densidad se describe por

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i p_i U|\psi_i\rangle\langle\psi_i|U^\dagger = U\rho U^\dagger$$

Ejemplo 3.18. Siguiendo con el Ejemplo 3.17, tomemos $U = H$, entonces el conjunto de estados puros original $\{(1/4, |+\rangle); (3/4, |1\rangle)\}$ evolucionará a $\{(1/4, |0\rangle); 3/4, |-\rangle)\}$ y su matriz densidad puede calcularse de dos maneras equivalentes:

1. A partir del conjunto de estados puros:

$$\rho' = 1/4|0\rangle\langle 0| + 3/4|-\rangle\langle -| = 5/8|0\rangle\langle 0| - 3/8|0\rangle\langle 1| - 3/8|1\rangle\langle 0| + 3/8|1\rangle\langle 1| = \begin{pmatrix} 5/8 & -3/8 \\ -3/8 & 3/8 \end{pmatrix}$$

2. O utilizando la igualdad dada más arriba: $\rho' = H\rho H^\dagger = H\rho H$.

Medición Supongamos que realizamos una medición descripta por las matrices M_m. Si el estado inicial era $|\psi_i\rangle$, entonces la probabilidad de obtener el resultado m es

$$p(m|i) = \langle\psi_i|M_m^\dagger M_m|\psi_i\rangle \overset{\text{Cor.3.14}}{=} \mathsf{tr}(M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|)$$

Usando la ley de probabilidades totales, la probabilidad de obtener el resultado m es

$$\begin{aligned} p(m) &= \sum_i p(m|i)p_i \\ &= \sum_i p_i \mathsf{tr}(M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|) \\ &= \mathsf{tr}(\sum_i p_i M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|) \\ &= \mathsf{tr}(M_m^\dagger M_m \sum_i p_i|\psi_i\rangle\langle\psi_i|) \\ &= \mathsf{tr}(M_m^\dagger M_m \rho) \end{aligned}$$

Si el estado inicial era $|\psi_i\rangle$, el estado luego de obtener el resultado m será

$$|\psi_i^m\rangle = \frac{M_m|\psi_i\rangle}{\sqrt{\langle\psi_i|M_m^\dagger M_m|\psi_i\rangle}}$$

Por lo tanto, luego de una medición que de resultado m tendremos el conjunto de estados $|\psi_i^m\rangle$, con probabilidades $p(i|m)$ respectivamente. Por lo tanto, el operador densidad ρ_m correspondiente es

$$\rho_m = \sum_i p(i|m)|\psi_i^m\rangle\langle\psi_i^m| = \sum_i p(i|m)\frac{M_m|\psi_i\rangle\langle\psi_i|M_m^\dagger}{\langle\psi_i|M_m^\dagger M_m|\psi_i\rangle} \tag{3.1}$$

Pero, usando teoría de probabilidad condicional,

$$p(i|m) = \frac{p(m\cap i)}{p(m)} = \frac{p(m|i)p_i}{p(m)} = p_i\frac{\mathsf{tr}(M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|)}{\mathsf{tr}(M_m^\dagger M_m\rho)} = p_i\frac{\langle\psi_i|M_m^\dagger M_m|\psi_i\rangle}{\mathsf{tr}(M_m^\dagger M_m\rho)}$$

Substituyendo en (3.1), obtenemos

$$\rho_m = \sum_i p_i\frac{M_m|\psi_i\rangle\langle\psi_i|M_m^\dagger}{\mathsf{tr}(M_m^\dagger M_m\rho)} = \frac{M_m\rho M_m^\dagger}{\mathsf{tr}(M_m^\dagger M_m\rho)}$$

Ejemplo 3.19. Volviendo al conjunto de estados del Ejemplo 3.17, tenemos

$$\rho = {}^1\!/_8|0\rangle\langle 0| + {}^1\!/_8|0\rangle\langle 1| + {}^1\!/_8|1\rangle\langle 0| + {}^7\!/_8|1\rangle\langle 1|$$

que corresponde a la matriz densidad del conjunto de estados $\{(\frac{1}{4}, |+\rangle); (\frac{3}{4}, |1\rangle)\}$. Utilizaremos la medición proyectiva $\{P_0, P_1\}$ con $P_0 = |0\rangle\langle 0|$ y $P_1 = |1\rangle\langle 1|$.

Entonces, la probabilidad de medir 0 viene dada por

$$\begin{aligned}\mathsf{tr}(P_0^\dagger P_0 \rho) &= |0\rangle\langle 0|(1/8|0\rangle\langle 0| + 1/8|0\rangle\langle 1| + 1/8|1\rangle\langle 0| + 7/8|1\rangle\langle 1|) \\ &= \mathsf{tr}(1/8|0\rangle\langle 0| + 1/8|0\rangle\langle 1|) \\ &= 1/8\,\mathsf{tr}(|0\rangle\langle 0|) + 1/8\,\mathsf{tr}(|0\rangle\langle 0|) \\ &= 1/8\end{aligned}$$

Similarmente, la probabilidad de medir 1 por

$$\mathsf{tr}(|1\rangle\langle 1|\rho) = 1/8\,\mathsf{tr}(|1\rangle\langle 0|) + 7/8\,\mathsf{tr}(|1\rangle\langle 1|) = 7/8$$

Podemos ver que el conjunto de estados está en el estado $|1\rangle$ con probabilidad $3/4$. Si ese es efectivamente el estado inicial, la probabilidad de medir 1 sería 1. Por otro lado, en el estado $|+\rangle$, la probabilidad de medir 1 es $1/2$. De ahí que la probabilidad de medir 1 es

$$3/4 \times 1 + 1/4 \times 1/2 = 7/8$$

tal y como dedujimos con la traza.
Luego de realizar la medición, si se midió 1, el estado del sistema podrá ser descripto por el operador siguiente:

$$\rho_1 = \frac{P_1 \rho P_1^\dagger}{7/8} = \frac{7/8|1\rangle\langle 1|}{7/8} = |1\rangle\langle 1|$$

Efectivamente, si se midió 1 y el estado inicial era $|+\rangle$, el estado final será $|1\rangle$, pero lo mismo pasa si el estado inicial era $|1\rangle$, por lo que la matriz densidad es la matriz densidad del conjunto de estados $\{1, |1\rangle\}$.

Definición 3.20. Un sistema cuántico donde el estado $|\psi\rangle$ se conoce exactamente se dice que está en un *estado puro*. En este caso, el operador densidad es simplemente $\rho = |\psi\rangle\langle\psi|$. Si no es un estado puro, ρ está en un *estado mixto* (o mezcla), o que es una mezcla de diferentes estados puros.

Teorema 3.21. Para todo operador densidad ρ se tiene $\mathsf{tr}(\rho^2) \leq 1$.
Más aún, la igualdad se cumple si y sólo si ρ está en un estado puro

Demostración. Ejercicio. □

Teorema 3.22. Un estado cuántico que está en estado ρ_i con probabilidad p_i, puede ser descripto por la matriz densidad $\sum_i p_i \rho_i$.

Demostración. Supongamos que ρ_i viene de un conjunto $\{p_{ij}, |\psi_{ij}\rangle\}$ de estados puros (con i fijo). Por lo tanto, la probabilidad de estar en el estado $|\psi_{ij}\rangle$ viene dada por $p_i p_{ij}$. Es decir que la matriz densidad es $\rho = \sum_i \sum_j p_i p_{ij} |\psi_{ij}\rangle\langle\psi_{ij}| = \sum_i p_i \rho_i$. □

3.2.3. Propiedades generales del operador densidad

Definición 3.23 (Operador positivo)**.** Un operador A se dice *positivo* si para todo vector $|\psi\rangle$, $\langle\psi|A|\psi\rangle \geq 0$. Si $\langle\psi|A|\psi\rangle > 0$ para todo $|\psi\rangle \neq 0$, decimos que A es *definido positivo*.

Teorema 3.24. Si A es un operador positivo, entonces existe una descomposición $A = \sum_j \lambda_j |j\rangle\langle j|$ donde los vectores $|j\rangle$ son ortonormales y $\lambda_j \in \mathbb{R}_0^+$ son autovalores de A. □

Teorema 3.25 (Caracterización de operadores densidad)**.** Un operador ρ es el operador densidad de un conjunto $\{p_i, |\psi_i\rangle\}$ si y sólo si satisface las siguientes condiciones:

1. $\mathsf{tr}(\rho) = 1$

2. ρ es un operador positivo

Demostración.

⇒) Sea $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ un operador densidad. Entonces,

1. $\mathsf{tr}(\rho) = \sum_i p_i \mathsf{tr}(|\psi_i\rangle\langle\psi_i|) = \sum_i p_i = 1$.
2. Sea $|\varphi\rangle$ un vector arbitrario en el espacio de estados. Entonces,

$$\langle\varphi|\rho|\varphi\rangle = \langle\varphi| \left(\sum_i p_i |\psi_i\rangle\langle\psi_i| \right) |\varphi\rangle = \sum_i p_i \langle\varphi|\psi_i\rangle\langle\psi_i|\varphi\rangle = \sum_i p_i |\langle\varphi|\psi_i\rangle|^2 \geq 0$$

⇐) Sea ρ cualquier operador positivo con traza igual a 1. Como ρ es positivo, usando el Teorema 3.24 tenemos $\rho = \sum_j \lambda_j |j\rangle\langle j|$, donde los vectores $|j\rangle$ son ortogonales y $\lambda_j \in \mathbb{R}_0^+$ son autovalores de ρ. Por la condición de traza 1, tenemos $\sum_j \lambda_j = 1$. Por lo tanto, un sistema en el estado $|j\rangle$ con probabilidad λ_j tendrá un operador de densidad ρ. □

El Teorema 3.25 nos permite reformular el Postulado 1 para no depender de vectores, y podemos entonces escribir todos los postulados en términos del operador densidad.

> **Postulado 1.** Todo sistema físico cuántico aislado tiene asociado un espacio vectorial complejo con producto escalar conocido como el *espacio de estados* del sistema. El sistema se describe completamente por su *operador densidad*, el cual es un operador positivo ρ con traza 1, que actúa en el espacio de estados del sistema. Si un sistema cuántico está en estado ρ_i con probabilidad p_i, entonces el operador densidad del sistema es $\sum_i p_i \rho_i$.

> **Postulado 2.** La evolución de un sistema físico cuántico aislado se describe por una *transformación unitaria*. Es decir, el estado ρ del sistema en el tiempo t_1 se relaciona con el estado ρ' del sistema en el tiempo t_2 a través del operador unitario U, el cual sólo depende de los tiempos t_1 y t_2.
>
> $$\rho' = U\rho U^\dagger$$

Postulado 3. La medición cuántica se describe por una colección $\{M_m\}$ de *matrices de medición*. Dichas matrices actúan en el espacio de estados del sistema que se mide. El índice m refiere a los resultados posibles de la medición. Si el estado del sistema es ρ, inmediatamente antes de la medición, entonces la probabilidad de que el resultado m ocurra viene dado por

$$p(m) = \mathsf{tr}(M_m^\dagger M_m \rho)$$

y el estado del sistema luego de la medición es

$$\frac{M_m \rho M_m^\dagger}{\mathsf{tr}(M_m^\dagger M_m \rho)}$$

Las matrices satisfacen la ecuación de completitud,

$$\sum_m M_m^\dagger M_m = I$$

Postulado 4. El espacio de estados de un sistema físico compuesto es el producto tensorial de los espacios de estados de los componentes. Más aún, si tenemos sistemas enumerados de 1 a n, donde el sistema i está en el estado ρ_i, el estado conjunto del sistema total es $\rho_1 \otimes \rho_2 \otimes \cdots \otimes \rho_n$.

3.2.4. El operador densidad reducido

Uno de los usos más interesantes del operador densidad es para describir subsistemas de un sistema cuántico compuesto. Tal descripción viene dada por el *operador densidad reducido*.

Definición 3.26. Sean A y B dos sistemas físicos tales que su estado es descripto por el operador densidad ρ^{AB}. El operador densidad reducido para A se define por

$$\rho^A = \mathsf{tr}_B(\rho^{AB})$$

donde tr_B es la *traza parcial sobre el sistema* B, y es un operador lineal definido por

$$\mathsf{tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = |a_1\rangle\langle a_2|\mathsf{tr}(|b_1\rangle\langle b_2|) = \langle b_2|b_1\rangle|a_1\rangle\langle a_2|$$

para todo $|a_1\rangle, |a_2\rangle$ en el espacio de estados de A y $|b_1\rangle$, $|b_2\rangle$ en el espacio de estados de B.

Ejemplos 3.27. Supongamos que tenemos un sistema cuántico en el estado $\rho^{AB} = \rho \otimes \sigma$, donde ρ es el operador densidad del sistema A y σ el del sistema B. Entonces,

$$\rho^A = \mathsf{tr}_B(\rho \otimes \sigma) = \rho\mathsf{tr}(\sigma) = \rho$$

Similarmente, $\rho^B = \sigma$.

Un ejemplo menos trivial es el estado de Bell $\beta_{00} = 1/\sqrt{2}(|00\rangle + |11\rangle)$, que tiene el siguiente operador densidad

$$\rho = \left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right)\left(\frac{\langle 00| + \langle 11|}{\sqrt{2}}\right) = \frac{|00\rangle\langle 00| + |11\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 11|}{2}$$

Haciendo la traza sobre el segundo qubit obtenemos el operador densidad del primer qubit:

$$\begin{aligned}
\rho^1 &= \mathsf{tr}_2(\rho) \\
&= \frac{\mathsf{tr}_2(|00\rangle\langle 00|) + \mathsf{tr}_2(|11\rangle\langle 00|) + \mathsf{tr}_2(|00\rangle\langle 11|) + \mathsf{tr}_2(|11\rangle\langle 11|)}{2} \\
&= \frac{\mathsf{tr}_2(|0\rangle\langle 0| \otimes |0\rangle\langle 0|) + \mathsf{tr}_2(|1\rangle\langle 0| \otimes |1\rangle\langle 0|) + \mathsf{tr}_2(|0\rangle\langle 1| \otimes |0\rangle\langle 1|) + \mathsf{tr}_2(|1\rangle\langle 1| \otimes |1\rangle\langle 1|)}{2} \\
&= \frac{\langle 0|0\rangle|0\rangle\langle 0| + \langle 0|1\rangle|1\rangle\langle 0| + \langle 1|0\rangle|0\rangle\langle 1| + \langle 1|1\rangle|1\rangle\langle 1|}{2} \\
&= \frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2} \\
&= \frac{I}{2}
\end{aligned}$$

Notar que este es un estado mixto, ya que $\mathsf{tr}((I/2)^2) = 1/2 < 1$. Es decir que al estar enredados, por más que el estado de dos qubits sea un estado puro, el primer qubit sólo está en un estado mixto: es decir, un estado que no conocemos completamente.

3.2.4.1. Teleportación cuántica y el operador densidad reducido

Podemos usar el operador densidad reducido para analizar el algoritmo de teleportación. Cuando presentamos el algoritmo de teleportación (Sección 1.6.2) dijimos que no contradice la teoría de la relatividad (que entre otras cosas determina que nada puede viajar a mayor velocidad que la luz, ni siquiera la información), ya que no hay trasmisión de información hasta que Alice no le envía (usando un canal clásico) el resultado de la medición a Bob. Podemos hacer esta afirmación de manera más rigurosa utilizando el operador densidad reducido.
Antes de que Alice haga la medición, el estado del sistema es

$$\frac{1}{2}\left(|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)\right)$$

por lo que al medir los dos primeros qubits, se obtendrá

$$\begin{array}{ll}
|00\rangle(\alpha|0\rangle + \beta|1\rangle) & \text{con probabilidad } 1/4 \\
|01\rangle(\alpha|1\rangle + \beta|0\rangle) & \text{con probabilidad } 1/4 \\
|10\rangle(\alpha|0\rangle - \beta|1\rangle) & \text{con probabilidad } 1/4 \\
|11\rangle(\alpha|1\rangle - \beta|0\rangle) & \text{con probabilidad } 1/4
\end{array}$$

Por lo tanto, el operador densidad del sistema es

$$\begin{aligned}\rho = \frac{1}{4}&(|00\rangle\langle 00|(\alpha|0\rangle + \beta|1\rangle)(\alpha^*\langle 0| + \beta^*\langle 1|)\\ &+|01\rangle\langle 01|(\alpha|1\rangle + \beta|0\rangle)(\alpha^*\langle 1| + \beta^*\langle 0|)\\ &+|10\rangle\langle 10|(\alpha|0\rangle - \beta|1\rangle)(\alpha^*\langle 0| - \beta^*\langle 1|)\\ &+|11\rangle\langle 11|(\alpha|1\rangle - \beta|0\rangle)(\alpha^*\langle 1| - \beta^*\langle 0|))\end{aligned}$$

Por lo tanto, si hacemos la traza parcial sobre el sistema de Alice, obtenemos que operador densidad del sistema de Bob es

$$\begin{aligned}\rho^B &= \frac{1}{4}((\alpha|0\rangle + \beta|1\rangle)(\alpha^*\langle 0| + \beta^*\langle 1|) + (\alpha|1\rangle + \beta|0\rangle)(\alpha^*\langle 1| + \beta^*\langle 0|)\\ &\quad + (\alpha|0\rangle - \beta|1\rangle)(\alpha^*\langle 0| - \beta^*\langle 1|) + (\alpha|1\rangle - \beta|0\rangle)(\alpha^*\langle 1| - \beta^*\langle 0|))\\ &= \frac{2(|\alpha|^2 + |\beta|^2)|0\rangle\langle 0| + 2(|\alpha|^2 + |\beta|^2)|1\rangle\langle 1|}{4}\\ &= \frac{|0\rangle\langle 0| + |1\rangle\langle 0|}{2}\\ &= \frac{I}{2}\end{aligned}$$

Por lo tanto, el estado de Bob *después* de que Alice hizo la medición, pero *antes* de que Bob obtuvo el resultado de esa medición es $I/2$. Este estado no depende del estado $|\psi\rangle$ que se transmitió, y por lo tanto, cualquier medición que haga Bob no contendrá información sobre $|\psi\rangle$, lo que previene que Alice use la teleportación para enviar información a mayor velocidad que la luz.

Capítulo 4

Introducción al lambda cálculo y a la teoría de tipos

4.1. PCF no tipado

En lugar de comenzar con lambda cálculo, veremos PCF (*Programming language for computable functions*[1]). PCF se puede ver como una extensión al lambda cálculo con algunas construcciones prácticas como números naturales y el operador de punto fijo. Dichas construcciones no son necesarias, ya que es posible codificar todas ellas en lambda cálculo, sin embargo, harán el razonamiento más sencillo.
Gran parte del material de este capítulo ha sido tomado del libro de Dowek y Lévy [2011].

4.1.1. Primeras definiciones

- Dos construcciones de base:
  - Construcción explícita de función (sin nombre): $\lambda x.t$.
  - Aplicación de una función a un argumento: tu.
- Una constante para cada número natural.
- Operaciones: $+$, $-$, $\times$, $/$[2].
- Test a cero: $\mathsf{isZ}(t)?u:v$ devuelve u si t es 0 y v si t es > 0.

Observaciones.

- Todo es función: Una función tomando un argumento constante u otra función, es lo mismo. Por ejemplo, la función que toma una función y la compone consigo misma, es

$$\lambda f.\lambda x.f(fx)$$

[1] PCF es un lenguaje de programación para funciones computables, basado en la lógica de Scott para funciones computables LCF [Plotkin, 1977].

[2] Dado que sólo usamos naturales, consideraremos que $n - m = 0$ si $m > n$, y / representa la división euclidiana (con resto), donde la división por 0 simplemente no está definida.

- No existen funciones que tomen varios argumentos. Por ejemplo, $f(x, y) = x^2 + y^2$, en PCF se escribe

$$\underbrace{\lambda x.\lambda y.x \times x + y \times y}_{F}$$

  $F3$ es una función que espera un argumento para elevarlo al cuadrado y sumarle 9.

- La aplicación asocia a la izquierda: $F34 = (F3)4$.

  El punto (.) de la construcción λ y los dos puntos (:) de la construcción $\mathsf{isZ}()?:$ son equivalentes a paréntesis de apertura que cierran lo más a la derecha posible.

  Ejemplo: $\lambda x.(\mathsf{isZ}(x)?t:rs)u$ es $\lambda x.((\mathsf{isZ}(x)?t:(rs))u)$.

Recursión

¿Cómo escribiríamos la función factorial en PCF?

$$\lambda n.\mathsf{isZ}(n)?1:n \times (\underbrace{Fact}_{?}(n-1))$$

En PCF hay un símbolo (palabra clave) "μ" que liga una variable en su argumento tal que $\mu f.G$ es el *punto fijo*[3] de $\lambda f.G$.

$$\mu f.G = (\lambda f.G)(\mu f.G)$$

Por lo tanto,

$$Fact = \mu f.\underbrace{\lambda n.\mathsf{isZ}(n)?1:n \times (f(n-1))}_{G}$$

Ejemplo 4.1.

$$\begin{aligned}
Fact\ 2 &= (\mu f.G)2 \\
&= ((\lambda f.G)Fact)2 \\
&= (\lambda n.\mathsf{isZ}(n)?1:n \times Fact(n-1))2 \\
&= \mathsf{isZ}(2)?1:2 \times Fact\ 1 \\
&= 2 \times Fact\ 1 \\
&= 2 \times (\mu f.G)1 \\
&= 2 \times ((\lambda f.G)Fact)1 \\
&= 2 \times (\lambda n.\mathsf{isZ}(n)?1:n \times Fact(n-1))1 \\
&= 2 \times \mathsf{isZ}(1)?1:1 \times Fact\ 0 \\
&= 2 \times 1 \\
&= 2
\end{aligned}$$

[3] En matemáticas, x es el punto fijo de f si y sólo si $f(x) = x$.

Let

La construcción let no es necesaria, pero será útil más adelante y por lo tanto también la vamos a considerar:

$$\mathsf{let}\ x = t\ \mathsf{in}\ u$$

es equivalente a $(\lambda x.u)t$.

4.1.2. Gramática de PCF

Los términos válidos de PCF los podemos describir con una gramática formal:

$$\begin{aligned} t ::= x \mid \lambda x.t \mid tt \mid n \in \mathbb{N} \mid t+t \mid t-t \mid t*t \mid t/t \\ \mid \mathsf{isZ}(t)?t:t \mid \mu x.t \mid \mathsf{let}\ x = t\ \mathsf{in}\ t \end{aligned}$$

PCF es Turing completo, es decir que todas las funciones de enteros computables son programables en PCF.

4.1.3. Semántica operacional

La gramática nos dice qué términos podemos escribir sintácticamente. La semántica operacional nos da el significado de los términos, al definir como operan.
Las siguientes reglas tienen la forma $t \to u$, y se lee "t reduce a u":

$$\begin{aligned} (\lambda x.u)t &\to u[t/x] && (\beta \text{ reducción}) \\ p \odot q &\to n && \text{Si } p \odot q = n, \text{ con } \odot = +, -, * \text{ o } / \\ \mathsf{isZ}(0)?t:u &\to t \\ \mathsf{isZ}(1)?t:u &\to u \\ \mu x.t &\to t[\mu x.t/x] \\ \mathsf{let}\ x = t\ \mathsf{in}\ u &\to u[t/x] \end{aligned}$$

También tendremos reglas de congruencia que permitirán reducir un término dentro de otro:

$$\frac{t \to u}{tv \to uv} \qquad \frac{t \to u}{vt \to vu} \qquad \frac{t \to u}{\lambda x.t \to \lambda x.u} \qquad \frac{t \to u}{t \odot v \to u \odot v} \qquad \frac{t \to u}{v \odot t \to v \odot u}$$

Ejercicio: escribir las que faltan.

Observación. La tercera regla, que permite reducir dentro de la función, corresponde a la posibilidad de optimizar programas.

4.1.4. No terminación

Ejemplos 4.2.

1. $\mu x.x \to x[\mu x.x/x] = \mu x.x$
2. Sin μ: $\Omega = (\lambda x.xx)(\lambda x.xx) \to xx[\lambda x.xx/x] = (\lambda x.xx)(\lambda x.xx) = \Omega$

Ejercicio:

$$(\mu f.\lambda x.fx)0$$

¿Termina?

Fix sin μ

$$Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

Sea F una función cualquiera:

$$YF \to (\lambda x.F(xx))(\lambda x.F(xx)) \to F(YF)$$

¡YF es el punto fijo de F!

$$\underbrace{\mu f.G}_{YF} \to (\underbrace{\lambda f.G}_{F})(\underbrace{\mu f.G}_{YF})$$

Ejercicio: escribir el factorial sin usar μ.

4.1.5. Captura de variables

Ejercicio: Reducir los siguientes términos

1. $(\lambda x.\lambda x.x)23$
2. $(\lambda x.\lambda y.((\lambda x.(x+y))x)54$
3. $\left.\begin{array}{c}\text{let } x = 4 \text{ in} \\ \text{let } f = \lambda y.y + x \text{ in} \\ \text{let } x = 5 \text{ in} \\ f6\end{array}\right\}$ En las primeras versiones de Lisp, este ejemplo daba 11 en lugar de 10.

Tenemos que definir precisamente qué significa $t[u/x]$. Damos una definición inductiva:

$$\begin{array}{rll}
x[u/x] = u & & \\
y[u/x] = y & & \\
(\lambda x.t)[u/x] = \lambda x.t & & \\
(\lambda y.t)[u/x] = \lambda y.t[u/x] & & \text{Si } y \notin \mathsf{FV}(u) \\
(\lambda y.t)[u/x] = \lambda z.t[z/y][u/x] & & \text{Si } y \in \mathsf{FV}(u) \\
(tv)[u/x] = t[u/x]v[u/x] & & \\
n[u/x] = n & & \\
(t \odot v)[u/x] = t[u/x] \odot v[u/x] & & \\
(\mathsf{isZ}(t)?v_1 : v_2)[u/x] = \mathsf{isZ}(t[u/x])?v_1[u/x] : v_2[u/x] & & \\
(\mu x.t)[u/x] = \mu x.t & & \\
(\mu y.t)[u/x] = \mu y.t[u/x] & & \text{Si } y \notin \mathsf{FV}(u) \\
(\mu y.t)[u/x] = \mu z.t[z/y][u/x] & & \text{Si } y \in \mathsf{FV}(u) \\
(\mathsf{let}\ x = t\ \mathsf{in}\ v)[u/x] = \mathsf{let}\ x = t[u/x]\ \mathsf{in}\ v & & \\
(\mathsf{let}\ y = t\ \mathsf{in}\ v)[u/x] = \mathsf{let}\ y = t[u/x]\ \mathsf{in}\ v[u/x] & & \text{Si } y \notin \mathsf{FV}(u) \\
(\mathsf{let}\ y = t\ \mathsf{in}\ v)[u/x] = \mathsf{let}\ y = t[u/x]\ \mathsf{in}\ v[z/y][u/x] & & \text{Si } y \in \mathsf{FV}(u)
\end{array}$$

Ejercicio: Definir, por inducción sobre t, $\mathsf{FV}(t)$.
Ejercicio: Definir, por inducción sobre t, $BV(t)$, es decir, el conjunto de variables ligadas de t ("bounded variables").

4.2. Estrategias de reducción

4.2.1. Primeras definiciones

Definición 4.3. Notamos $\to^*$ al cierre reflexivo y transitivo de $\to$. Es decir, si $t \to^* u$, entonces, $t = v_0 \to v_1 \to v_2 \to \cdots \to v_n = u$, con $n \geq 0$. Notamos $\to^+$ al cierre transitivo de $\to$. Es decir, si $t \to^* u$, $t = v_0 \to v_1 \to \cdots \to v_n = u$, con $n \geq 1$.

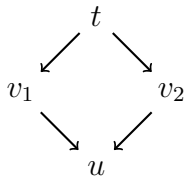
Ejemplo 4.4. $(\lambda x.x + 2)1 \to^* 3$ porque $(\lambda x.x + 2)1 \to 1 + 2 \to 3$.
También, $(\lambda x.x + 2)1 \to^+ 3$, ya que $(\lambda x.x + 2)1 \neq 3$.

Definición 4.5.

1. Un término t está en forma normal si no existe u tal que $t \to u$.
2. Un término t es normalizable (o tiene forma normal) si existe u en forma normal tal que $t \to^* u$.
3. Un término t es fuertemente normalizable si no existe una secuencia infinita v_0, v_1, ... tal que $t \to v_0 \to v_1 \to \ldots$. Es decir, toda secuencia de reducción comenzada en t debe ser finita y terminar en un término en forma normal.

Definición 4.6. Sea $\to_R$ una relación binaria, y $\to_R^*$ su cierre reflexivo y transitivo.

- $\to_R$ satisface la *propiedad del diamante* si $t \to_R v_1$ y $t \to_R v_2$ implica que $v_1 \to_R u$ y $v_2 \to_R u$ para algún u.



- $\to_R$ es *Church-Rosser* o *confluente* si $\to_R^*$ satisface la propiedad del diamante. Es decir, si $t \to_R^* v_1$ y $t \to_R^* v_2$ implica que $v_1 \to_R^* u$ y $v_2 \to_R^* u$ para algún u.
- $\to_R$ tiene *formas normales únicas* si $t \to_R^* v_1$ y $t \to_R^* v_2$, para términos en forma normal v_1 y v_2, implica $v_1 = v_2$.

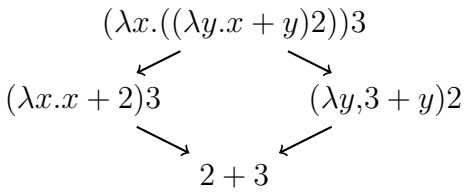
Lema 4.7.

1. Si $\to_R$ satisface la propiedad del diamante, entonces es Church-Rosser.
2. Si $\to_R$ es Church-Rosser, entonces tiene formas normales únicas.

Demostración. Ejercicio.

Teorema 4.8. La relación definida en la Sección 4.1.3 (semántica operacional de PCF) es Church-Rosser. □

Ejemplo 4.9.



Pero esta propiedad, cuando hay términos que no terminan, no es suficiente. Por ejemplo:

$$Fact = \mu f.\underbrace{\lambda n.\mathsf{isZ}(n)?1:n * f(n-1)}_{G} = \mu f.G$$

Entonces:

$$\begin{aligned} Fact\ 0 &= (\mu f.G)0 \\ &\to (G[Fact/f])0 \\ &\to (G[G[Fact/f]/f])0 \\ &\to (G[G[G[Fact/f]/f]/f])0 \\ &\to \cdots \to \infty \end{aligned}$$

$Fact\ 0$ tiene un único resultado, pero no cualquier camino llega a él.
Solución (en este caso): cuando hay un ifz, reducir primero el ifz antes que sus ramas. Ésto, como veremos luego, es una *estrategia*.
Otro ejemplo:

$$\begin{aligned} C_0 &= \lambda x,0 \\ b_1 &= (\mu f.\lambda x.fx)0 \end{aligned}$$

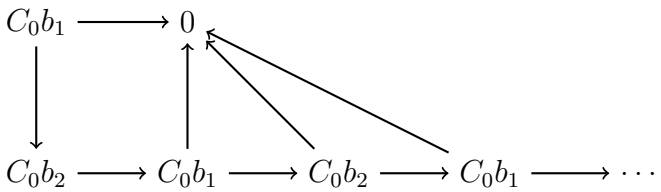
Entonces,

$$b_1 = (\mu f.\lambda x.fx)0 \to \underbrace{(\lambda x.(\mu f.\lambda x.fx)x)0}_{b_2} \to (\mu f.\lambda x.fx)0 = b_1$$

Es decir,

$$b_1 \to b_2 \to b_1 \to b_2 \to \cdots$$

Por lo tanto, tenemos el siguiente diagrama:



En Caml,

```
let rec f x = f x in
let g x = 0 in
g (f 0)
```

no termina nunca.
Mismo ejemplo en Java:

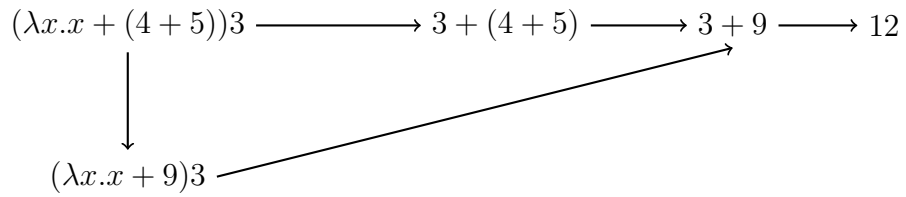
```
class Omega {
  static int f (int x) {return f(x);}
  static int g (int x) {return 0;}
  static public void main (String [] args) {
    System.out.println (g(f(0)));
  }
}
```

La noción de *estrategia de reducción* permite definir el orden en el cual se debe reducir un término.

Definición 4.10. Llamamos *redex* (por *Red*ucible *Ex*pression) a un subtérmino de un término que puede reducir.

4.2.2. Reducción débil

Ejemplo motivador:



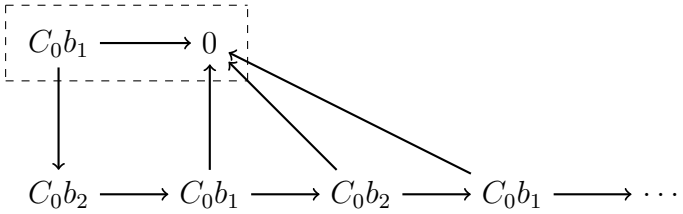
- La dirección $\rightarrow$ dice qué sucede cuando se ejecuta el programa.
- La dirección $\downarrow$ comienza a ejecutar el programa antes de recibir los argumentos, es decir, no ejecuta el programa sino que lo optimiza.

Definición 4.11. Una estrategia de reducción es *débil* si no reduce nunca el cuerpo de una función, es decir, si no reduce bajo λ.

Observación. La estrategia débil no optimiza programa, los ejecuta. Sólo hace falta para ésto eliminar la regla

$$\frac{t \rightarrow u}{\lambda x.t \rightarrow \lambda x.u}$$

4.2.3. Call-by-name



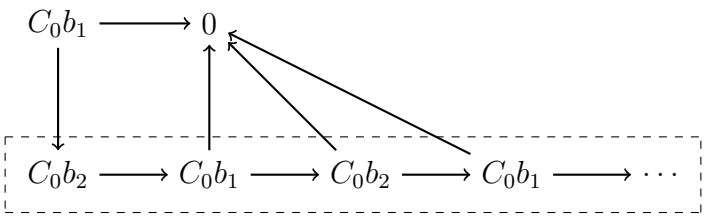
Definición 4.12. La estrategia *call-by-name* reduce siempre el radical más a la izquierda. En caso de ser además débil, será el más a la izquierda que no esté debajo de un λ.

Teorema 4.13 (Estandarización)**.** Si un término reduce a un término en forma normal, entonces la estrategia call-by-name termina. □

Una ventaja de ésta estrategia es el teorema de estandarización. Otra ventaja es que si tenemos, por ejemplo $(\lambda x,0)(\mathit{Fact}\ 10)$ no necesitamos calcular el factorial de 10. Por otro lado, si tenemos $(\lambda x.x+x)(\mathit{Fact}\ 10)$, tendremos que calcular el factorial de 10 dos veces. De todas maneras, la mayoría de los lenguajes que usan call-by-name usan alguna manera de "compartir" información (por ejemplo, con punteros que dicen que $(\lambda x.x+x)(\mathit{Fact}\ 10)$ reduce a $x+x$, donde x es un puntero a $\mathit{Fact}\ 10$. A eso se le llama *reducción lazy*.

Ejercicio: Escribir las reglas de reducción y congruencia que implementan call-by-name.

4.2.4. Call-by-value



Definición 4.14. A los términos t de PCF tales que $\mathsf{FV}(t) = \emptyset$ y que t esté en forma normal, se les llaman *valores*.

Definición 4.15. La estrategia *call-by-value* consiste en evaluar siempre los argumentos antes de pasarlos a la función. La idea es que

$$(\lambda x.t)v$$

reduce sólo cuando v esté en forma normal (si la estrategia es débil, y sólo reducimos términos cerrados, v es un valor).

En $(\lambda x.x+x)(\mathit{Fact}\ 10)$ comenzamos por reducir el factorial, obtenemos 3628800 y recién ahí lo pasamos a la función. De esa manera el factorial es calculado una vez.

Ejercicio: Escribir las reglas que implementan call-by-value.

Observación. Un poco de pereza es necesaria: ifz *siempre* debe evaluar primero la condición, estemos en call-by-name o call-by-value.

4.3. PCF tipado

4.3.1. Introducción

Ejemplos motivadores:

$$\begin{aligned}(\lambda x.x+1)\lambda x.x+2 &\to (\lambda x.x+2)+1\\ \mathsf{isZ}(\lambda x.x)?0:1 &\not\to\\ (\lambda x.x)1\lambda x.x &\to 1\lambda x.x\end{aligned}$$

¡Todo es aplicable a todo! Sin restricciones. Sumar 1 a una función no tiene sentido. Hacer un ifz sobre algo que no es un número o pasarle un argumento a un número, tampoco. **Idea:** detectar este tipo de errores sintácticamente. Por ejemplo:

$$\frac{\lambda x.x \text{ recive un argumento y devuelve lo mismo} \quad 1 \text{ es una constante}}{(\lambda x.x)1 \text{ es una constante}}$$

Es decir, deducimos que no tiene sentido pasarle un argumento a $(\lambda x.x)1$, ya que es una constante, y lo dedujimos sin tener que *ejecutar* el programa.

En matemáticas:

$$\text{Función: Dominio} \longrightarrow \text{Codominio}$$
$$\text{Cualquier conjunto}$$

Ejemplo:

$$f : \mathit{Pares} \to \mathbb{N}$$
$$f(x) \mapsto \frac{x}{2}$$

¿Está bien definido $f(3+(4+1))$? Hay que determinar si $3+(4+1)$ pertenece al dominio, es decir, si es par.

En general, determinar si un objeto cualquiera pertenece a un conjunto cualquiera es un problema *indecidible*.

De todas maneras, $\frac{x}{2}$ lo podemos calcular si x es un número (y no, por ejemplo, una función), y poco importa si es par o no. Así que vamos a restringir las clases de conjuntos que se pueden utilizar como dominios. A estos conjuntos los llamamos **tipos**.

4.3.2. Gramática de PCF tipado

Definición 4.16. Los tipos de PCF los definimos inductivamente por:

- nat (es decir $\mathbb{N}$) es un tipo.
- Si A y B son tipos, $A \Rightarrow B$ es un tipo que representa las funciones de A en B.

Teniendo tipos, tendremos que escribir de qué tipo son cada una de las variables. Como sólo nos vamos a interesar en términos sin variables libres (sólo los subtérminos tendrán variables libres), es suficiente con escribir el tipo cuando se liga la variable. Por ejemplo, en lugar de $\lambda x.x$ la función identidad es una para cada tipo:

- $\lambda x : \mathsf{nat}.x$ es la identidad sobre los naturales
- $\lambda x : \mathsf{nat} \Rightarrow \mathsf{nat}.x$ es la identidad sobre las funciones de naturales en naturales.

En μ y let también es necesario marcar el tipo de la variable ligada.

La gramática de PCF tipado la definimos con una gramática de tipos y una de términos, de la siguiente manera:

$$A ::= \mathsf{nat} \mid A \Rightarrow A$$
$$t ::= x \mid \lambda x : A.t \mid tt \mid n \mid t \odot t \mid \mathsf{isZ}(t)?t{:}t \mid \mu x : A.t \mid \mathsf{let}\ x : A = t\ \mathsf{in}\ t$$

donde $\odot = +, -, \times, /$ y $n \in \mathbb{N}$.

4.3.3. La relación de tipado

Queremos definir inductivamente la relación $t : A$ (es decir, el término t tiene el tipo A). Pero si hay variables libres en t, ¿cómo las tipo?[4].
Por ejemplo:

$$\lambda x : \mathsf{nat}.yx$$

¿Qué tipo tiene y? Claramente tiene que ser una función de nat en algo, pero ¿cómo defino ese algo?

Contextos Un *contexto* nos da tipos para variables, entonces, en vez de decir $\lambda x : \mathsf{nat}.yx : \mathsf{nat} \Rightarrow \mathsf{nat}$, decimos, si $y : \mathsf{nat} \Rightarrow \mathsf{nat}$, entonces $\lambda x : \mathsf{nat}.yx : \mathsf{nat} \Rightarrow \mathsf{nat}$. La notación que usamos es la siguiente:

$$\underbrace{y : \mathsf{nat} \Rightarrow \mathsf{nat}}_{\text{contexto}} \vdash \lambda x : \mathsf{nat}.yx : \mathsf{nat} \Rightarrow \mathsf{nat}$$

Genéricamente, queremos definir la relación $\Gamma \vdash t : A$ que asocia un término t y un contexto Γ a un tipo A.

Definición 4.17. La relación de tipado $\Gamma \vdash t : A$ se define inductivamente por:

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax_v \qquad \frac{}{\Gamma \vdash n : \mathsf{nat}}\ ax_c$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.t : A \Rightarrow B}\ \Rightarrow_i \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B}\ \Rightarrow_e$$

$$\frac{\Gamma \vdash t : \mathsf{nat} \quad \Gamma \vdash u : \mathsf{nat}}{\Gamma \vdash t \odot u : \mathsf{nat}}\ \odot \qquad \frac{\Gamma \vdash t : \mathsf{nat} \quad \Gamma \vdash u : A \quad \Gamma \vdash v : A}{\Gamma \vdash \mathsf{isZ}(t)?u{:}v : A}\ \mathsf{ifz}$$

$$\frac{\Gamma, x : A \vdash t : A}{\Gamma \vdash \mu x : A.t : A}\ \mu \qquad \frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{let}\ x : A = u\ \mathsf{in}\ t : B}\ \mathsf{let}$$

Donde $\odot$ son cuatro reglas, una para cada operación aritmética.

[4] Dijimos que sólo vamos a interesarnos por términos cerrados (sin variables libres), pero como buscamos una definición inductiva, necesitamos poder tipar cada subtérmino, y un subtérmino de un término sin variables libres, puede perfectamente tener variables libres.

Ejemplo 4.18. Tipar $\lambda x : \mathsf{nat} \Rightarrow \mathsf{nat}.x((\lambda y : \mathsf{nat}.y + 2)3)$.
Sean $\Delta = x : \mathsf{nat} \Rightarrow \mathsf{nat}$, y $\Gamma = \Delta, y : \mathsf{nat}$. Entonces,

$$\dfrac{\dfrac{}{\Delta \vdash x : \mathsf{nat} \Rightarrow \mathsf{nat}}\, ax_v \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash y : \mathsf{nat}}\, ax_v \quad \dfrac{}{\Gamma \vdash 2 : \mathsf{nat}}\, ax_c}{\Gamma \vdash y + 2 : \mathsf{nat}}\, +}{\Delta \vdash \lambda y : \mathsf{nat}.y + 2 : \mathsf{nat} \Rightarrow \mathsf{nat}}\, \Rightarrow_i \quad \dfrac{}{\Delta \vdash 3 : \mathsf{nat}}\, ax_c}{\Delta \vdash (\lambda y : \mathsf{nat}.y + 2)3 : \mathsf{nat}}\, \Rightarrow_e}{\dfrac{\Delta \vdash x((\lambda y : \mathsf{nat}.y + 2)3) : \mathsf{nat}}{\vdash \lambda x : \mathsf{nat} \Rightarrow \mathsf{nat}.x((\lambda y : \mathsf{nat}.y + 2)3) : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}}\, \Rightarrow_i}\, \Rightarrow_e$$

Ejercicio 1: tipar factorial.
Ejercicio 2: tipar $\lambda x : A.xx$ para algún A.

4.3.4. Correctitud

Si deducimos el tipo A para un término (usando la Definición 4.17), o sea, sin ejecutar el programa, y luego ejecutamos el programa obteniendo u, entonces queremos que el término u tienga el mismo tipo, ya que la intención fue desde el principio saber qué *tipo* de resultado voy a tener al ejecutar un programa (un número, una función, etc). El teorema de subject reduction o de conservación de tipos (Teorema 4.20) nos asegura eso en el sistema que acabamos de definir. Para demostrar este teorema necesitaremos un resultado previo (Lema 4.19).

Lema 4.19 (Substitución)**.** Si $\Gamma, x : A \vdash u : B$ y $\Gamma \vdash t : A$, entonces $\Gamma \vdash u[t/x] : B$.

Demostración. Procedemos por inducción estructural sobre u.

$u = x$**.** Es decir que $\Gamma, x : A \vdash x : B$, entonces $B = A$, y dado que $x[t/x] = t$, el caso queda demostrado.

$u = y$**.** Es decir que $\Gamma, x : A \vdash y : B$. Pero como $x \neq y$, se puede demostrar por inducción en la relación de tipado que $\Gamma \vdash y : B$. Dado que $y[t/x] = y$, el caso queda demostrado.

$u = \lambda y{:}C.r$**.** Es decir que $\Gamma, x : A \vdash \lambda y{:}C.r : B$, por lo tanto $B = C \Rightarrow D$ y $\Gamma, y : C, x : A \vdash r : D$. Entonces, por hipótesis de inducción, $\Gamma, y : C \vdash r[t/x] : D$, y por la regla $\Rightarrow_i$, $\Gamma \vdash \lambda y{:}C.r[t/x] : B$. Dado que $\lambda y : C.r[t/x] = (\lambda y{:}C.r)[t/x]$, el caso queda demostrado.

$u = rs$**.** Es decir que $\Gamma, x : A \vdash rs : B$. Entonces $\Gamma, x : A \vdash r : C \Rightarrow B$ y $\Gamma, x : A \vdash s : C$, y por hipótesis de inducción, $\Gamma \vdash r[t/x] : C \Rightarrow B$ y $\Gamma \vdash s[t/x] : C$. Por la regla $\Rightarrow_e$, tenemos $\Gamma \vdash r[t/x]s[t/x] : B$. Notar que $r[t/x]s[t/x] = (rs)[t/x]$, por lo que el caso queda demostrado.

$u = n$**.** Es decir que $\Gamma, x : A \vdash n : B$, por lo tanto $B = \mathsf{nat}$ y $\Gamma \vdash n : \mathsf{nat}$. Notar que $n[t/x] = n$, por lo que el caso queda demostrado.

$u = r \odot s$. Es decir que $\Gamma, x : A \vdash r \odot s : B$, y entonces $B = \mathsf{nat}$ y tenemos $\Gamma, x : A \vdash r : \mathsf{nat}$ y $\Gamma, x : A \vdash s : \mathsf{nat}$. Entonces, por hipótesis de inducción $\Gamma \vdash r[t/x] : \mathsf{nat}$ y $\Gamma \vdash s[t/x] : \mathsf{nat}$. Por lo tanto, por regla $\odot$, $\Gamma \vdash r[t/x] \odot s[t/x] : \mathsf{nat}$. Dado que $r[t/x] \odot s[t/x] = (r \odot s)[t/x]$, el caso queda demostrado.

$u = \mathsf{isZ}(r)?s:o$. Es decir que $\Gamma, x : A \vdash \mathsf{isZ}(r)?s : o : B$, y entonces $\Gamma, x : A \vdash r : \mathsf{nat}$, $\Gamma, x : A \vdash s : B$ y $\Gamma, x : A \vdash o : B$. Por hipótesis de inducción, $\Gamma \vdash r[t/x] : \mathsf{nat}$, $\Gamma \vdash s[t/x] : B$ y $\Gamma \vdash o[t/x] : B$. Entonces, por regla ifz, $\Gamma \vdash \mathsf{isZ}(r[t/x])?s[t/x]:o[t/x] : B$. Dado que $\mathsf{isZ}(r[t/x])?s[t/x]:o[t/x] = (\mathsf{isZ}(r)?s:o)[t/x]$, el caso queda demostrado.

$u = \mu y : C.r$. Es decir que $\Gamma, x : A \vdash \mu y : C.r : B$, y entonces $C = B$ y $\Gamma, x : A, y : B \vdash r : B$. Por hipótesis de inducción, $\Gamma, y : B \vdash r[t/x] : B$. Entonces, por regla μ, $\Gamma \vdash \mu y : B.r[t/x] : B$. Dado que $\mu y : B.r[t/x] = (\mu y : C.r)[t/x]$ el caso queda demostrado.

$u = \mathsf{let}\ y : C = r\ \mathsf{in}\ s$. Es decir que $\Gamma, x : A \vdash \mathsf{let}\ y : C = r\ \mathsf{in}\ s : B$, y entonces $\Gamma, y : C, x : A \vdash s : B$ y $\Gamma, x : A \vdash r : C$. Por hipótesis de inducción $\Gamma, y : C \vdash s[t/x] : B$ y $\Gamma \vdash r[t/x] : C$, y entonces por la regla let, $\Gamma \vdash \mathsf{let}\ y : C = r[t/x]\ \mathsf{in}\ s[t/x] : B$. Dado que $\mathsf{let}\ y : C = r[t/x]\ \mathsf{in}\ s[t/x] = (\mathsf{let}\ y : C = r\ \mathsf{in}\ s)[t/x]$, el caso queda demostrado. □

Con el lema anterior podemos demostrar el siguiente teorema.

Teorema 4.20 (Subject reduction)**.** Si $\Gamma \vdash t : A$ y $t \to u$ entonces $\Gamma \vdash u : A$.

Demostración. Procedemos por inducción sobre la relación $\to$.

$(\lambda x : B.u)t \to u[t/x]$. Entonces $\Gamma \vdash (\lambda x : B.u)t : A$, por lo tanto $\Gamma \vdash \lambda x : B.u : B \Rightarrow A$ y $\Gamma \vdash t : B$, y entonces $\Gamma, x : B \vdash u : A$. Entonces, por el Lema 4.19, $\Gamma \vdash u[t/x] : A$.

$p \odot q \to n$ **si** $p \odot q = n$. Entonces $\Gamma \vdash p \odot q : A$, por lo tanto $A = \mathsf{nat}$, y por la regla ax_c, $\Gamma \vdash n : \mathsf{nat}$.

$\mathsf{isZ}(0)?t:u \to t$. Entonces $\Gamma \vdash \mathsf{isZ}(0)?t:u : A$, por lo que $\Gamma \vdash t : A$.

$\mathsf{isZ}(1)?t:u \to u$. Entonces $\Gamma \vdash \mathsf{isZ}(1)?t:u : A$, por lo que $\Gamma \vdash u : A$.

$\mu x.t \to t[\mu x.t/x]$. Entonces $\Gamma \vdash \mu x.t : A$, por lo tanto $\Gamma, x : A \vdash t : A$. Entonces, por Lema 4.19, $\Gamma \vdash t[\mu x.t/x] : A$.

$\mathsf{let}\ x : B = t\ \mathsf{in}\ u \to u[t/x]$. Entonces $\Gamma \vdash \mathsf{let}\ x : B = t\ \mathsf{in}\ u : A$, por lo que $\Gamma, x : B \vdash u : A$ y $\Gamma \vdash t : B$. Entonces, por Lema 4.19, $\Gamma \vdash u[t/x] : A$. □

4.3.5. Normalización fuerte

En esta sección vamos a demostrar el siguiente teorema.

Teorema 4.21 (Normalización fuerte)**.** Todo término tipado que no contenga μ, termina.

Esta demostración utiliza el concepto de semántica denotacional, que aún no vimos (lo veremos en la Sección 4.7). Sin embargo, aquí veremos una versión muy reducida de la semántica denotacional: la idea es interpretar los tipos como conjuntos de términos fuertemente normalizables, y luego verificamos que si un término tiene un tipo, está dentro de la interpretación de dicho tipo, y por lo tanto es fuertemente normalizable.
Sea $\mathcal{SN}$ el conjunto de todos los términos fuertemente normalizables.

Definición 4.22. La interpretación de los tipos es la siguiente:

$$\begin{aligned} [\![\mathsf{nat}]\!] &= \mathcal{SN} \\ [\![A \Rightarrow B]\!] &= \{t \mid \forall r \in [\![A]\!], tr \in [\![B]\!]\} \end{aligned}$$

Es decir, al tipo nat lo interpretamos en el conjunto de todos los términos que normalizan fuertemente y al tipo $A \Rightarrow B$, en el conjunto de términos que aceptan como argumento un término de $[\![A]\!]$ y devuelven un término de $[\![B]\!]$.

El siguiente lema dice que todos los tipos son conjuntos de términos fuertemente normalizables.

Lema 4.23. $\forall A, [\![A]\!] \subseteq \mathcal{SN}$.

Demostración. Procedemos por inducción sobre A.

- Si $A = \mathsf{nat}$, entonces $[\![\mathsf{nat}]\!] = \mathcal{SN} \subseteq \mathcal{SN}$.
- Si $A = B \Rightarrow C$, entonces para todo $t \in [\![A]\!]$ se tiene que para todo $r \in [\![B]\!]$, $tr \in [\![C]\!]$. Por hipótesis de inducción sobre $[\![C]\!]$, $tr \in \mathcal{SN}$ y por lo tanto $t \in \mathcal{SN}$. □

El lema que sigue, nos dice que las variables están en todos los tipos.

Lema 4.24. $\forall A, \forall x,\ x \in [\![A]\!]$.

Demostración. Procedemos por inducción sobre A.

- Si $A = \mathsf{nat}$, entonces claramente $x \in [\![\mathsf{nat}]\!] = \mathcal{SN}$.
- Si $A = B \Rightarrow C$, entonces tenemos que mostrar que $\forall r \in [\![B]\!]$, $xr \in [\![C]\!]$. Vamos a fortalecer la hipótesis y demostrar que si $A = B_1 \Rightarrow B_2 \Rightarrow \cdots \Rightarrow B_n \Rightarrow \mathsf{nat}$, entonces para todo $r_i \in [\![B_i]\!]$ tenemos $xr_1r_2 \ldots r_n \in [\![\mathsf{nat}]\!] = \mathcal{SN}$. Procedemos por inducción sobre n.
  - Si $n = 1$, entonces tenemos que mostrar que $\forall r \in [\![B_1]\!]$, $xr \in \mathcal{SN}$. Por Lema 4.23, $r \in \mathcal{SN}$, y por lo tanto $xr \in \mathcal{SN}$.
  - Por hipótesis de inducción, para $r_i \in [\![B_i]\!]$, con $i = 1, \ldots, n$, tenemos $xr_1 \ldots r_n \in \mathcal{SN}$. Si $r_{n+1} \in [\![B_{n+1}]\!]$, por Lema 4.23, $r_{n+1} \in \mathcal{SN}$ y por lo tanto $xr_1 \ldots r_n r_{n+1} \in \mathcal{SN}$. □

  Por lo tanto, si $A = B_1 \Rightarrow \cdots \Rightarrow B_n \Rightarrow \mathsf{nat}$, tenemos $x \in [\![A]\!]$.

Definición 4.25 (Términos neutrales)**.** A los términos de la siguiente gramática se les llama términos neutrales:

$$N = tt \mid \mathsf{let}\ x : A = t\ \mathsf{in}\ t \mid \mathsf{isZ}(t)?t:t$$

Para cualquier término t, notamos por $|t|$ a la suma de nodos en el árbol formado por todos los caminos de reducción que comienzan en t.

Lema 4.26. Si todas las reducciones de un término neutral N están en $[\![A]\!]$, entonces $N \in [\![A]\!]$.

Demostración. Procedemos por inducción en A.

- Si $A = \mathsf{nat}$ entonces tenemos que mostrar que si todas las reducciones de N están en $[\![\mathsf{nat}]\!] = \mathcal{SN}$, entonces N está en $\mathcal{SN}$, lo cual es cierto por definición.
- Si $A = B \Rightarrow C$ y todas las reducciones de N están en $[\![B \Rightarrow C]\!]$, entonces si $N \to t$, tenemos que para todo $r \in [\![B]\!]$, $tr \in [\![C]\!]$. Queremos mostrar que $Nr \in [\![C]\!]$. Por Lema 4.23, $r \in \mathcal{SN}$. Por lo tanto procedemos por inducción sobre $|r|$ para mostrar que todas las reducciones de Nr están en $[\![C]\!]$, y por lo tanto, por hipótesis de inducción $Nr \in [\![C]\!]$.

  Las reducciones posibles de Nr son las siguientes

  - tr con $N \to t$, y ya tenemos por suposición que $tr \in [\![C]\!]$.
  - Nr' con $r \to r'$, entonces dado que $|r'| < |r|$, podemos usar la hipótesis de inducción y concluir que $Nr' \in [\![C]\!]$.

  Dado que N es un término neutral, N no es de la forma $\lambda x : D.s$ y por lo tanto esas son las únicas reducciones posibles.

  Entonces, como $Nr \in [\![C]\!]$, tenemos que $N \in [\![B \Rightarrow C]\!]$. □

Definición 4.27. Una valuación θ es una función que a cada variable de tipo A le asigna un elemento del conjunto $[\![A]\!]$. Además, usamos la notación $\theta(t)$ para un término t cualquiera para representar $t[\theta(x_1)/x_1][\theta(x_2)/x_2]\dots[\theta(x_n)/x_n]$, donde $FV(t) = \{x_1, x_2, \dots, x_n\}$.
Decimos que una valuación θ es válida en un contexto Γ (notación $\theta \Vdash \Gamma$), si para todo $x : A \in \Gamma$ se tiene $\theta(x) \in [\![A]\!]$.

El Lema 4.28 nos dice que si un término tiene un tipo en un contexto, entonces cualquier valuación, válida en ese contexto, del término, será un elemento de la interpretación de su tipo.

Lema 4.28 (Adecuación)**.** Para todo θ tal que $\theta \Vdash \Gamma$, si $\Gamma \vdash t : A$ entonces $\theta(t) \in [\![A]\!]$.

Demostración. Procedemos por inducción en la relación de tipado.

- $\Gamma, x : A \vdash x : A$.

  $\theta(x) \in [\![A]\!]$ por definición.

- $\Gamma \vdash n : \mathsf{nat}$.

  $\theta(n) = n \in \mathcal{SN} = [\![\mathsf{nat}]\!]$.

- $$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.t : A \Rightarrow B}$$
  $\theta(\lambda x : A.t) = \lambda x : A.\theta(t)$.

  Quiero mostrar que $\theta(\lambda x : A.t) \in [\![A \Rightarrow B]\!]$, es decir que $\forall r \in [\![A]\!]$, tenemos $(\lambda x : A.\theta(t))r \in [\![B]\!]$. Procedemos por inducción en $|\theta(t)| + |r|$, donde $|s|$ es la cantidad de pasos del camino más largo para llegar a la forma normal de s. Como $\theta(t)$ y r son $\mathcal{SN}$, podemos hacer esa inducción.

  Las reducciones posibles de $(\lambda x : A.\theta(t))r$ son:

  - $\theta(t)[r/x]$. Por hipótesis de inducción, si tomamos $\theta' = \theta, x = r$ tenemos $\theta'(t) \in [\![B]\!]$.
  - $(\lambda x : A.t')r$ con $\theta(t) \to t'$. Como $|t'|+|r| < |\theta(t)|+|r|$, la hipótesis de inducción aplica y podemos concluir que $(\lambda x : A.t')r \in [\![B]\!]$.
  - $(\lambda x : A.\theta(t))r'$ con $r \to r'$. Como $|\theta(t)| + |r'| < |\theta(t)| + |r|$, la hipótesis de inducción aplica y podemos concluir que $(\lambda x : A.\theta(t))r' \in [\![B]\!]$.

  Por lo tanto, todas las reducciones de $(\lambda x : A.\theta(t))r$ están en $[\![B]\!]$, y entonces, por Lema 4.26, $(\lambda x : A.\theta(t))r \in [\![B]\!]$.

- $$\frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash r : A}{\Gamma \vdash tr : B}$$
  Por hipótesis de inducción, $\theta(t) \in [\![A \Rightarrow B]\!]$ y $\theta(r) \in [\![A]\!]$. Por lo tanto, por definición de $[\![A \Rightarrow B]\!]$ tenemos $\theta(t)\theta(r) \in [\![B]\!]$. Nótese que $\theta(t)\theta(r) = \theta(tr)$.

- $$\frac{\Gamma \vdash t : \mathsf{nat} \quad \Gamma \vdash r : \mathsf{nat}}{\Gamma \vdash t \otimes r : \mathsf{nat}}$$
  Por hipótesis de inducción $\theta(t) \in \mathcal{SN}$ y $\theta(r) \in \mathcal{SN}$, por lo tanto $\theta(t) \otimes \theta(r) = \theta(t \otimes r) \in \mathcal{SN} = [\![\mathsf{nat}]\!]$.

- $$\frac{\Gamma \vdash t : \mathsf{nat} \quad \Gamma \vdash r : A \quad \Gamma \vdash s : A}{\Gamma \vdash \mathsf{isZ}(t)?r : s : A}$$
  Por hipótesis de inducción $\theta(t) \in \mathcal{SN}$, $\theta(r) \in [\![A]\!]$ y $\theta(s) \in [\![A]\!]$. Procedemos por inducción sobre $|\theta(t)|$ para mostrar que todos los reductos de $\mathsf{isZ}(\theta(t))?\theta(r) : \theta(s)$ están en $[\![A]\!]$.

  Los reductos son:

  - $\mathsf{isZ}(t')?\theta(r) : \theta(s)$ con $\theta(t) \to t'$. Entonces $t' \in \mathcal{SN}$ y $|t'| < |\theta(t)|$. Por lo tanto, la hipótesis de inducción aplica y $\mathsf{isZ}(t')?\theta(r) : \theta(s) \in [\![A]\!]$.
  - $\theta(r)$ si $\theta(t) = 0$, y por hipótesis $\theta(r) \in [\![A]\!]$.
  - $\theta(s)$ si $\theta(t) = n \neq 0$, y por hipótesis $\theta(s) \in [\![A]\!]$.

  Por lo tanto, por Lemma 4.26, $\theta(\mathsf{isZ}(t)?r : s) = \mathsf{isZ}(\theta(t))?\theta(r) : \theta(s) \in [\![A]\!]$.

- $\dfrac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{let}\ x : A = u\ \mathsf{in}\ t : B}$

  Por hipótesis de inducción, para todo θ válido en Γ, $\theta(u) \in [\![A]\!]$, y para todo θ' válido en $\Gamma, x : A$, $\theta'(t) \in [\![B]\!]$. Dado que $\theta(u) \in [\![A]\!]$, tenemos que $\theta' = \theta, x = \theta(u)$ es una valuación válida en $\Gamma, x : A$, y por lo tanto $\theta'(t) \in [\![B]\!]$. Procedemos por inducción sobre $|\theta(t)| + |\theta(u)|$ para mostrar que todos los reductos de $\mathsf{let}\ x : A = \theta(u)\ \mathsf{in}\ \theta(t)$ están en $[\![B]\!]$.

  Los reductos son:

  - $\mathsf{let}\ x : A = u'\ \mathsf{in}\ \theta(t)$ con $\theta(u) \to u'$. Entonces la hipótesis de inducción aplica.
  - $\mathsf{let}\ x : A = \theta(u)\ \mathsf{in}\ t'$ con $\theta(t) \to t'$. Entonces la hipótesis de inducción aplica.
  - $\theta(t)[\theta(u)/x] = \theta'(t) \in [\![B]\!]$.

  Por lo tanto, por Lemma 4.26, $\theta(\mathsf{let}\ x : A = u\ \mathsf{in}\ t) = \mathsf{let}\ x : A = \theta(u)\ \mathsf{in}\ \theta(t) \in [\![A]\!]$. □

Demostración del Teorema 4.21 (Normalización fuerte). Sea $\Gamma \vdash t : A$ donde t no contiene μ. Por el Lema 4.24, $\theta = \mathsf{id}$ es una valuación válida en Γ. Por lo tanto, por el Lema 4.28, tenemos $t \in [\![A]\!]$, y por el Lema 4.23, $t \in \mathcal{SN}$. □

¿Qué sucede con $\Omega = (\lambda x.xx)(\lambda x.xx)$? No es tipable. Es decir, no existe un tipo A tal que $\vdash \Omega : A$.

Ejercicio: Extender PCF con constructores para pares: (t, u), $\pi_1(t, u)$ y $\pi_2(t, u)$. Dar la gramática, semántica operacional, reglas de tipado, y extender la prueba de normalización fuerte para incluir a los pares.

4.4. Inferencia de tipos simples

4.4.1. Introducción

En muchos lenguajes el programador debe indicar el tipo de las variables. Por ejemplo

$$\lambda x : \mathsf{nat}.x + 1$$

Sin embargo, en este caso es un poco innecesario, ya que la operación + sólo puede ocurrir entre naturales, y por lo tanto era posible inferir que x debía ser de tipo nat.
Podemos entonces liberar al programador de escribir los tipos y escribir un algoritmo que los infiera.

Estilo Curry. Escribimos variables sin tipos, como hicimos con PCF no tipado, y definimos independientemente el lenguaje de tipos, como ya lo definimos.
Así, en vez de $\vdash \lambda x : \mathsf{nat}.x + 1 : \mathsf{nat} \Rightarrow \mathsf{nat}$ escribimos $\vdash \lambda x.x + 1 : \mathsf{nat} \Rightarrow \mathsf{nat}$.

Gramática

$$A ::= \mathsf{nat} \mid A \Rightarrow A$$
$$t ::= x \mid \lambda x.t \mid tt \mid n \mid t \odot t \mid \mathsf{isZ}(t)?t{:}t \mid \mu x.t \mid \mathsf{let}\ x = t\ \mathsf{in}\ t$$

Reglas de tipado

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax_v \qquad \frac{}{\Gamma \vdash n : \mathsf{nat}}\ ax_c$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B}\ \Rightarrow_i \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B}\ \Rightarrow_e$$

$$\frac{\Gamma \vdash t : \mathsf{nat} \quad \Gamma \vdash u : \mathsf{nat}}{\Gamma \vdash t \odot u : \mathsf{nat}}\ \odot \qquad \frac{\Gamma \vdash t : \mathsf{nat} \quad \Gamma \vdash u : A \quad \Gamma \vdash v : A}{\Gamma \vdash \mathsf{isZ}(t)?u\!:\!v : A}\ \mathsf{ifz}$$

$$\frac{\Gamma, x : A \vdash t : A}{\Gamma \vdash \mu x.t : A}\ \mu \qquad \frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{let}\ x = u\ \mathsf{in}\ t : B}\ \mathsf{let}$$

Ahora, para el mismo término, podemos derivar diferentes tipos. Por ejemplo:

$$\frac{\dfrac{}{x : \mathsf{nat} \vdash x : \mathsf{nat}}\ ax_v}{\vdash \lambda x.x : \mathsf{nat} \Rightarrow \mathsf{nat}}\ \Rightarrow_i \qquad \frac{\dfrac{}{x : \mathsf{nat} \Rightarrow \mathsf{nat} \vdash x : \mathsf{nat} \Rightarrow \mathsf{nat}}\ ax_v}{\vdash \lambda x.x : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow (\mathsf{nat} \Rightarrow \mathsf{nat})}\ \Rightarrow_i$$

podemos simplemente decir:

$$\frac{\dfrac{}{x : X \vdash x : X}\ ax_v}{\vdash \lambda x.x : X \Rightarrow X}\ \Rightarrow_i$$

donde X es una variable en el sentido de que "desconocido", o "cualquier tipo".

Definición 4.29. Notamos $A(X)$ a un tipo cualquiera que contiene alguna variable X. Notamos θ a una substitución de meta-variables por tipos.

Ejemplo 4.30. Sea $\theta = \mathsf{nat}/X, \mathsf{nat} \Rightarrow \mathsf{nat}/Y$. Entonces,

$$\theta(X \Rightarrow Y) = \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$$

Teorema 4.31. Si $\vdash t : A(X)$, entonces $\vdash t : \theta A(X)$ para cualquier substitución θ.

Ejemplo 4.32. Como vimos anteriormente, $\vdash \lambda x.x : X \Rightarrow X$. Por lo tanto, tomando $\theta = \mathsf{nat}/X$,

$$\vdash \lambda x.x : \mathsf{nat} \Rightarrow \mathsf{nat}$$

Tomando $\theta = (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}/X$ tenemos

$$\vdash \lambda x.x : ((\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}) \Rightarrow ((\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat})$$

4.4.2. Algoritmo de Hindley

Supongamos que queremos tipar $\lambda f, 2 + f1$, entonces ponemos en el contexto $f : X$ y tenemos que llegar a $f : X \vdash 2 + f1 : Y$ para algún X e Y. Dado que hay una suma, tenemos que tipar $f : X \vdash 2 : \mathsf{nat}$ y $f : X \vdash f1 : \mathsf{nat}$, y como la suma sólo puede ser nat, sabemos que $Y = \mathsf{nat}$. Por lo tanto tenemos $f : X \vdash 2 : \mathsf{nat}$ y $f : X \vdash f1 : \mathsf{nat}$. Entonces, como es una aplicación, tenemos que $f : X \vdash f : Z \Rightarrow \mathsf{nat}$ y $f : X \vdash 1 : Z$. Entonces $Z = \mathsf{nat}$ y $X = \mathsf{nat} \Rightarrow \mathsf{nat}$. Por lo tanto:

$$\dfrac{\dfrac{}{f : \mathsf{nat} \Rightarrow \mathsf{nat} \vdash 2 : \mathsf{nat}}\, ax_c \quad \dfrac{\dfrac{}{f : \mathsf{nat} \Rightarrow \mathsf{nat} \vdash f : \mathsf{nat} \Rightarrow \mathsf{nat}}\, ax_v \quad \dfrac{}{f : \mathsf{nat} \Rightarrow \mathsf{nat} \vdash 1 : \mathsf{nat}}\, ax_c}{f : \mathsf{nat} \Rightarrow \mathsf{nat} \vdash f1 : \mathsf{nat}}\, \Rightarrow_e}{\dfrac{f : \mathsf{nat} \Rightarrow \mathsf{nat} \vdash 2 + f1 : \mathsf{nat}}{\vdash \lambda f{,}2 + f1 : \mathsf{nat} \Rightarrow \mathsf{nat}}\, \Rightarrow_i} +$$

Es decir, que en este caso hay una única solución.

Vamos a describir la primera parte del algoritmo inductivamente como una relación

$$\Gamma \vdash t \rightsquigarrow A, \tau$$

entre un contexto Γ y un término t, con un tipo A y un conjunto de ecuaciones τ.

$$\dfrac{}{\Gamma, x : A \vdash x \rightsquigarrow A, \emptyset} \qquad \dfrac{}{\Gamma \vdash n \rightsquigarrow \mathsf{nat}, \emptyset}$$

$$\dfrac{\Gamma, x : A \vdash t \rightsquigarrow B, \tau}{\Gamma \vdash \lambda x.t \rightsquigarrow A \Rightarrow B, \tau} \qquad \dfrac{\Gamma \vdash t \rightsquigarrow A, \tau \quad \Gamma \vdash u \rightsquigarrow B, \sigma}{\Gamma \vdash tu \rightsquigarrow X, \tau \cup \sigma \cup \{A = B \Rightarrow X\}}\ (X \text{ nueva})$$

$$\dfrac{\Gamma \vdash t \rightsquigarrow A, \tau \quad \Gamma \vdash u \rightsquigarrow B, \sigma}{\Gamma \vdash t \odot u \rightsquigarrow \mathsf{nat}, \tau \cup \sigma \cup \{A = \mathsf{nat}, B = \mathsf{nat}\}}$$

$$\dfrac{\Gamma \vdash t \rightsquigarrow B, \tau \quad \Gamma \vdash u \rightsquigarrow A, \sigma \quad \Gamma \vdash v \rightsquigarrow C, \rho}{\Gamma \vdash \mathsf{isZ}(t)?u{:}v \rightsquigarrow A, \tau \cup \sigma \cup \rho \cup \{B = \mathsf{nat}, A = C\}}$$

$$\dfrac{\Gamma, x : A \vdash t \rightsquigarrow B, \tau}{\Gamma \vdash \mu x.t \rightsquigarrow B, \tau \cup \{A = B\}} \qquad \dfrac{\Gamma \vdash u \rightsquigarrow C, \tau \quad \Gamma, x : B \vdash t \rightsquigarrow A, \sigma}{\Gamma \vdash \mathsf{let}\ x = u\ \mathsf{in}\ t \rightsquigarrow A, \tau \cup \sigma \cup \{B = C\}}$$

Ejemplo 4.33.

$$\dfrac{\dfrac{}{f : A \vdash 2 \rightsquigarrow \mathsf{nat}, \emptyset} \quad \dfrac{\dfrac{}{f : A \vdash f \rightsquigarrow A, \emptyset} \quad \dfrac{}{f : A \vdash 1 \rightsquigarrow \mathsf{nat}, \emptyset}}{f : A \vdash f1 \rightsquigarrow X, \{A = \mathsf{nat} \Rightarrow X\}}}{\dfrac{f : A \vdash 2 + f1 \rightsquigarrow \mathsf{nat}, \{A = \mathsf{nat} \Rightarrow X, \mathsf{nat} = \mathsf{nat}, X = \mathsf{nat}\}}{\vdash \lambda f{,}2 + f1 \rightsquigarrow A \Rightarrow \mathsf{nat}, \{A = \mathsf{nat} \Rightarrow X, \mathsf{nat} = \mathsf{nat}, X = \mathsf{nat}\}}}$$

4.4.3. Algoritmo de unificación de Robinson

La segunda parte consiste en resolver las ecuaciones sobre los tipos. El lenguaje de tipos no tiene variables y está formado por la constante nat y el símbolo $\Rightarrow$ de dos argumentos. Para resolver las ecuaciones utilizamos el algoritmo de unificación de Robinson, que permite resolver las ecuaciones de cualquier lenguaje sin variables ligadas.

Definición 4.34. Sea θ una substitución $A_1/X_1, \ldots, A_n/X_n$. Decimos que θ es una *solución* del conjunto de ecuaciones τ si para todo $B = C$ en τ, θB y θC son idénticos.

Teorema 4.35. Si $\vdash t \rightsquigarrow A, \tau$, entonces para toda solución θ de τ, $\vdash t : \theta A$.
Más general: si $\Gamma \vdash t \rightsquigarrow A, \tau$, entonces para toda solución θ de τ, $\theta\Gamma \vdash t : \theta A$, donde $\theta\Gamma$ es la substitución en cada tipo que aparece en Γ.

Algoritmo de unificación de Robinson

- Si una ecuación tiene forma $A \Rightarrow B = C \Rightarrow D$, reemplazarla por las ecuaciones $A = C$ y $B = D$.
- Si una ecuación tiene la forma $\mathsf{nat} = \mathsf{nat}$, borrarla.
- Si una ecuación tiene la forma $\mathsf{nat} = A \Rightarrow B$, o $A \Rightarrow B = \mathsf{nat}$, responder error.
- Si una ecuación tiene forma $X = X$, borrarla.
- Si una ecuación tiene forma $A = X$, $X = A$ y X aparece en A, pero $A \neq X$, responder error.
- Si una ecuación tiene forma $A = X$, $X = A$ y X no aparece en A pero aparece en otras ecuaciones ecuaciones, substituir X por A en todas las ecuaciones.

Si el algoritmo termina en error: en el sistema no había solución.
Si el algoritmo termina sin error, tendremos una lista de ecuaciones $X_1 = A_1, \ldots, X_n = A_n$, con X_i distintas y $\forall ij$, $X_i \notin A_j$. En ese caso, la substitución $\theta = A_1/X_1, \ldots, A_n/X_n$ es una solución.
En ese caso, θ es "principal", en el sentido de que para toda otra solución α, existe una substitución γ tal que $\alpha = \gamma \circ \theta$. Decimos que $\theta = \mathsf{mgu}(\tau)$ ("most general unifier").

Ejemplo 4.36. Continuando con el ejemplo anterior, teníamos

$$\vdash \lambda f{,}2 + f1 \rightsquigarrow A \Rightarrow \mathsf{nat}, \{A = \mathsf{nat} \Rightarrow X, \mathsf{nat} = \mathsf{nat}, X = \mathsf{nat}\}$$

$$\left\{ \begin{array}{c} A = \mathsf{nat} \Rightarrow X \\ \mathsf{nat} = \mathsf{nat} \\ X = \mathsf{nat} \end{array} \right. \Rightarrow \left\{ \begin{array}{c} A = \mathsf{nat} \Rightarrow X \\ X = \mathsf{nat} \end{array} \right. \Rightarrow \left\{ \begin{array}{c} A = \mathsf{nat} \Rightarrow \mathsf{nat} \\ X = \mathsf{nat} \end{array} \right.$$

Solución: $[\mathsf{nat} \Rightarrow \mathsf{nat}/A, \mathsf{nat}/X]$. Por lo tanto,

$$\vdash \lambda f{,}2 + f1 : (\mathsf{nat} \Rightarrow \mathsf{nat}) \Rightarrow \mathsf{nat}$$

4.5. Polimorfismo

4.5.1. Introducción

En la sección anterior vimos que el tipo principal de $\lambda x.x$ es $X \Rightarrow X$. Es decir, $\lambda x.x$ tiene tipo $X \Rightarrow X$ para todo X. Podemos entonces atribuirle el tipo

$$\forall X.(X \Rightarrow X)$$

y agregar una regla según la cual si t tiene tipo $\forall X.A$, entonces t tiene tipo $A[B/X]$ para cualquier B.
A tales lenguajes se los llama *polimórficos*.

Ejemplo 4.37. Usemos los algoritmos de Hindley y Robinson para inferir el tipo del término $\mathsf{let}\ i = \lambda x.x\ \mathsf{in}\ ii$:

$$\dfrac{\dfrac{\overline{i : X \vdash i \rightsquigarrow X, \emptyset} \quad \overline{i : X \vdash i \rightsquigarrow X, \emptyset}}{i : X \vdash ii \rightsquigarrow Y, \{X = X \Rightarrow Y\}} \quad \dfrac{\overline{x : Z \vdash x \rightsquigarrow Z, \emptyset}}{\vdash \lambda x.x \rightsquigarrow Z \Rightarrow Z, \emptyset}}{\vdash \mathsf{let}\ i = \lambda x.x\ \mathsf{in}\ ii \rightsquigarrow Y, \{X = X \Rightarrow Y, X = Z \Rightarrow Z\}}$$

El algoritmo de Robinson finalizará con error al leer la ecuación $X = X \Rightarrow Y$.

Sin embargo, el término equivalente $(\lambda x.x)(\lambda x.x)$ sí puede ser tipado:

$$\dfrac{\dfrac{\dfrac{\overline{x : X \vdash x \rightsquigarrow X, \emptyset}}{\vdash \lambda x.x \rightsquigarrow X \Rightarrow X, \emptyset} \quad \dfrac{\overline{x : Y \vdash x \rightsquigarrow Y, \emptyset}}{\vdash \lambda x.x \rightsquigarrow Y \Rightarrow Y, \emptyset}}{\vdash (\lambda x.x)(\lambda x.x) \rightsquigarrow Z, \{X \Rightarrow X = (Y \Rightarrow Y) \Rightarrow Z\}}$$

$$X \Rightarrow X = (Y \Rightarrow Y) \Rightarrow Z \implies \left\{ \begin{array}{lcl} X & = & Y \Rightarrow Y \\ X & = & Z \end{array} \right\} \implies Y \Rightarrow Y = Z$$

Es decir, el término tiene tipo $Y \Rightarrow Y$ para cualquier Y.

La diferencia entre el primer término y el segundo es que en el primero, i debe tener el mismo tipo en ambas instancias, $Z \Rightarrow Z$ para cualquier Z, en cambio en el segundo ejemplo, cada función identidad puede instanciar la Z de manera diferente.
Con un tipo $\forall$, podemos hacer que el i del let tenga tipo $\forall X.(X \Rightarrow X)$ y cada ocurrencia de i se instancie en un tipo diferente, haciendo que el término $\mathsf{let}\ i = \lambda x.x\ \mathsf{in}\ ii$ pueda ser tipado como $\forall Y.(Y \Rightarrow Y)$.

El ejemplo anterior no es un ejemplo aislado. Podríamos por ejemplo hacer una función map para árboles, sea cual sea el tipo de los elementos en los árboles, lo que permite reusabilidad de código y por lo tanto programas más concisos.

4.5.2. Tipos polimórficos

Podemos dar un tipo cuantificado a las variables ligadas por let, pero vamos a dejar tipos simples para las variables ligadas por λ y μ, de otra manera, está demostrado que no es posible hacer inferencia de tipos.

Tenemos que distinguir tipos sin cuantificadores (los que llamaremos simplemente "tipos") de tipos cuantificados (que llamaremos "esquemas de tipos").

Definición 4.38. Un esquema de tipo tiene forma $\forall X_1 \ldots \forall X_n.A$, donde A es un tipo, con $n \geq 0$.

Definimos entonces una gramática a dos niveles: uno para tipos y otro para esquemas de tipos:

$$A ::= X \mid \mathsf{nat} \mid A \Rightarrow A$$
$$\alpha ::= [A] \mid \forall X.\alpha$$

Si A es un tipo, $[A]$ es un esquema de tipo formado por el tipo A donde ninguna variable está cuantificada.

Definición 4.39. Ahora que tenemos variables y un ligador ($\forall$) en los tipos, extendemos la definición de variables libres (FV) para tipos:

$$\begin{aligned}\mathsf{FV}(X) &= \{X\}\\ \mathsf{FV}(\mathsf{nat}) &= \emptyset\\ \mathsf{FV}(A \Rightarrow B) &= \mathsf{FV}(A) \cup \mathsf{FV}(B)\\ \mathsf{FV}([A]) &= \mathsf{FV}(A)\\ \mathsf{FV}(\forall X.\alpha) &= \mathsf{FV}(\alpha) \setminus \{X\}\end{aligned}$$

Los contextos ahora dan un esquema de tipo a cada variable de término.

Definición 4.40. El sistema de tipos asocia contextos y términos con esquemas de tipos, $\Gamma \vdash t : \alpha$, y viene dado por

$$\frac{}{\Gamma, x : \alpha \vdash x : \alpha}\ ax_v \qquad \frac{}{\Gamma \vdash n : [\mathsf{nat}]}\ ax_c$$

$$\frac{\Gamma, x : [A] \vdash t : [B]}{\Gamma \vdash \lambda x.t : [A \Rightarrow B]}\ \Rightarrow_i \qquad \frac{\Gamma \vdash t : [A \Rightarrow B] \quad \Gamma \vdash u : [A]}{\Gamma \vdash tu : [B]}\ \Rightarrow_e$$

$$\frac{\Gamma \vdash t : [\mathsf{nat}] \quad \Gamma \vdash u : [\mathsf{nat}]}{\Gamma \vdash t \odot u : [\mathsf{nat}]}\ \odot \qquad \frac{\Gamma \vdash t : [\mathsf{nat}] \quad \Gamma \vdash u : [A] \quad \Gamma \vdash v : [A]}{\Gamma \vdash \mathsf{isZ}(t)?u : v : [A]}\ \mathsf{ifz}$$

$$\frac{\Gamma, x : [A] \vdash t : [A]}{\Gamma \vdash \mu x.t : [A]}\ \mu \qquad \frac{\Gamma \vdash t : \alpha \quad \Gamma, x : \alpha \vdash u : [A]}{\Gamma \vdash \mathsf{let}\ x = t\ \mathsf{in}\ u : [A]}\ \mathsf{let}$$

$$\text{Si } X \notin \mathsf{FV}(\Gamma),\ \frac{\Gamma \vdash t : \alpha}{\Gamma \vdash t : \forall X.\alpha}\ \forall_i \qquad \frac{\Gamma \vdash t : \forall X.\alpha}{\Gamma \vdash t : \alpha[A/X]}\ \forall_e$$

Observaciones.

- En la definición anterior se atribuye un esquema de tipo a cada término, en particular a las variables. Las reglas μ y λ piden $x : [A]$, es decir, un esquema sin cuantificar. Sólo let permite darle a la variable cualquier esquema.
- La condición de la regla $\forall_i$ permite tipar, por ejemplo,

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : [X] \vdash x : [X]}\ ax_v}{\vdash \forall x x : [X \Rightarrow X]}\ \Rightarrow_i}{\vdash \lambda x.x : \forall X.[X \Rightarrow X]}\ \forall_i}{\vdash \lambda x.x : [\mathsf{nat} \Rightarrow \mathsf{nat}]}\ \forall_e$$

  pero impide algo como

$$\dfrac{\dfrac{}{x : [X] \vdash x : [X]}\ \cancel{ax_v}}{\cancel{x : [X] \vdash x : \forall X.[X]}}\ \forall_i$$

Ejercicios: Tipar los siguientes términos con tipos polimórficos

1. $\lambda x.x$

2. let $i = \lambda x.x$ in ii

3. $(\lambda f.ff)(\lambda x.x)$

4. $(\lambda x.xx)(\lambda x.xx)$

Teorema 4.41. Todo término tipado que no contenga a μ, termina.

Observación. Si permitimos $\forall$ en λ obtenemos "System F" de Girard [1972] y Reynolds [1974], pero la tipabilidad es indecidible (teorema de Wells [1994]), es decir, no existe algoritmo de inferencia.
Similarmente, $\forall$ en μ hace el sistema indecidible (teorema de Kfoury, Tiuryn, y Urzyczyn [1990]).
Por eso, dar polimorfismo sólo a let es un buen compromiso que permite reusabilidad de código e inferencia de tipos.

4.6. Interpretación

El programa que calcula el valor de un término se llama *intérprete*.

4.6.1. Interpretación en CBN

Supongamos que queremos un programa que tome $(\lambda x.x \times x)4$ y nos devuelva 16.
El programa debe reemplazar todas las x por 4 para obtener 4×4 puede ser muy costoso en tiempo. Una alternativa es guardar $x = 4$ en una estructura anexa llamada contexto y evaluar $x \times x$ en ese contexto.
En un contexto permitimos tener la misma variable varias veces, y se dará la prioridad a la de más a la derecha.

Ejemplo 4.42. En el contexto $x = 3, y - 4, x = 5, z = 8$, x vale 5 y no 3.

Evaluamos términos con variables libres y cuando queremos evaluar la variable en sí, la buscamos en el contexto.
Si el término inicial es cerrado, cada vez que busquemos una variable, esta estará en el contexto.

Si en el contexto encontramos $x = t$, donde t no es un valor, deberíamos encontrar también el contexto en el que t fue evaluado.

Definición 4.43 (Thunk)**.** Un par $\langle t, \Gamma \rangle$ formado por un término y un contexto de evaluación, se llama *thunk*.

Definición 4.44 (Cierre)**.** Cuando queremos en cambio evaluar un término $\lambda x.t$ en un contexto, el resultado no puede ser simplemente $\lambda x.t$, tiene que ser t pero con t "cerrado". Introducimos para esto otro valor llamado *cierre* que se compone de un término $\lambda x.t$ y un contexto Γ, y se nota $\langle x, t, \Gamma \rangle$.

Definición 4.45 (Relación $\hookrightarrow$ en CBN)**.** Vamos a definir la relación $\Gamma \vdash t \hookrightarrow v$ que se lee "t se interpreta a v en Γ".

$$\frac{\Gamma' \vdash t \hookrightarrow v}{\Gamma, x = \langle t, \Gamma' \rangle, \Delta \vdash x \hookrightarrow v}\ x \notin \mathsf{D}(\Delta) \qquad \frac{}{\Gamma \vdash n \hookrightarrow n} \qquad \frac{\Gamma \vdash t \hookrightarrow n \quad \Gamma \vdash u \hookrightarrow m}{\Gamma \vdash t \odot u \hookrightarrow p}\ \text{Si } n \odot m = p$$

$$\frac{}{\Gamma \vdash \lambda x.t \hookrightarrow \langle x, t, \Gamma \rangle} \qquad \frac{\Gamma \vdash t \hookrightarrow \langle x, t', \Gamma' \rangle \quad \Gamma', x = \langle r, \Gamma \rangle \vdash t' \hookrightarrow v}{\Gamma \vdash tr \hookrightarrow v}$$

$$\frac{\Gamma \vdash t \hookrightarrow 0 \quad \Gamma \vdash r \hookrightarrow v}{\Gamma \vdash \mathsf{isZ}(t)?r : s \hookrightarrow v} \qquad \frac{\Gamma \vdash t \hookrightarrow n \quad \Gamma \vdash s \hookrightarrow v}{\Gamma \vdash \mathsf{isZ}(t)?r : s \hookrightarrow v}\ \text{Si } n \neq 0$$

$$\frac{\Gamma, x = \langle r, \Gamma \rangle \vdash t \hookrightarrow v}{\Gamma \vdash \mathsf{let}\ x = r\ \mathsf{in}\ t \hookrightarrow v} \qquad \frac{\Gamma, x = \langle \mu x.t, \Gamma \rangle \vdash t \hookrightarrow v}{\Gamma \vdash \mu x.t \hookrightarrow v}$$

Ejemplo 4.46. (Leer de abajo hacia arriba)

$$\dfrac{\dfrac{}{\vdash \lambda x.x \times x \hookrightarrow \langle x, x \times x, \emptyset \rangle} \qquad \dfrac{\dfrac{\dfrac{}{\vdash 4 \hookrightarrow 4}}{x = \langle 4, \emptyset \rangle \vdash x \hookrightarrow 4} \quad \dfrac{\dfrac{}{\vdash 4 \hookrightarrow 4}}{x = \langle 4, \emptyset \rangle \vdash x \hookrightarrow 4}}{x = \langle 4, \emptyset \rangle \vdash x \times x \hookrightarrow 16}}{\vdash (\lambda x.x \times x)4 \hookrightarrow 16}$$

4.6.2. Interpretación en CBV

En call-by-name es más fácil, ya que siempre se interpretan primero los argumentos, así que, por ejemplo, en lugar de

$$\frac{\Gamma' \vdash t \hookrightarrow v}{\Gamma, x = \langle t, \Gamma' \rangle, \Delta \vdash x \hookrightarrow v}\ x \notin \mathsf{D}(\Delta) \quad \text{tenemos} \quad \frac{}{\Gamma, x = v, \Delta \vdash x \hookrightarrow v}\ x \notin \mathsf{D}(\Delta)$$

Lo mismo sucede con let. Así que los contextos ligan variables con valores, no con thunks. Sin embargo, la regla μ pide substituir la variable por una expresión, no por un valor, y si evaluamos esto antes de introducirlo al contexto, el cálculo no termina.

Por lo tanto el contexto contendrá valores extendidos que son o bien valores o bien thunks formados por un término $\mu x.t$ y un contexto.

Definición 4.47 (Relación $\hookrightarrow$ en CBV)**.**

$$\frac{}{\Gamma, x = v, \Delta \vdash x \hookrightarrow v}\ x \notin \mathsf{D}(\Delta) \qquad \frac{\Gamma' \vdash \mu y.t \to v}{\Gamma, x = \langle \mu y.t, \Gamma \rangle', \Delta \vdash x \hookrightarrow v}\ x \notin \mathsf{D}(\Delta)$$

$$\frac{}{\Gamma \vdash n \hookrightarrow n} \qquad \frac{\Gamma \vdash t \hookrightarrow n \quad \Gamma \vdash u \hookrightarrow m}{\Gamma \vdash t \odot u \hookrightarrow p}\ \text{Si } n \odot m = p$$

$$\frac{}{\Gamma \vdash \lambda x.t \hookrightarrow \langle x, t, \Gamma \rangle} \qquad \frac{\Gamma \vdash r \hookrightarrow w \quad \Gamma \vdash t \hookrightarrow \langle x, t', \Gamma' \rangle \quad \Gamma', x = w \vdash t' \hookrightarrow v}{\Gamma \vdash tr \hookrightarrow v}$$

$$\frac{\Gamma \vdash t \hookrightarrow 0 \quad \Gamma \vdash r \hookrightarrow v}{\Gamma \vdash \mathsf{isZ}(t)?r : s \hookrightarrow v} \qquad \frac{\Gamma \vdash t \hookrightarrow n \quad \Gamma \vdash s \hookrightarrow v}{\Gamma \vdash \mathsf{isZ}(t)?r : s \hookrightarrow v}\ \text{Si } n \neq 0$$

$$\frac{\Gamma \vdash r \hookrightarrow w \quad \Gamma, x = w \vdash t \hookrightarrow v}{\Gamma \vdash \mathsf{let}\ x = r\ \mathsf{in}\ t \hookrightarrow v} \qquad \frac{\Gamma, x = \langle \mu x.t, \Gamma \rangle \vdash t \hookrightarrow v}{\Gamma \vdash \mu x.t \hookrightarrow v}$$

4.7. Semántica denotacional

4.7.1. Primeras definiciones

Sintaxis (o gramática) : Cómo escribir los términos. Cuáles son válidos y cuáles no.

Semántica: Qué significan.

Ejemplo: "A perro un". Sintacticamente correcto. Semánticamente incorrecto, ya que la frase no significa nada.

Definición 4.48 (Semántica)**.** La semántica de un lenguaje es una relación $\hookrightarrow$ que a cada expresión le asocia *algo* que le da significado.

Semántica denotacional (en programas deterministas). Para cada programa p, la relación entre las entradas y las salidas de p es una función que escribimos $[\![p]\!]$. La relación se define entonces como

$$p, e \hookrightarrow s \iff [\![p]\!]e = s$$

La pregunta es, obviamente, cómo definir $[\![p]\!]$. (Lo veremos más adelante en esta sección).

Semántica operacional a grandes pasos También llamada semántica operacional a grandes pasos o semántica natural. Consiste en dar una definición inductiva de $\hookrightarrow$ que nos relacione un término con su valor. Por ejemplo, el intérprete de la Sección 4.6:

$$\underbrace{(\lambda x.\lambda y.x + y)((\lambda x.x)4)5}_{\text{Significado de esta expresión: 9}} \hookrightarrow 9$$

En ese ejemplo damos semántica de acuerdo a lo que calcula.
Así, si considero

$$(\lambda x.\lambda y.x + y)4\ 5 \quad \text{y} \quad (\lambda x.\lambda y.x + y)5\ 4$$

puedo ver que los 3 programas tienen la misma semántica.

Semántica operacional a pequeños pasos También llamada semántica por reescritura. Consiste en definir $\hookrightarrow$ a partir de otra relación $\to$ que describe las etapas elementales. Ejemplo:

$$(\lambda x.(x \times x) + x)4 \to (4 \times 4) + 4 \to 16 + 4 \to 20$$

$$t \hookrightarrow r \iff t \to^* r \qquad \text{y } r \text{ irreducible}$$

donde $\to^*$ es la clausura reflexiva y transitiva de $\to$.

La no terminación. Un programa puede dar un resultado, producir un error o no terminar. Los errores se pueden considerar como resultados particulares. Para expresar programas que no terminan hay varias formas de expresar su semántica:
La primera consiste en considerar que si t no termina, entonces no existe r tal que $t \hookrightarrow r$.
La segunda consiste en agregar un elemento particular $\bot$ a los valores de salida y considerar que si t no termina, entonces $t \hookrightarrow \bot$.

4.7.2. La semántica denotacional de PCF tipado

En general, en los lenguajes funcionales buscamos reducir la distancia que separa la noción de programa de la de función. Es decir, se busca reducir la distancia entre un programa y su semántica denotacional.

Interpretación de los tipos. A cada tipo le asociamos un conjunto:

$$\begin{aligned}[\![\mathsf{nat}]\!] &= \mathbb{N}\\ [\![A \Rightarrow B]\!] &= [\![A]\!] \to [\![B]\!]\end{aligned}$$

donde $A \to B$ es el conjunto de funciones de A en B.

Interpretación de los términos. A cada término t de tipo A le asociamos un elemento $[\![t]\!]$ del conjunto $[\![A]\!]$. Si t tiene variables libres, necesitamos dar una función que a cada $x : A$ en el contexto Γ, le asigne un elemento $a \in [\![A]\!]$.

$$\begin{aligned}[\![x]\!]_\theta &= \theta(x)\\ [\![\lambda x : A.t]\!]_\theta &= \lambda a : [\![A]\!].[\![t]\!]_{\theta,x=a}\\ [\![tr]\!]_\theta &= [\![t]\!]_\theta [\![r]\!]_\theta\\ [\![n]\!]_\theta &= n\\ [\![t \odot r]\!]_\theta &= [\![t]\!]_\theta \odot [\![r]\!]_\theta\\ [\![\mathsf{isZ}(t)?r:s]\!]_\theta &= \begin{cases} [\![r]\!]_\theta & \text{si } [\![t]\!]_\theta = 0\\ [\![s]\!]_\theta & \text{si } [\![t]\!]_\theta \in \mathbb{N}^*\end{cases}\\ [\![\mathsf{let}\ x : A = r\ \mathsf{in}\ t]\!]_\theta &= [\![t]\!]_{\theta,x=[\![r]\!]_\theta}\end{aligned}$$

Hasta ahí es trivial: un programa es una función y su semántica es la misma función. Esta trivialidad es uno de los objetivos de los lenguajes funcionales.

Observaciones.

1. $t/0$ tirar error en PCF y no está definida en matemática. Para que esta definición sea correcta hay que agregar un elemento error a cada conjunto $[\![A]\!]$ y adaptar la definición
$$[\![t/r]\!] = \begin{cases} \mathsf{error} & \text{si } [\![r]\!] = 0\\ [\![t]\!]/[\![r]\!] & \text{si } [\![t]\!] \in \mathbb{N}, [\![r]\!] \in \mathbb{N}^*\end{cases}$$
2. Aún no dijimos como interpretar μ.

La construcción μ es la única donde la semántica denotacional es interesante, porque es la única que se aleja de matemática, respecto a la definición de funciones:
En matemática, contrariamente a PCF, sólo podemos tomar un punto fijo *si este existe* y si hay varios, ¡tenemos que especificar cual!

Ejemplo 4.49.

1. $f(x) = x + 1$ no tiene punto fijo. Pero en PCF $\mu x : \mathsf{nat}.x + 1$ es válido.

2. $\lambda f : \mathsf{nat} \Rightarrow \mathsf{nat}.\lambda x : \mathsf{nat}.(fx) + 1$ tampoco tiene punto fijo... y basta con cmabiar el primer λ y tengo el μ.

3. $\lambda x : \mathsf{nat}.x$ tiene infinitos puntos fijos.

4. $\mu x : \mathsf{nat}.x$ también.

Teorema 4.50. Si tomamos el punto fijo de una función que no tiene punto fijo o que tiene varios, el programa que obtenemos no termina.

Observación. También existen programas con un sólo punto fijo y que no terminan en PCF. Por ejemplo $\lambda x : \mathsf{nat}.x + x$.

Para comprender la semántica denotacional del punto fijo necesitamos comprender la semántica de los términos que no terminan.

- La semántica operacional a pequeños pasos no atribuye ningún resultado a estos términos.

- La semántica operacional a grandes pasos tampoco, pero podemos completar la relación $\hookrightarrow$ agregando $\bot$ tal que $\mu x : \mathsf{nat}.(x + 1) \hookrightarrow \bot$.

- En la semántica denotacional la idea es hacer lo mismo: definir una función parcial $[\![\cdot]\!]$ tal que $[\![\mu x : \mathsf{nat}.(x + 1)]\!]$ no esté definido, y adjuntamos un valor $\bot$ a $[\![\mathsf{nat}]\!]$ tal que

$$[\![\mu x : \mathsf{nat}.(x + 1)]\!] = \bot$$

Agregando el valor $\bot$, cuando interpretamos, por ejemplo $t + r$, comenzamos por interpretar t y r y si alguno loopea, también lo hace $t + r$. Entonces:

$$[\![t + r]\!]_\theta = \begin{cases} [\![t]\!]_\theta + [\![r]\!]_\theta & \text{si } [\![t]\!]_\theta, [\![r]\!]_\theta \in \mathbb{N} \\ \bot & \text{si } [\![t]\!]_\theta = \bot \text{ o } [\![r]\!]_\theta = \bot \end{cases}$$

Ahora vemos que la función $[\![\lambda x : \mathsf{nat}.x + 1]\!]$ que no tenía punto fijo cuando $[\![\mathsf{nat}]\!] = \mathbb{N}$, si $[\![\mathsf{nat}]\!] = \mathbb{N} \cup \bot$ le podemos dar $\bot$ como semántica a ese término.
$[\![\lambda x : \mathsf{nat}.x]\!]$ que tenía muchos puntos fijos, ahora tiene uno más: $\bot$.
$[\![\lambda x : \mathsf{nat}.x + x]\!]$ que tenía sólo un punto fijo en 0, ahora tiene 2: 0 y $\bot$, y el segundo es el que queremos atribuirle como semántica.

Definición 4.51 (Orden de Scott)**.** $\bot$ es el elemento más chico de cualquier conjunto que lo contenga.

Definimos entonces $[\![\mu x : \mathsf{nat}.t]\!]$ como el punto fijo más chico de $[\![\lambda x : \mathsf{nat}.t]\!]$.

Semántica denotacional completa de PCF (sin error)

$$
\begin{aligned}
[\![x]\!]_\theta &= \theta(x) \\
[\![\lambda x : A.t]\!]_\theta &= \lambda a : [\![A]\!].[\![t]\!]_{\theta,x=a} \\
[\![tr]\!]_\theta &= [\![t]\!]_\theta [\![r]\!]_\theta \\
[\![n]\!]_\theta &= n \\
[\![t \odot r]\!]_\theta &= \begin{cases} [\![t]\!]_\theta \odot [\![r]\!]_\theta & \text{si } [\![t]\!]_\theta, [\![r]\!]_\theta \in \mathbb{N} \\ \bot_{\mathbb{N}} & \text{si no} \end{cases} \\
[\![\mathsf{isZ}(t)?r:s]\!]_\theta &= \begin{cases} [\![r]\!]_\theta & \text{si } [\![t]\!]_\theta = 0 \\ [\![s]\!]_\theta & \text{si } [\![t]\!]_\theta \in \mathbb{N}^* \\ \bot_A & \text{si } [\![t]\!]_\theta = \bot_{\mathbb{N}} \text{ y } A \text{ es el tipo del término} \end{cases} \\
[\![\mathsf{let}\ x : A = r\ \mathsf{in}\ t]\!]_\theta &= [\![t]\!]_{\theta,x=[\![r]\!]_\theta} \\
[\![\mu x : A.t]\!]_\theta &= \mathsf{FIX}(\lambda a : [\![A]\!].[\![t]\!]_{\theta,x=a})
\end{aligned}
$$

donde $\mathsf{FIX}(f)$ es el mínimo punto fijo de f.

Observación. El mínimo de $[\![A]\!]$ es $\bot_A$ y el mínimo de $[\![A \Rightarrow B]\!] = \bot_{[\![A]\!]\to[\![B]\!]}$ que es la función constante $\bot_{[\![B]\!]}$.

Teorema 4.52. Si $\Gamma \vdash t : A$, entonces para toda valuación θ válida en Γ se tiene $[\![t]\!]_\theta \in [\![A]\!]$.

Demostración. Ejercicio (Ayuda: Inducción sobre la derivación de $\Gamma \vdash t : A$). □

Teorema 4.53 (Soundness)**.** Si $\Gamma \vdash t : A$ y $t \to r$, entonces para toda valuación θ válida en Γ se tiene $[\![t]\!]_\theta = [\![r]\!]_\theta$.

Demostración. Ejercicio (Ayuda: Inducción sobre la relación $\to$). □

4.8. Rapid(ísim)a descripción de la lógica lineal

Esta sección es una adaptación libre de la Sección 3.1 del artículo de Di Cosmo y Miller [2016]. Para más detalles, se sugiere recurrir a dicha fuente.

4.8.1. Introducción

La lógica lineal fue introducida por Jean-Yves Girard [1987]. Aquí daremos una presentación por medio de cálculo de secuentes, ya que es muy similar a las reglas de tipado que hemos visto en las secciones anteriores.
La idea principal a retener es que la lógica lineal es una lógica de recursos: la fórmula $A \Rightarrow B$ normalmente se entiende como "Si me das A, te devuelvo B", pero, en la práctica, significa más bien "Si me das tantas A como necesite, te devuelvo B". Por ejemplo, el término

$$\lambda x.x + \mathsf{Fact}\ x$$

tiene tipo $\mathsf{nat} \Rightarrow \mathsf{nat}$, pero para calcular $x + \mathsf{Fact}\ x$, se necesitan dos copias del número x. Es decir, el recurso (el número x), fue duplicado para poder calcular el resultado. Si,

por ejemplo, el recurso fuese Fact 1000, y trabajamos en call-by-value (para asegurar que si hay un resultado, llegaremos al mismo), entonces duplicar el recurso tiene un costo. En lógica lineal no podemos duplicar recursos, salvo que explícitamente se autorice. Así, el tipo $\mathsf{nat} \multimap \mathsf{nat}$ significa: "Si me das un nat, te devuelvo un nat usándolo exactamente una vez", y, en cambio, el tipo $!\mathsf{nat} \multimap \mathsf{nat}$ significa $\mathsf{nat} \Rightarrow \mathsf{nat}$.
Ésta lógica nos será de utilidad para definir cálculos cuánticos, ya que el teorema de no clonado (Teorema 1.27) nos impide clonar recursos cuánticos.
De la misma manera, la función $\lambda x,1$, que descarta su argumento, no podríamos decir que tiene tipo $\mathsf{nat} \multimap \mathsf{nat}$. En cambio, se utiliza el tipo $?\mathsf{nat} \multimap \mathsf{nat}$, donde ? se utiliza, justamente, para permitir descartar el argumento.

4.8.2. Cálculo de secuentes

Los conectivos de la lógica lineal se dividen en multiplicativos y aditivos, y los conectivos clásicos tienen su paralelo en ambos:

Clásico		Multiplicativo		Aditivo	
$\wedge$	(conjunción)	$\otimes$	(tensor)	$\&$	(with)
$\boldsymbol{T}$	(verdadero)	$\mathbf{1}$	(uno)	$\top$	(top)
$\vee$	(disjunción)	$\parr$	(par)	$\oplus$	(oplus)
$\boldsymbol{F}$	(falso)	$\bot$	(bottom)	0	(cero)

Implicación lineal: $A \multimap B := \neg A \parr B$
Implicación intuicionista: $A \Rightarrow B := !A \multimap B$
A ! y a ? se les llama exponenciales, y existen las equivalencias normales de los exponenciales con respecto a la suma y la adición: $n^{a+b} = n^a n^b$.

$$!(A \,\&\, B) \equiv (!A \otimes !B) \qquad ?(A \oplus B) \equiv (?A \parr ?B)$$

$$!\top \equiv \mathbf{1} \qquad ?0 \equiv \bot$$

Donde $A \equiv B$ significa que $(A \multimap B) \wedge (B \multimap A)$ es derivable.
A continuación se detallan las reglas en el formato $\Delta \vdash \Gamma$, que significa que la conjunción (multiplicativa) de las fórmulas en Δ, implican la disjunción (multiplicativa) de las fórmulas en Γ.

Gramática

$$A := p \mid \neg A \mid A \otimes A \mid A \parr A \mid A \,\&\, A \mid A \oplus A \mid \mathbf{1} \mid \bot \mid \top \mid 0$$

Donde p representa una fórmula atómica.

Reglas de indentidad y negación

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Delta \vdash B, \Gamma \quad \Delta', B \vdash \Gamma'}{\Delta, \Delta' \vdash \Gamma, \Gamma'}\ cut \qquad \frac{\Delta \vdash A, \Gamma}{\Delta, \neg A \vdash \Gamma}\ \neg_l \qquad \frac{\Delta, A \vdash \Gamma}{\Delta \vdash \neg A, \Gamma}\ \neg_r$$

Reglas multiplicativas

$$\frac{\Delta \vdash \Gamma}{\Delta, \mathbf{1} \vdash \Gamma}\ \mathbf{1}_l \qquad \frac{}{\vdash \mathbf{1}}\ \mathbf{1}_r \qquad \frac{\Delta, A, B \vdash \Gamma}{\Delta, A \otimes B \vdash \Gamma}\ \otimes_l \qquad \frac{\Delta \vdash A, \Gamma \quad \Delta' \vdash B, \Gamma'}{\Delta, \Delta' \vdash A \otimes B, \Gamma, \Gamma'}\ \otimes_r$$

$$\frac{}{\bot \vdash}\ \bot_l \qquad \frac{\Delta \vdash \Gamma}{\Delta \vdash \bot, \Gamma}\ \bot_r \qquad \frac{\Delta, A \vdash \Gamma \quad \Delta', B \vdash \Gamma'}{\Delta, \Delta', A \parr B \vdash \Gamma, \Gamma'}\ \parr_l \qquad \frac{\Delta \vdash A, B, \Gamma}{\Delta \vdash A \parr B, \Gamma}\ \parr_r$$

Reglas aditivas

$$\frac{}{\Delta, 0 \vdash \Gamma}\ 0_l \qquad \frac{\Delta, A \vdash \Gamma}{\Delta, A \,\&\, B \vdash \Gamma}\ \&_{l1} \qquad \frac{\Delta, B \vdash \Gamma}{\Delta, A \,\&\, B \vdash \Gamma}\ \&_{l2} \qquad \frac{\Delta \vdash A, \Gamma \quad \Delta \vdash B, \Gamma}{\Delta \vdash A \,\&\, B, \Gamma}\ \&_r$$

$$\frac{}{\Delta \vdash \top, \Gamma}\ \top_r \qquad \frac{\Delta, A \vdash \Gamma \quad \Delta, B \vdash \Gamma}{\Delta, A \oplus B \vdash \Gamma}\ \oplus_l \qquad \frac{\Delta \vdash A, \Gamma}{\Delta \vdash A \oplus B, \Gamma}\ \oplus_{r1} \qquad \frac{\Delta \vdash B, \Gamma}{\Delta \vdash A \oplus B, \Gamma}\ \oplus_{r2}$$

Reglas de los exponenciales

$$\frac{!\Delta, A \vdash ?\Gamma}{!\Delta, ?A \vdash ?\Gamma}\ ?_l \qquad \frac{!\Delta \vdash A, !\Gamma}{!\Delta \vdash !A, ?\Gamma}\ !_r$$

$$\frac{\Delta \vdash \Gamma}{\Delta, !A \vdash \Gamma}\ W_! \qquad \frac{\Delta, !A, !A \vdash \Gamma}{\Delta, !A \vdash \Gamma}\ C_! \qquad \frac{\Delta, A \vdash \Gamma}{\Delta, !A \vdash \Gamma}\ D_!$$

$$\frac{\Delta \vdash \Gamma}{\Delta \vdash ?A, \Gamma}\ W_? \qquad \frac{\Delta \vdash ?A, ?A, \Gamma}{\Delta \vdash ?A, \Gamma}\ C_? \qquad \frac{\Delta \vdash A, \Gamma}{\Delta \vdash ?A, \Gamma}\ D_?$$

4.8.3. Un ejemplo simple de sistema de tipos lineal

Supongamos que queremos redefinir el fragmento de PCF que no contiene al punto fijo, de manera que los tipos simples permitan sólo funciones que usan su variable una sola vez, y si se van a utilizar más de una vez o ninguna, entonces el tipo deberá ser marcado con !.
En particular, en este sistema podremos duplicar números naturales, pero no funciones.
Definimos el siguiente sistema de tipos para obtener el efecto deseado (ver la diferencia con la relación de tipado de la Definición 4.17).

$$\frac{}{x : A \vdash x : A}\ ax_v \qquad \frac{}{\vdash n : \mathsf{!nat}}\ ax_c$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x{:}A.t : A \multimap B}\ \multimap_i \qquad \frac{\Gamma \vdash t : A \multimap B \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash tu : B}\ \multimap_e$$

$$\frac{\Gamma \vdash t : \mathsf{!nat} \quad \Delta \vdash u : \mathsf{!nat}}{\Gamma, \Delta \vdash t \odot u : \mathsf{!nat}}\ \odot \qquad \frac{\Gamma \vdash t : \mathsf{!nat} \quad \Delta \vdash u : A \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash \mathsf{isZ}(t)?u{:}v : A}\ \mathsf{ifz}$$

$$\frac{\Gamma, x : A \vdash t : B \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash \mathsf{let}\ x : A = u\ \mathsf{in}\ t : B}\ \mathsf{let}$$

$$\frac{\Gamma \vdash t : A}{\Gamma, x : !B \vdash t : A}\ W \qquad \frac{\Gamma, x : !B, y : !B \vdash t : A}{\Gamma, x : !B \vdash t[x/y] : A}\ C$$

Donde $\Gamma \cup \Delta = \emptyset$.

Ejemplo 4.54. El término $\mathsf{apply} := \lambda x.\lambda y.x(y+y)$ puede ser tipado como $(!\mathsf{nat} \multimap !\mathsf{nat}) \multimap !\mathsf{nat} \multimap !\mathsf{nat}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : !\mathsf{nat} \multimap !\mathsf{nat} \vdash x : !\mathsf{nat} \multimap !\mathsf{nat}}\, ax_v \qquad \dfrac{\dfrac{\dfrac{}{y : !\mathsf{nat} \vdash y : !\mathsf{nat}}\, ax_v \quad \dfrac{}{z : !\mathsf{nat} \vdash z : !\mathsf{nat}}\, ax_v}{y : !\mathsf{nat}, z : !\mathsf{nat} \vdash y + z : !\mathsf{nat}}\, +}{y : !\mathsf{nat} \vdash y + y : !\mathsf{nat}}\, C}{x : !\mathsf{nat} \multimap !\mathsf{nat}, y : !\mathsf{nat} \vdash x(y+y) : !\mathsf{nat}}\, \multimap_e}{x : !\mathsf{nat} \multimap !\mathsf{nat} \vdash \lambda y.x(y+y) : !\mathsf{nat} \multimap !\mathsf{nat}}\, \multimap_i}{\vdash \lambda x.\lambda y.x(y+y) : (!\mathsf{nat} \multimap !\mathsf{nat}) \multimap !\mathsf{nat} \multimap !\mathsf{nat}}\, \multimap_i$$

Notar que el segundo argumento no es lineal, ya que se utiliza dos veces, y eso se ve reflejado en el tipo ($!\mathsf{nat}$), en cambio, el primer argumento es lineal ($!\mathsf{nat} \multimap !\mathsf{nat}$) y se utiliza sólo una vez.

En la práctica veremos que todas las reglas de tipado que dimos más arriba, son lógicamente derivables.

Capítulo 5

Extensiones cuánticas al lambda cálculo

5.1. Control clásico, datos cuánticos

El paradigma de control clásico y datos cuánticos se atribuye a Peter Selinger [2004], y la idea es que en cualquier lenguaje de programación cuántico, los datos (qubits), son cuánticos, pero el flujo de control del programa será clásico. Es decir, no se pueden superponer programas, sólo datos. La primer extensión a lambda cálculo en este paradigma vino de la mano de Selinger y Valiron [2006], y es el cálculo que vamos a estudiar en la Sección 5.1.1.

5.1.1. El cálculo de Selinger y Valiron

Gramática

La gramática del cálculo es la siguiente:

$$t ::= x \mid \lambda x.t \mid tt \mid \mathsf{isZ}(t)?t:t \mid 0 \mid 1 \mid \mathsf{new} \mid \mathsf{meas} \mid U \mid * \mid (t,t) \mid \mathsf{let}\ (x,y) = t\ \mathsf{in}\ t$$

Donde

- new mapea un bit clásico en un qubit.
- meas mapea un qubit en un bit clásico, a través de una medición cuántica.
- U es un cualquier matriz unitaria.

Las que siguen son notaciones prácticas:

$$\begin{aligned}(t_1, t_2, \ldots, t_n) &= (t_1, (t_2, (\ldots, t_n)))\\ \mathsf{let}\ x = r\ \mathsf{in}\ t &= (\lambda x.t)r\\ \lambda(x,y).t &= \lambda z.(\mathsf{let}\ (x,y) = z\ \mathsf{in}\ t)\end{aligned}$$

Programas

El estado de un programa se representa con una tripleta $[q, \ell, t]$ donde

- q es un vector normalizado de $\bigotimes_{i=1}^{n} \mathbb{C}^2$, para algún $n > 0$.

- t es un término lambda.
- ℓ es una función de W en $\mathbb{N}^{\leq n}$, donde $\mathsf{FV}(t) \subseteq W$. A L se la llama función de linkeado.

La función de linkeado linkea variables libres específicas de t a qubits específicos de q.

Ejemplo 5.1. La tripleta

$$[\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \{x \mapsto 2\}, \lambda y.Xx]$$

representa el programa que comienza con el estado de Bell β_{00} y al pasarle un argumento cualquiera aplica la compuerta X (not) al segundo qubit, transformando dicho estado en β_{01}, pero aún no hemos dicho cómo reduce un programa para poder verificar esta última afirmación.

Notación. Para simplificar la notación, se usan p_i para denotar las variables libres x tal que $\ell(x) = i$. Es decir, p_i es la variable que referencia al i-ésimo qubit. De esa manera, un programa $[q, \ell, t]$ es abreviado en $[q, t'''']$, donde $t' = t[p_{\ell(x_1)}/x_1] \dots [p_{\ell(x_n)}/x_n]$.

Ejemplo 5.2. La tripleta del ejemplo 5.1 se escribe abreviadamente como

$$[\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \lambda y.Xp_2]$$

Observación. El teorema de no clonado (ver Teorema 1.27) en este lenguaje es traducido en que cada qubit cuántico no puede ser referenciado más de una vez a través de la función de linkeado. Sintácticamente esta restricción se traduce en la condición de linealidad: una lambda abstracción $\lambda x.t$ se dice lineal si la variable x aparece exactamente una vez en t. El sistema de tipos se encargará de que las variables linkeadas a través de la función ℓ se usen linealmente, mientras al resto de las variables se permite una utilización no-lineal.

Semántica operacional: preliminares

Estrategia de reducción. Aún no hemos definido las reglas de reducción, sin embargo, veremos un ejemplo que ayuda a decidir la estrategia a utilizar.
Sea $\mathsf{xor} = \lambda x.\lambda y.\mathsf{isZ}(x)?(\mathsf{isZ}(y)?0:1):y$. Luego, definimos el siguiente término:

$$t = (\lambda x.\mathsf{xor}\ xx)(\mathsf{meas}(H(\mathsf{new}\ 0)))$$

Call-by-value. En call-by-value la reducción es la siguiente

$$\begin{aligned}
[|\rangle, t] &\to_{CBV} [|0\rangle, (\lambda x.\mathsf{xor} xx)(\mathsf{meas}(Hp_1))] \\
&\to_{CBV} [\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), (\lambda x.\mathsf{xor}\ xx)(\mathsf{meas}\ p_1)] \\
&\to_{CBV} \begin{cases} [|0\rangle, (\lambda x.\mathsf{xor}\ xx)0] \\ [|1\rangle, (\lambda x.\mathsf{xor}\ xx)1] \end{cases} \\
&\to_{CBV} \begin{cases} [|0\rangle, \mathsf{xor}\ 0\ 0] \\ [|1\rangle, \mathsf{xor}\ 1\ 1] \end{cases} \\
&\to_{CBV} \begin{cases} [|0\rangle, 0] \\ [|1\rangle, 0] \end{cases}
\end{aligned}$$

Donde las dos ramas tienen probabilidad 1/2 cada una, por lo tanto, en call-by-value este programa produce el valor booleano 0 con probabilidad 1.

Call-by-name. En call-by-name, en cambio, la reducción es la siguiente:

$$[|\rangle, t] \to_{CBN} [|\rangle, \mathsf{xor}\ (\mathsf{meas}(H(\mathsf{new}\ 0)))\ (\mathsf{meas}(H(\mathsf{new}\ 0)))] \to^*_{CBN} \begin{cases} [|01\rangle, 1] \\ [|10\rangle, 1] \\ [|00\rangle, 0] \\ [|11\rangle, 0] \end{cases}$$

(No se detalla paso a paso ya que hay algunas sutilezas con la construcción $\mathsf{isZ}()?:$ que aún no hemos mencionado).
Entonces, en call-by-name este programa produce 0 o 1 con la misma probabilidad.

Sin estrategia. Si no se establece una estrategia, este término podría incluso reducir a un término mal formado, por ejemplo:

$$\begin{aligned} [|\rangle, t] &\to_{CBV} [|0\rangle, (\lambda x.\mathsf{xor} xx)(\mathsf{meas}(Hp_1))] \\ &\to_{CBV} [\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), (\lambda x.\mathsf{xor}\ xx)(\mathsf{meas}\ p_1)] \\ &\to_{CBN} [\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \mathsf{xor}\ (\mathsf{meas}\ p_1)\ (\mathsf{meas}\ p_1)] \end{aligned}$$

Notar que el último término no es válido ya que contiene dos ocurrencias de p_1.

Conclusión: Se utiliza la estrategia call-by-value, ya que es la más natural.

Reescritura probabilista. Como se vio en el ejemplo anterior, es necesario utilizar un sistema de reescritura probabilista. Esto es, un sistema de reescritura donde algunos términos pueden reducir en más de una forma, cada una de ellas con una probabilidad asociada. No nos detendremos más a formalizar ésto, basta decir que todas las reglas de reducción tendrán probabilidad 1, a excepción de meas que tendrá diferentes formas de reducirlo con una probabilidad dada por el vector de estado cuántico del programa.
Por ejemplo,

$$[\alpha|0\rangle + \beta|1\rangle, \mathsf{meas}\ p_1] \to_{|\alpha|^2} [|0\rangle, 0] \quad \text{y} \quad [\alpha|0\rangle + \beta|1\rangle, \mathsf{meas}\ p_1] \to_{|\beta|^2} [|1\rangle, 1]$$

donde $t \to_p r$ se lee "t reduce a r con probabilidad p".

Semántica operacional: formalización

Dado que se ha elegido una estrategia call-by-value, definimos los valores de la siguiente manera:

$$v ::= x \mid \lambda x.t \mid 0 \mid 1 \mid \mathsf{meas} \mid \mathsf{new} \mid U \mid * \mid (v, v)$$

El conjunto de estados de valores es $\mathbb{V} = \{[q, \ell, v]\}$.

Las reglas de reducción se dan a continuación, donde se ha utilizado la convención para simplificar la notación sin la función de linkeado.

$$
\begin{aligned}
[q, (\lambda x.t)v] &\to_1 [q, t[v/x]] \\
[q, \mathsf{isZ}(1)?t:r] &\to_1 [q, t] \\
[q, \mathsf{isZ}(0)?t:r] &\to_1 [q, r] \\
[q, U(p_{j_1}, \ldots, p_{j_n})] &\to_1 [q', (p_{j_1}, \ldots, p_{j_n})] && (*) \\
[\alpha|q_0\rangle + \beta|q_1\rangle, \mathsf{meas}\ p_i] &\to_{|\alpha|^2} [|q_0\rangle, 0] && (**) \\
[\alpha|q_0\rangle + \beta|q_1\rangle, \mathsf{meas}\ p_i] &\to_{|\beta|^2} [|q_1\rangle, 1] && (**) \\
[q, \mathsf{new}\ 0] &\to_1 [q \otimes |0\rangle, p_{n+1}] && (***) \\
[q, \mathsf{new}\ 1] &\to_1 [q \otimes |1\rangle, p_{n+1}] && (***) \\
[q, \mathsf{let}\ (x_1, x_2) = (v_1, v_2)\ \mathsf{in}\ t] &\to_1 [q, t[v_1/x_1][v_2/x_2]]
\end{aligned}
$$

Para las siguientes reglas, sea $[q, t] \to_p [q', t']$. Entonces:

$$
\begin{aligned}
[q, rt] &\to_p [q', rt'] \\
[q, tv] &\to_p [q', t'v] \\
[q, (t, r)] &\to_p [q', (t', r)] \\
[q, (v, t)] &\to_p [q', (v, t')] \\
[q, \mathsf{isZ}(t)?r:s] &\to_p [q', \mathsf{isZ}(t')?r:s] \\
[q, \mathsf{let}\ (x, y) = t\ \mathsf{in}\ r] &\to [q', \mathsf{let}\ (x, y) = t'\ \mathsf{in}\ r] \\
[q, \mathsf{let}\ (x, y) = r\ \mathsf{in}\ t] &\to [q', \mathsf{let}\ (x, y) = r\ \mathsf{in}\ t']
\end{aligned}
$$

(∗) U es una compuerta cuántica de n-qubits y q' es q luego de aplicar la compuerta a los qubits $j_1, \ldots, j_n$.

(∗∗) $|q_0\rangle$ y $|q_1\rangle$ son qubits normalizados de la forma

$$|q_0\rangle = \sum_j \alpha_j |\phi_j^0\rangle \otimes |0\rangle \otimes |\psi_j^0\rangle \qquad y \qquad |q_1\rangle = \sum_j \alpha_j |\phi_j^1\rangle \otimes |1\rangle \otimes |\psi_j^1\rangle$$

donde $|\phi_j^0\rangle, |\phi_j^1\rangle \in \mathbb{C}^{2^{i-1}}$.

(∗ ∗ ∗) $q \in \mathbb{C}^{2^n}$

Tipos

El sistema de tipos captura la noción de duplicabilidad, como se discutió anteriormente. Se utiliza la notación de la lógica lineal de Girard [1987]. Un término de tipo A se asume no duplicable, y a los términos duplicables se les asignará tipos de la forma $!A$. La gramática de los tipos se define como sigue:

$$A ::= \alpha \mid X \mid\ !A \mid A \multimap A \mid \top \mid A \otimes A$$

donde α es alguno de un conjunto de constantes y X es alguno de un conjunto de variables de tipo.

Escribimos $!^n A$ en lugar de $!!\ldots!!A$ con n repeticiones de !, y A^n para el producto tensorial de n As: $A \otimes \cdots \otimes A$.

En general, un valor de tipo $!A$ podremos utilizarlo más de una vez. Pero no hay ningún problema si decimos que ese valor tiene también tipo A, y en ese caso debe usarse sólo una vez. Por lo tanto, se definen reglas de subtipado, que nos permitirán hacer que si un término tiene un tipo, también tenga cualquier subtipo de éste.

$$\frac{}{\alpha <: \alpha}\ (\alpha) \qquad \frac{}{X <: X}\ (X) \qquad \frac{}{\top <: \top}\ (\top) \qquad \frac{A <: B}{!A <: B}\ (D) \qquad \frac{!A <: B}{!A <: !B}\ (!)$$

$$\frac{A_1 <: B_1 \quad A_2 <: B_2}{A_1 \otimes A_2 <: B_1 \otimes B_2}\ (\otimes) \qquad \frac{A <: A' \quad B <: B'}{A' \multimap B <: A \multimap B'}\ (\multimap)$$

Que un programa $[q, \ell, t]$ esté bien tipado, significa simplemente que t esté bien tipado, por lo tanto, vamos a dar las reglas de tipos sólo para t.
Para cada constante c del lenguaje se asocia un tipo fijo A_c:

$$\begin{array}{lllllllll} A_0 & = & !\mathsf{bit} & A_{\mathsf{new}} & = & !(\mathsf{bit} \multimap \mathsf{qbit}) & & & \\ A_1 & = & !\mathsf{bit} & A_{\mathsf{meas}} & = & !(\mathsf{qbit} \multimap \mathsf{bit}) & A_U & = & !(\mathsf{qbit}^n \multimap \mathsf{qbit}^n) \end{array}$$

$$\frac{A <: B}{\Gamma, x : A \vdash x : B}\ (var) \qquad \frac{A_c <: B}{\Gamma \vdash c : B}\ (const)$$

$$\frac{\Gamma_1, !\Delta \vdash t : \mathsf{bit} \quad \Gamma_2, !\Delta \vdash r : A \quad \Gamma_2, !\Delta \vdash s : A}{\Gamma_1, \Gamma_2, !\Delta \vdash \mathsf{isZ}(t)?r\,:\,s : A}\ (if)$$

$$\frac{\Gamma_1, !\Delta \vdash t : A \multimap B \quad \Gamma_2, !\Delta \vdash r : A}{\Gamma_1, \Gamma_2, !\Delta \vdash tr : B}\ (app)$$

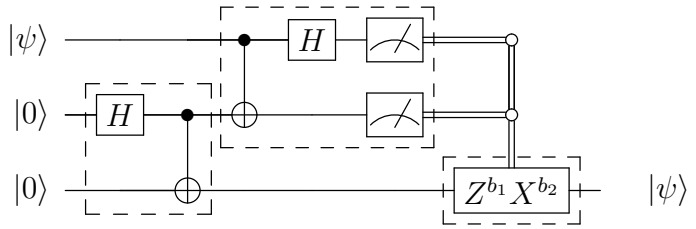
$$\frac{x : A, \Delta \vdash t : B}{\Delta \vdash \lambda x.t : A \multimap B}\ (\lambda_1) \qquad \text{Si } \mathsf{FV}(t) \cap |\Gamma| = \emptyset,\ \frac{\Gamma, !\Delta, x : A \vdash t : B}{\Gamma, !\Delta \vdash \lambda x.t :!^{n+1}(A \multimap B)}\ (\lambda_2)$$

$$\frac{\Gamma_1, !\Delta \vdash t :!^n A_1 \quad \Gamma_2, !\Delta \vdash r :!^n A_2}{\Gamma_1, \Gamma_2, !\Delta \vdash (t, r) :!^n (A_1 \otimes A_2)}\ (\otimes_i) \qquad \frac{}{\Delta \vdash * :!^n \top}\ (\top)$$

$$\frac{\Gamma_1, !\Delta \vdash t :!^n (A_1 \otimes A_2) \quad \Gamma_2, !\Delta, x :!^n A_1, y :!^n A_2 \vdash r : A}{\Gamma_1, \Gamma_2, !\Delta \vdash \mathsf{let}\ (x, y) = t\ \mathsf{in}\ r : A}\ (\otimes_e)$$

Ejemplo: Teleportación

La teleportación fue presentada en la Sección 1.6.2. Reproducimos el circuito aquí por conveniencia.



Las líneas punteadas delimitan tres partes del circuito: la primera es la creación del estado de Bell β_{00}, y le llamaremos Bell. La segunda es las operaciones que realiza Alice,

y llamaremos a esta parte del algoritmo Alice. Finalmente, la tercera es la que realiza Bob, por lo que le llamaremos Bob. Los tipos de cada parte del programa serán:

$$\vdash \mathsf{Bell} :! (\top \mapsto (\mathsf{qbit} \otimes \mathsf{qbit})$$
$$\vdash \mathsf{Alice} :! (\mathsf{qbit} \multimap \mathsf{qbit} \multimap \mathsf{bit} \otimes \mathsf{bit})$$
$$\vdash \mathsf{Bob} :! (\mathsf{qbit} \multimap \mathsf{bit} \otimes \mathsf{bit} \multimap \mathsf{qbit})$$

Esas funciones se definen por

$$\begin{aligned} \mathsf{Bell} &= \lambda x.CNOT(H(\mathsf{new}\ 0), \mathsf{new}\ 0) \\ \mathsf{Alice} &= \lambda q_1.\lambda q_2.(\mathsf{let}\ (p, p') = CNOT(q_1, q_2)\ \mathsf{in}\ (\mathsf{meas}(H\ p), \mathsf{meas}\ p')) \\ \mathsf{Bob} &= \lambda q.\lambda(x, y).\mathsf{isZ}(x)?(\mathsf{isZ}(y)?ZXq : Zq) : (\mathsf{isZ}(y)?Xq : q) \end{aligned}$$

Luego,

$$\mathsf{Telep} = \lambda q.(\mathsf{let}\ (p, p') = \mathsf{Bell}\ *\ \mathsf{in}\ (\mathsf{Bob}\ p'\ (\mathsf{Alice}\ q\ p)))$$

Ejercicios:

1. Dar la traza de $[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \mathsf{Telep}\ p_1]$.

2. Dar términos para Deutsch y la codificación superdensa, y tiparlos.

5.2. Control y datos cuánticos

Antes del cálculo de Selinger y Valiron, y de la idea de control clásico y datos cuánticos, hubo otras extensiones al lambda cálculo. La más notoria, quizá, es el cálculo de van Tonder [2004] del cual veremos algunos detalles en la Sección 5.2.1. Sin embargo, luego de la introducción del paradigma de control clásico y datos cuánticos también hubo desarrollos en el paradigma de control y datos cuánticos. En particular, dicho paradigma puede ser más apropiado para estudiar la computación cuántica desde un punto de vista lógico a través del isomorfismo de Curry-Howard (ver [Sørensen y Urzyczyn, 2006]). El cálculo lineal-algebraico (*Lineal*) de Arrighi y Dowek [2008, 2017] dio origen al estudio de los lenguajes cuánticos a nivel de su semántica operacional y lo veremos en la Sección 5.2.2, así como su versión tipada, del trabajo de Arrighi, Díaz-Caro, y Valiron [2017]. Luego veremos una modificación a dicho cálculo, agregando medición y un sistema de tipos simples en la Sección 5.2.3, donde se describe el paper de Díaz-Caro y Dowek [2017] y se da una intuición de su semántica denotacional.

5.2.1. El cálculo de van Tonder

Al igual que con el cálculo de Selinger y Valiron, van Tonder también utiliza un sistema de tipos lineal para evitar el clonado. Sin embargo, el cálculo de van Tonder estudia otra propiedad interesante: la reversibilidad. Efectivamente, la computación cuántica, excluyendo la medición, es reversible, ya que las compuertas cuánticas son unitarias.

Inicialmente van Tonder plantea una gramática simple, con constantes para las operaciones cuánticas

$$t ::= x \mid \lambda x.t \mid tt \mid c$$
$$c ::= 0 \mid 1 \mid H \mid CNOT \mid X \mid Z \mid \dots$$

donde, como puede observarse, no hay medición.
La computación reversible fue estudiada en los 70s, en particular, el trabajo de Bennett [1973] mostró una manera simple de obtener reversibilidad: llevar un historial de los pasos de reducción. Por ejemplo, si $t_0 \to t_1 \to t_2 \to \dots$, la reducción reversible sería

$$(t_0) \to (t_0, t_1) \to (t_0, t_1, t_2) \to \dots$$

Sin embargo, en el caso cuántico, no es tan directo. Supongamos que el estado inicial viene dado por H aplicada al qubit $|0\rangle$, que van Tonder lo representa dentro de un ket de la siguiente manera:

$$|(H\ 0)\rangle$$

Las reglas de reducción para este término deberían ser tales que $|(H\ 0)\rangle$ reduzca a $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ y $|(H\ 1)\rangle$ a $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Sin embargo, estas reglas no son reversibles. Usando el truco de Bennett podríamos tener

$$|(H\ 0)\rangle \to \frac{1}{\sqrt{2}}(|(H\ 0); 0\rangle + |(H\ 0); 1\rangle) = |(H\ 0)\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

En este ejemplo, el "historial" se factoriza a la izquierda y el término reducido queda a la derecha. Sin embargo, consideremos el siguiente ejemplo:

$$\begin{aligned} |(H\ (H\ 0))\rangle &\to \frac{1}{\sqrt{2}}(|(H\ (H\ 0)); (H\ 0)\rangle + |(H\ (H\ 0)); (H\ 1)\rangle) \\ &\to \frac{1}{2}|(H\ (H\ 0))\rangle \otimes (|(H\ 0); 0\rangle + |(H\ 0); 1\rangle + |(H\ 1); 0\rangle - |(H\ 1); 1\rangle) \end{aligned}$$

Aquí el término reducido no puede ser factorizado: el registro quedo enredado con parte del historial.
Pero en este ejemplo vemos que se está guardando más información de la necesaria. Con guardar simplemente que subtérmino redujo y con qué operación es suficiente. Retomando el mismo ejemplo, tendríamos:

$$\begin{aligned} |(H\ (H\ 0))\rangle &\to \frac{1}{\sqrt{2}}(|(\_\ (H\ \_)); (H\ 0)\rangle + |(\_\ (H\ \_)); (H\ 1)\rangle) \\ &\to \frac{1}{2}|(\_\ (H\ \_))\rangle \otimes (|(H\ \_); 0\rangle + |(H\ \_); 1\rangle + |(H\ \_); 0\rangle - |(H\ \_); 1\rangle) \\ &= |(\_\ (H\ \_)); (H\ \_)\rangle \otimes |0\rangle \end{aligned}$$

Con estas ideas en mente se define el modelo computacional. El estado computacional se toma como una superposición cuántica de secuencias

$$h_1; \dots; h_n; t$$

donde $h_1; \dots; h_n$ es llamado historial, y t registro computacional.
No vamos a discutir el cálculo de van Tonder en este curso, sólo estas observaciones y se recomienda al interesado el paper [van Tonder, 2004].

5.2.2. El lambda cálculo lineal algebraico

El lambda cálculo lineal algebraico (de ahora en más, Lineal) sigue un paradigma diferente a los dos vistos anteriormente: la idea es tener un lambda cálculo puro, donde no hay distinción entre datos y programas, y, por lo tanto, dado que los datos pueden superponerse, también pueden hacerlo los programas.
Este cálculo es un primer paso hacia un cálculo cuántico. En este primer paso el cálculo se centra en la noción computacional de espacios vectoriales.

Gramática

La gramática del lenguaje incluye todos los términos de lambda cálculo, y sus combinaciones lineales.

$$t ::= x \mid \lambda x.t \mid tr \mid 0 \mid \alpha.t \mid t + t$$

donde $\alpha \in \mathbb{C}$. Aquí, la constante 0 tiene un significado diferente a la de los cálculos anteriores: recordemos que Lineal no es un cálculo cuántico, sino un cálculo *vectorial*, y por lo tanto 0 representa simplemente al vector nulo.
Los valores de este cálculo son las variables y las abstracciones, y los escribimos utilizando la letra v.

Semántica operacional

Las reglas de reducción son, además de la beta reducción, una versión orientada de los axiomas de espacios vectoriales. La orientación de cada regla ha sido elegida de manera de obtener un cálculo confluente.
La idea principal es que una abstracción, sobre una combinación lineal, se comporte linealmente como lo hace una matriz sobre un vector. Así,

$$(\lambda x.t)(\alpha.r + \beta.s) \to^* \alpha.(\lambda x.t)r + \beta.(\lambda x.t)s$$

De esa manera no se impone no clonado con una restricción de lógica lineal (ver el cálculo de Selinger y Valiron en la Sección 5.1.1), sino que siempre que haya una superposición, la función actuará linealmente, y por lo tanto sólo actuará en los vectores de base. Recordemos que los vectores de base son clonables: por ejemplo, la compuerta CNOT clona los qubits $|0\rangle$ y $|1\rangle$:

$$\begin{aligned} CNOT|00\rangle &= |00\rangle \\ CNOT|10\rangle &= |11\rangle \end{aligned}$$

En cambio, la misma función aplicada a una superposición, actúa de la siguiente manera:

$$\begin{aligned} CNOT(\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle &= CNOT(\alpha|00\rangle + \beta|10\rangle) \\ &= \alpha CNOT|00\rangle + \beta CNOT|10\rangle \\ &= \alpha|00\rangle + \beta|11\rangle \end{aligned}$$

Notar que ese qubit difiere de $(\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle)$.

Por lo tanto el cálculo deberá ser call-by-value, ya que si el argumento reduce a una superposición, es necesario reducirlo antes de pasarlo a una abstracción.

β-reducción		Reglas de factorización	
$(\lambda x.t)v$	$\to t[v/x]$	$\alpha.t + \beta.t$	$\to (\alpha+\beta).t$
Reglas elementales		$\alpha.t + t$	$\to (\alpha+1).t$
$0.t$	$\to 0$	$t + t$	$\to 2.t$
$1.t$	$\to t$	$t + 0$	$\to 0$
α,0	$\to 0$	Reglas de aplicación	
$\alpha.(\beta$,0)	$\to (\alpha \times \beta)$,0	$(t+r)s$	$\to ts + rs$
$\alpha.(t+r)$	$\to \alpha.t + \alpha.r$	$t(r+s)$	$\to tr + ts$
		$(\alpha.t)r$	$\to \alpha.tr$
		$t(\alpha.r)$	$\to \alpha.tr$
		$0t$	$\to 0$
		$t0$	$\to 0$

Si $t \to r$ entonces $C[t] \to C[r]$ para cualquier contexto $C[\cdot]$

Ejemplo 5.3. Consideremos los términos true y false de Church, $\lambda x.\lambda y.x$ y $\lambda x.\lambda y.y$ como los qubits $|0\rangle$ y $|1\rangle$. De esa manera podemos codificar el término $\mathsf{isZ}(t)?r:s$ simplemente como trs, ya que si t es $|0\rangle$, $\lambda x.\lambda y.xrs \to^* r$ y si t es $|1\rangle$, $\lambda x.\lambda y.yrs \to^* s$.
La compuerta Hadamard podría ser codificada en Lineal de la siguiente manera:

$$Hx = x(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle))$$

Sin embargo, de esta manera trivial no funciona, ya que los mecanismos utilizados para no-clonado (las reglas de aplicación), harán que el término reduzca de la siguiente manera:

$$\begin{aligned}
H|0\rangle &= H\lambda x.\lambda y.x \\
&= \lambda x.\lambda y.x(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) \\
&\to \frac{1}{\sqrt{2}}\lambda x.\lambda y.x(|0\rangle + |1\rangle)(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) \\
&\to \frac{1}{\sqrt{2}}(\lambda x.\lambda y.x|0\rangle + \lambda x.\lambda y.x|1\rangle)(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)) \\
&\to^* \frac{1}{2}(\lambda y.|0\rangle + \lambda y.|1\rangle)(|0\rangle - |1\rangle) \\
&\to^* \frac{1}{2}(|0\rangle - |0\rangle + |1\rangle - |1\rangle) \\
&\to^* 0
\end{aligned}$$

El problema está en la linealidad. Lo que debemos hacer es que el encodaje del $\mathsf{isZ}()?:$ detenga la linealidad. Por ejemplo:

$$\mathsf{isZ}(t)?r:s = (t(\lambda x.r)(\lambda x.s))(\lambda x.x)$$

Para hacerlo más legible, usamos la siguiente notación:

$$\begin{aligned} [t] &= \lambda x.t \qquad \text{con } x \notin \mathsf{FV}(t) \\ t &= t(\lambda x.x) \end{aligned}$$

Y entonces, el encodaje del $\mathsf{isZ}()?:$ es el siguiente:

$$\mathsf{isZ}(t)?r:s = \{t[r][s]\}$$

Por ejemplo, a la compuerta Hadamard la codificamos correctamente como

$$\{Hx\} = \{x[(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))][(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle))]\}$$

Tipos

El sistema de tipos vectorial para Lineal [Arrighi, Díaz-Caro, y Valiron, 2017] propone que los tipos lleven cuenta de las superposiciones en los términos. Así, si el término t tiene tipo A y el término r tiene tipo B, el término $\alpha.t + \beta.r$ tendrá tipo $\alpha.A + \beta.B$. Si consideramos $T = X \Rightarrow Y \Rightarrow X$ y $F = X \Rightarrow Y \Rightarrow Y$, el término $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ tendrá tipo $\frac{1}{\sqrt{2}}(T + F)$. Notar que este tipo tiene norma 1, al igual que el término, y por lo tanto, un tal tipo nos permite verificar facilmente la norma del vector que produce un programa.

Dada la estrategia call-by-value, las variables sólo pueden tener tipos no superpuestos, los que llamamos *unit types*. Esta necesidad se comprende mejor con el siguiente ejemplo: Supongamos que permitimos variables con tipos escalados, como $\alpha.V$. Entonces, el término $\lambda x.x + t$ podría tener tipo $(\alpha.V) \Rightarrow \alpha.V + U$, con t de tipo U. Luego, tomemos un término v de tipo V y tenemos $(\lambda x.x + t)(\alpha.v)$ de tipo $\alpha.V + U$. Sin embargo

$$(\lambda x.x + t)(\alpha.v) \to \alpha.(\lambda x.x + t)v \to \alpha.(v + t) \to \alpha.v + \alpha.t$$

lo cual es problemático, ya que $\alpha.V + U$ no refleja esta superposición.

Por el contrario, las variables de tipo no necesitan ser solo unit. Sin embargo, debemos distinguir variables unit de las que no lo son, ya que sólo las variables unit pueden aparecer a la izquierda de una flecha. Por ese motivo definimos dos tipos de variables: las variables $\mathcal{X}$, que sólo podrán ser reemplazadas por tipos unit, y las variables $\mathbb{X}$, que pueden ser reemplazadas por cualquier tipo (escribimos simplemente X cuando nos refiramos a cualquiera de las dos). El tipo $\mathcal{X}$ es unit, mientras el tipo $\mathbb{X}$ no lo es.

En particular, $T = \forall\mathcal{X}.\forall\mathcal{Y}.X \Rightarrow Y \Rightarrow X$ y $F = \forall\mathcal{X}.\forall\mathcal{Y}.X \Rightarrow Y \Rightarrow Y$. El tipo de H es

$$\forall\mathbb{X}.\left([\frac{1}{\sqrt{2}}(T + F)] \Rightarrow [\frac{1}{\sqrt{2}}(T + F)] \Rightarrow \mathbb{X}\right) \Rightarrow \mathbb{X}$$

Donde $[A] = (\forall\mathcal{X}.\mathcal{X} \Rightarrow \mathcal{X}) \Rightarrow A$.

Gramática de los tipos. Formalizando lo anterior, la gramática de los tipos es la siguiente:

$$A ::= U \mid \alpha.A \mid A + A \mid \mathbb{X}$$
$$U ::= \mathcal{X} \mid U \Rightarrow A \mid \forall \mathcal{X}.U \mid \forall \mathbb{X}.U$$

Consideramos también las siguientes equivalencias entre tipos, la cual es una congruencia:

$$\begin{array}{rclcrcl} 1.A & \equiv & A & & \alpha.A + \beta.A & \equiv & (\alpha+\beta).A \\ \alpha.(\beta.A) & \equiv & (\alpha\times\beta).A & & A+B & \equiv & B+A \\ \alpha.A+\alpha.B & \equiv & \alpha.(A+B) & & A+(B+C) & \equiv & (A+B)+C \end{array}$$

Observación. Por una cuestión técnica, la equivalencia $A + 0.B \equiv A$ no es válida, y por lo tanto la propiedad de subject reduction es más débil, en el sentido de que si $\Gamma \vdash t : A$ y $t \to_p r$, sólo se puede asegurar que $\Gamma \vdash r : B$ donde si la regla por la cual se redujo t a r no es una regla de factorización, entonces $B = A$, en otro caso, A puede ser $B + 0.C$. De todas maneras, obviaremos este detalle técnico en este apunte, y se refiere al lector a [Arrighi, Díaz-Caro, y Valiron, 2017, §4.2] para más detalles.

Reglas de tipado. Las reglas de tipado se detallan a continuación. Usamos $[T/X]$ para referir a $[U/\mathcal{X}]$ o $[A/\mathbb{X}]$.

$$\frac{}{\Gamma, x : U \vdash x : U}\ ax \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash 0 : 0.A}\ 0_I \qquad \frac{\Gamma, x : U \vdash t : A}{\Gamma \vdash \lambda x.t : U \Rightarrow A}\ \Rightarrow_I$$

$$\frac{\Gamma \vdash t : \sum_{i=1}^n \alpha_i.\forall X.(U \Rightarrow A_i) \quad \Gamma \vdash r : \sum_{j=1}^m \beta_j.U[T/X]}{\Gamma \vdash tr : \sum_{i=1}^n \sum_{j=1}^m \alpha_i \times \beta_j.A_i[T_j/X]}\ \Rightarrow_E$$

$$\frac{\Gamma \vdash t : \sum_{i=1}^n \alpha_i.U_i \quad X \notin \mathsf{FV}(\Gamma)}{\Gamma \vdash t : \sum_{i=1}^n \alpha_i.\forall X.U_i}\ \forall_I \qquad \frac{\Gamma \vdash t : \sum_{i=1}^n \alpha_i.\forall X.U_i}{\Gamma \vdash t : \sum_{i=1}^n \alpha_i.U_i[T/X]}\ \forall_E$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \alpha.t : \alpha.A}\ \alpha_I \qquad \frac{\Gamma \vdash t : A \quad \Gamma \vdash r : B}{\Gamma \vdash t + r : A + B}\ +_I \qquad \frac{\Gamma \vdash t : A \quad A \equiv B}{\Gamma \vdash t : B}\ \equiv$$

Ejemplo 5.4 (Tipando Hadamard). Sean $|0\rangle = \lambda x.\lambda y.x$ y $|1\rangle = \lambda x.\lambda y.y$. Es fácil verificar que

$$\vdash |0\rangle : \forall \mathcal{X}\mathcal{Y}.\mathcal{X} \Rightarrow \mathcal{Y} \Rightarrow \mathcal{X},$$
$$\vdash |1\rangle : \forall \mathcal{X}\mathcal{Y}.\mathcal{X} \Rightarrow \mathcal{Y} \Rightarrow \mathcal{Y}.$$

Observación. Usamos la notación $\forall XY.A$ para $\forall X.\forall Y.A$.

También definimos las siguientes superposiciones:

$$|+\rangle = \frac{1}{\sqrt{2}}.(|0\rangle + |1\rangle) \qquad \text{y} \qquad |-\rangle = \frac{1}{\sqrt{2}}.(|0\rangle - |1\rangle).$$

De la misma manera,definimos

$$\boxplus = \frac{1}{\sqrt{2}}.((\forall \mathcal{X}\mathcal{Y}.\mathcal{X} \Rightarrow \mathcal{Y} \Rightarrow \mathcal{X}) + (\forall \mathcal{X}\mathcal{Y}.\mathcal{X} \Rightarrow \mathcal{Y} \Rightarrow \mathcal{Y})),$$
$$\boxminus = \frac{1}{\sqrt{2}}.((\forall \mathcal{X}\mathcal{Y}.\mathcal{X} \Rightarrow \mathcal{Y} \Rightarrow \mathcal{X}) - (\forall \mathcal{X}\mathcal{Y}.\mathcal{X} \Rightarrow \mathcal{Y} \Rightarrow \mathcal{Y})).$$

Es fácil verificar que $\vdash [|+\rangle] : [\boxplus]$ y $\vdash [|-\rangle] : [\boxminus]$.
Para simplificar la notación, tomamos $A = [\boxplus] \Rightarrow [\boxminus] \Rightarrow [\mathbb{X}]$. Entonces

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}\, ax \quad x : A \vdash [|+\rangle] : [\boxplus]}{x : A \vdash x[|+\rangle] : [\boxminus] \Rightarrow [\mathbb{X}]}\Rightarrow_E \quad x : A \vdash [|-\rangle] : [\boxminus]}{x : A \vdash x[|+\rangle][|-\rangle] : [\mathbb{X}]}\Rightarrow_E}{x : A \vdash \{x[|+\rangle][|-\rangle]\} : \mathbb{X}}\Rightarrow_E}{\vdash \lambda x.\{x[|+\rangle][|-\rangle]\} : A \Rightarrow \mathbb{X}}\Rightarrow_I}{\vdash \lambda x.\{x[|+\rangle][|-\rangle]\} : \forall \mathbb{X}.([\boxplus] \Rightarrow [\boxminus] \Rightarrow [\mathbb{X}]) \Rightarrow \mathbb{X}}\,\forall_I$$

Ahora podemos aplicar Hadamard a un qubit para obtener el tipo correcto. Sea $H = \lambda x.\{x[|+\rangle][|-\rangle]\}$.

$$\dfrac{\dfrac{\vdash H : \forall \mathbb{X}.([\boxplus] \Rightarrow [\boxminus] \Rightarrow [\mathbb{X}]) \Rightarrow \mathbb{X}}{\vdash H : ([\boxplus] \Rightarrow [\boxminus] \Rightarrow [\boxplus]) \Rightarrow \boxplus}\,\forall_E \quad \dfrac{\dfrac{\vdash |0\rangle : \forall \mathcal{X}\mathcal{Y}.\mathcal{X} \Rightarrow \mathcal{Y} \Rightarrow \mathcal{X}}{\vdash |0\rangle : \forall \mathcal{Y}.[\boxplus] \Rightarrow \mathcal{Y} \Rightarrow [\boxplus]}\,\forall_E}{\vdash |0\rangle : [\boxplus] \Rightarrow [\boxminus] \Rightarrow [\boxplus]}\,\forall_E}{\vdash H|0\rangle : \boxplus}\Rightarrow_E$$

Un ejemplo aún más interesante es el siguiente. Sea

$$\boxplus_I = \frac{1}{\sqrt{2}}.(([\boxplus] \Rightarrow [\boxminus] \Rightarrow [\boxplus]) + ([\boxplus] \Rightarrow [\boxminus] \Rightarrow [\boxminus]))$$

Es decir, $\boxplus_I$ es $\boxplus$ donde los forall han sido instanciados. Es fácil verificar que $\vdash |+\rangle : \boxplus_I$. Entonces,

$$\frac{\vdash H : \forall \mathbb{X}.([\boxplus] \Rightarrow [\boxminus] \Rightarrow [\mathbb{X}]) \Rightarrow \mathbb{X} \quad \vdash |+\rangle : \boxplus_I}{\vdash H|+\rangle : \frac{1}{\sqrt{2}}.\boxplus + \frac{1}{\sqrt{2}}.\boxminus}\Rightarrow_E$$

Y dado que $\frac{1}{\sqrt{2}}.\boxplus + \frac{1}{\sqrt{2}}.\boxminus \equiv \forall \mathcal{X}\mathcal{Y}.\mathcal{X} \to \mathcal{Y} \to \mathcal{X}$, podemos concluir

$$\vdash H|+\rangle : \forall \mathcal{X}\mathcal{Y}.\mathcal{X} \Rightarrow \mathcal{Y} \Rightarrow \mathcal{X}.$$

Notar que $H|+\rangle \to^* |0\rangle$.

Ejercicio: Escribir el algoritmo de teleportación en Lineal no tipado.

5.2.3. Tipando superposiciones y mediciones proyectivas

En esta sección veremos un trabajo muy reciente [Díaz-Caro y Dowek, 2017; Rinaldi, 2018].
El principal objetivo es agregar medición a Lineal, el cálculo presentado en la Sección 5.2.2. Como se mencionó anteriormente, para evitar el clonado se conocen dos técnicas:

(LL) Usar términos lineales en el sentido de la lógica lineal, y entonces $\lambda x.x \otimes x$ es un término mal formado.

(AL) Usar un sistema de reescritura que defina las aplicaciones como aplicaciones lineales, así $\lambda x.x \otimes x$ es permitido, pero al aplicarlo a $\alpha.|0\rangle + \beta.|1\rangle$ producirá $\alpha.|00\rangle + \beta.|11\rangle$ y no $(\alpha.|0\rangle + \beta.|1\rangle) \otimes (\alpha.|0\rangle + \beta.|1\rangle)$.

Sin embargo, definir aplicaciones lineales por medio de reescritura no funciona si el cálculo tiene medición, ya que sólo las compuertas cuánticas se comportan de esa manera. Por ejemplo, digamos que tenemos un operador de medición notado por π, entonces

$$(\lambda x.\pi x)(\alpha.|0\rangle + \beta.|1\rangle) \to^* \alpha.(\lambda x.\pi x)|0\rangle + \beta.(\lambda x.\pi x)|1\rangle \to^* \alpha.|0\rangle + \beta.|1\rangle$$

lo cual claramente es un error.
En este cálculo se propone usar una combinación de las dos técnicas, LL y AL, de esa manera, una abstracción podrá tomar una superposición, pero sólo en el caso de que la trate linealmente, en el sentido de LL, en otro caso, la aplicación sólo podrá comportarse en el sentido de AL.
Claro que para eso, debemos distinguir términos superpuestos de términos que no lo están.

Gramáticas

La gramática de tipos se separa en dos niveles ya que este cálculo es de primer orden (por razones que luego discutiremos).

$$\begin{array}{ll} \Psi := \mathbb{B}^n \mid S(\Psi) \mid \Psi \times \Psi & \text{Tipos qubit } (\mathcal{Q}) \\ A := \Psi \mid \Psi \Rightarrow A \mid S(A) \mid A \times A & \text{Tipos generales } (\mathcal{T}) \end{array}$$

La gramática de términos es la siguiente:

$$\begin{array}{ll} b := x \mid \lambda x{:}\Psi.t \mid |0\rangle \mid |1\rangle \mid b \times b & \text{Términos de base } (\mathfrak{B}) \\ v := b \mid (v + v) \mid \vec{0}_{S(A)} \mid \alpha.v \mid v \times v & \text{Valores } (\mathcal{V}) \\ t := v \mid tt \mid (t + t) \mid \pi_j t \mid \mathsf{isZ}()?t{:}t \mid \alpha.t \mid t \times t \mid \mathit{head}\ t \mid \mathit{tail}\ t \mid \Uparrow_r t \mid \Uparrow_\ell t & \text{Términos } (\Lambda) \end{array}$$

con $\alpha \in \mathbb{C}$.
Utilizamos la notación $\mathsf{isZ}(t)?r:s$ para $(\mathsf{isZ}()?r:s)t$. El motivo de considerar a $\mathsf{isZ}()?r:s$ como una función es aprovechar la linealidad AL de las funciones, de esa manera,

$$\mathsf{isZ}((\alpha.|0\rangle + \beta.|1\rangle))?r:s \to^* \alpha.\mathsf{isZ}(|0\rangle)?r:s + \beta.\mathsf{isZ}(|1\rangle)?r:s$$

Dentro de la gramática de términos, distinguimos dos subgramáticas, la de los términos de base, y la de los valores, que son combinaciones lineales de términos de base.

Tipos

Dado que la semántica operacional de una aplicación será diferente si el argumento es una superposición o no, y esa información la sabremos utilizando los tipos, debemos dar primero los tipos, y luego la semántica operacional dependiente de ellos.
La lógica de los tipos es que marcamos con una S a las superposiciones, de la misma manera que en lógica lineal se marca con ! a los términos que pueden ser duplicados. En realidad, nuestras superposiciones son exactamente los términos que no pueden ser copiados, y por ese motivo no utilizamos notación de lógica lineal.
Naturalmente existe un subtipado: si un término no es una superposición, es decir, puede ser copiado, también puede ser tratado como una superposición y por lo tanto no ser copiado. Es decir, $\mathbb{B} \preceq S(\mathbb{B})$.
Formalmente, la relación $\preceq$ es un preorden definido por

$$\frac{}{A \preceq S(A)} \qquad \frac{}{S(S(A)) \preceq S(A)}$$

$$\frac{A \preceq B}{\Psi \Rightarrow A \preceq \Psi \Rightarrow B} \qquad \frac{A \preceq B}{S(A) \preceq S(B)} \qquad \frac{A \preceq B}{A \times C \preceq B \times C} \qquad \frac{A \preceq B}{C \times A \preceq C \times B}$$

Observación. Notar que con esta definición, $S(S(A)) \equiv S(A)$.

El sistema de tipos se define a continuación:

$$\frac{}{x : \Psi \vdash x : \Psi}\ Ax \qquad \frac{}{\vdash \vec{0}_{S(A)} : S(A)}\ Ax_{\vec{0}} \qquad \frac{}{\vdash |0\rangle : \mathbb{B}}\ Ax_{|0\rangle} \qquad \frac{}{\vdash |1\rangle : \mathbb{B}}\ Ax_{|1\rangle}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \alpha.t : S(A)}\ S_I^{\alpha} \qquad \frac{\Gamma \vdash t : A \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash (t + u) : S(A)}\ S_I^{+} \qquad \frac{\Gamma \vdash t : S(\mathbb{B}^n)}{\Gamma \vdash \pi_j t : \mathbb{B}^j \times S(\mathbb{B}^{n-j})}\ S_E$$

$$\frac{\Gamma \vdash t : A \ (A \preceq B)}{\Gamma \vdash t : B}\ \preceq \qquad \frac{\Gamma \vdash t : A \quad \Gamma \vdash r : A}{\Gamma \vdash \mathsf{isZ}()?t{:}r : \mathbb{B} \Rightarrow A}\ If \qquad \frac{\Gamma, x : \Psi \vdash t : A}{\Gamma \vdash \lambda x{:}\Psi.t : \Psi \Rightarrow A}\ \Rightarrow_I$$

$$\frac{\Delta \vdash u : \Psi \quad \Gamma \vdash t : \Psi \Rightarrow A}{\Delta, \Gamma \vdash tu : A}\ \Rightarrow_E \qquad \frac{\Delta \vdash u : S(\Psi) \quad \Gamma \vdash t : S(\Psi \Rightarrow A)}{\Delta, \Gamma \vdash tu : S(A)}\ \Rightarrow_{ES}$$

$$\frac{\Gamma \vdash t : A}{\Gamma, x : \mathbb{B}^n \vdash t : A}\ W \qquad \frac{\Gamma, x : \mathbb{B}^n, y : \mathbb{B}^n \vdash t : A}{\Gamma, x : \mathbb{B}^n \vdash (x/y)t : A}\ C$$

$$\frac{\Gamma \vdash t : A \quad \Delta \vdash u : B}{\Gamma, \Delta \vdash t \times u : A \times B}\ \times_I \qquad \frac{\Gamma \vdash t : \mathbb{B}^n \quad n{>}1}{\Gamma \vdash \mathit{head}\ t : \mathbb{B}}\ \times_{Er} \qquad \frac{\Gamma \vdash t : \mathbb{B}^n \quad n{>}1}{\Gamma \vdash \mathit{tail}\ t : \mathbb{B}^{n-1}}\ \times_{El}$$

$$\frac{\Gamma \vdash t : S(S(A) \times B)}{\Gamma \vdash \Uparrow_r t : S(A \times B)}\ \Uparrow_r \qquad \frac{\Gamma \vdash t : S(A \times S(B))}{\Gamma \vdash \Uparrow_\ell t : S(A \times B)}\ \Uparrow_\ell$$

Semántica operacional

Beta	If b has type $\mathbb{B}^n$ and $b \in \mathfrak{B}$, $(\lambda x{:}\mathbb{B}^n.t)b \longrightarrow_{(1)} (b/x)t$	(β_{b})
	If u has type $S(\Psi)$, $(\lambda x{:}S(\Psi).t)u \longrightarrow_{(1)} (u/x)t$	(β_{n})
If	$\mathsf{isZ}(\lvert 1\rangle)?t:r \longrightarrow_{(1)} t$	(if_1)
	$\mathsf{isZ}(\lvert 0\rangle)?t:r \longrightarrow_{(1)} r$	(if_0)
Distribuciones lineales	If t has type $\mathbb{B}^n \Rightarrow A$, $t(u+v) \longrightarrow_{(1)} (tu+tv)$	$(\mathsf{lin}_{\mathsf{r}}^{+})$
	If t has type $\mathbb{B}^n \Rightarrow A$, $(\alpha.u) \longrightarrow_{(1)} \alpha.tu$	$(\mathsf{lin}_{\mathsf{r}}^{\alpha})$
	If t has type $\mathbb{B}^n \Rightarrow A$, $t\vec{0}_{S(\mathbb{B}^n)} \longrightarrow_{(1)} \vec{0}_{S(A)}$	$(\mathsf{lin}_{\mathsf{r}}^{0})$
	$(t+u)v \longrightarrow_{(1)} (tv+uv)$	$(\mathsf{lin}_{\mathsf{l}}^{+})$
	$(\alpha.t)u \longrightarrow_{(1)} \alpha.tu$	$(\mathsf{lin}_{\mathsf{l}}^{\alpha})$
	$\vec{0}_{S(\mathbb{B}^n \Rightarrow A)}t \longrightarrow_{(1)} \vec{0}_{S(A)}$	$(\mathsf{lin}_{\mathsf{l}}^{0})$
Axiomas de espacios vectoriales	$(\vec{0}_{S(A)}+t) \longrightarrow_{(1)} t$	$(\mathsf{neutral})$
	$1.t \longrightarrow_{(1)} t$	(unit)
	If t has type A, $0.t \longrightarrow_{(1)} \vec{0}_{S(A)}$	(zero_{α})
	$\alpha.\vec{0}_{S(A)} \longrightarrow_{(1)} \vec{0}_{S(A)}$	(zero)
	$\alpha.(\beta.t) \longrightarrow_{(1)} (\alpha\beta).t$	(prod)
	$\alpha.(t+u) \longrightarrow_{(1)} (\alpha.t+\alpha.u)$	$(\alpha\mathsf{dist})$
	$(\alpha.t+\beta.t) \longrightarrow_{(1)} (\alpha+\beta).t$	(fact)
	$(\alpha.t+t) \longrightarrow_{(1)} (\alpha+1).t$	(fact^1)
	$(t+t) \longrightarrow_{(1)} 2.t$	(fact^2)
	$\vec{0}_{S(S(A))} \longrightarrow_{(1)} \vec{0}_{S(A)}$	$(\mathsf{zero}_{\mathsf{S}})$
Listas	If $h \neq u \times v$ and $h \in \mathfrak{B}$, $\mathit{head}\ h \times t \longrightarrow_{(1)} h$	(head)
	If $h \neq u \times v$ and $h \in \mathfrak{B}$, $\mathit{tail}\ h \times t \longrightarrow_{(1)} t$	(tail)
Casteo	$\Uparrow_r (r+s) \times u \longrightarrow_{(1)} (\Uparrow_r r \times u + \Uparrow_r s \times u)$	$(\mathsf{dist}_{\mathsf{r}}^{+})$
	$\Uparrow_\ell u \times (r+s) \longrightarrow_{(1)} (\Uparrow_\ell u \times r + \Uparrow_\ell u \times s)$	$(\mathsf{dist}_{\mathsf{l}}^{+})$
	$\Uparrow_r (\alpha.r) \times u \longrightarrow_{(1)} \alpha.\ \Uparrow_r r \times u$	$(\mathsf{dist}_{\mathsf{r}}^{\alpha})$
	$\Uparrow_\ell u \times (\alpha.r) \longrightarrow_{(1)} \alpha.\ \Uparrow_r u \times r$	$(\mathsf{dist}_{\mathsf{l}}^{\alpha})$
	If u has type B, $\Uparrow_r \vec{0}_{S(A)} \times u \longrightarrow_{(1)} \vec{0}_{S(A\times B)}$	$(\mathsf{dist}_{\mathsf{r}}^{0})$
	If u has type A, $\Uparrow_\ell u \times \vec{0}_{S(B)} \longrightarrow_{(1)} \vec{0}_{S(A\times B)}$	$(\mathsf{dist}_{\mathsf{l}}^{0})$
	$\Uparrow (t+u) \longrightarrow_{(1)} (\Uparrow t + \Uparrow u)$	$(\mathsf{dist}_{\Uparrow}^{+})$
	$\Uparrow (\alpha.t) \longrightarrow_{(1)} \alpha.\ \Uparrow t$	$(\mathsf{dist}_{\Uparrow}^{\alpha})$
	If $u \in \mathfrak{B}$, $\Uparrow_r u \times v \longrightarrow_{(1)} u \times v$	$(\mathsf{neut}_{\mathsf{r}}^{\Uparrow})$
	If $v \in \mathfrak{B}$, $\Uparrow_\ell u \times v \longrightarrow_{(1)} u \times v$	$(\mathsf{neut}_{\mathsf{l}}^{\Uparrow})$
Proy.	$\pi_j(\sum_{i=1}^{n}[\alpha_i.]\prod_{h=1}^{m}\lvert b_{hi}\rangle) \longrightarrow_{(1)} \Vert_{k=0}^{2^j-1} p_k(\lvert k\rangle \times \lvert\phi_k\rangle)$	(proj)
Reglas contextuales	If $t \longrightarrow_{(1)} u$, then	
	$tv \longrightarrow_{(1)} uv$ $\quad (\lambda x^{\mathbb{B}^n}.v)t \longrightarrow_{(1)} (\lambda x^{\mathbb{B}^n}.v)u$ $\quad (t+v) \longrightarrow_{(1)} (u+v)$	
	$\alpha.t \longrightarrow_{(1)} \alpha.u$ $\quad \pi_j t \longrightarrow_{(1)} \pi_j u$ $\quad t \times v \longrightarrow_{(1)} u \times v$	
	$v \times t \longrightarrow_{(1)} v \times u$ $\quad \Uparrow_r t \longrightarrow_{(1)} \Uparrow_r u$ $\quad \Uparrow_\ell t \longrightarrow_{(1)} \Uparrow_\ell u$	
	$\mathit{head}\ t \longrightarrow_{(1)} \mathit{head}\ u$ $\quad \mathit{tail}\ t \longrightarrow_{(1)} \mathit{tail}\ u$ $\quad \mathsf{isZ}(t)?r:s \longrightarrow_{(1)} \mathsf{isZ}(u)?r:s$	

donde, en la regla (proj), se tiene:

$$
\begin{aligned}
&j \leq m \\
&|k\rangle = |b_1\rangle \times \cdots \times |b_j\rangle \text{ donde } b_1 \dots b_j \text{ es la representación binaria de } k \\
&|\phi_k\rangle = \sum_{i \in T_k} \left(\frac{\alpha_i}{\sqrt{\sum_{r \in T_k} |\alpha_r|^2}} \right) \prod_{h=j+1}^{m} |b_{hi}\rangle \\
&p_k = \sum_{i \in T_k} \left(\frac{|\alpha_i|^2}{\sum_{r=1}^{n} |\alpha_r|^2} \right) \\
&T_k = \{ i \leq n \mid |b_{1i}\rangle \times \cdots \times |b_{ji}\rangle = |k\rangle \}
\end{aligned}
$$

Primer orden. El motivo de utilizar primer orden es que en este cálculo hemos mezclado los dos enfoques precedentes: LL y AL, y por lo tanto ahora es posible construir una máquina de clonado si se permite alto orden. El truco es esconder dentro de una abstracción una superposición, por ejemplo $\lambda x{:}\mathbb{B}.\alpha.|0\rangle + \beta.|1\rangle$. Éste es un término duplicable, y no hay problema en ello (no es una superposición, es un programa que produce una). Sin embargo, dado que ahora tenemos términos LL, podríamos también producir $\lambda y{:}S(\mathbb{B}).\lambda x{:}\mathbb{B}.y$, el cual nos permite generar superposiciones duplicables. La solución es impedir tomar una abstracción como argumento, y por lo tanto este término no podrá ser duplicado.

Multiple qubits: casteo

Consideremos el siguiente ejemplo:

$$|0\rangle \times (|0\rangle + |1\rangle) \to |0\rangle \times |0\rangle + |0\rangle \times |1\rangle \tag{5.1}$$

El primer término podría ser tipado con $\mathbb{B} \otimes S(\mathbb{B})$, en cambio el segundo debería ser tipado con $S(\mathbb{B} \otimes \mathbb{B})$. Naturalmente el subipado va en sentido contrario al necesario: $\mathbb{B} \otimes S(\mathbb{B}) \preceq S(\mathbb{B} \otimes \mathbb{B})$, y por lo tanto este ejemplo rompe la propiedad de preservación de tipos.
Es normal, en matemática, que al desarrollar un término perdamos información. Por ejemplo, $(x-1)(x-2) = x^2 - 3x + 2$. La información del término izquierdo, que da sus raíces y una factorización, es perdida al desplegar el término. Por este motivo, no permitimos la reducción (5.1). En cambio, para poder realizar esa reducción, se debe castear el término, y entonces el tipo es preservado:

$$\Uparrow_\ell |0\rangle \times (|0\rangle + |1\rangle) \to |0\rangle \times |0\rangle + |0\rangle \times |1\rangle$$

Teorema 5.5 (Preservación de tipos en términos cerrados, [Díaz-Caro y Dowek, 2017, Teorema 2])**.** Si $t \longrightarrow_{(p_i)} u_i$ and $\vdash t : A$, entonces $\vdash u_i : A$. □

Teorema 5.6 (Normalización fuerte, [Rinaldi, 2018, Teorema 5.16])**.** Si $\vdash t : A$ entonces t es fuertemente normalizante. □

Algoritmo de Deutsch

La compuerta Hadamard puede ser implementada de la siguiente manera:

$$\mathsf{H} = \lambda x{:}\mathbb{B}.{}^1\!/\!_{\sqrt{2}}.(|0\rangle + (\mathsf{isZ}(x)?(-|1\rangle):|1\rangle))$$

Notar que la variable es un tipo de base, y por lo tanto, si H se aplica a una superposición, por ejemplo $(\alpha.|0\rangle + \beta.|1\rangle)$, reduce de la siguiente manera:

$$\mathsf{H}(\alpha.|0\rangle + \beta.|1\rangle) \xrightarrow{(\mathsf{lin}^+_{\mathsf{r}})}_{(1)} (\mathsf{H}\alpha.|0\rangle + \mathsf{H}\beta.|1\rangle) \xrightarrow{(\mathsf{lin}^{\alpha}_{\mathsf{r}})^2}_{(1)} (\alpha.\mathsf{H}|0\rangle + \beta.\mathsf{H}|1\rangle)$$

y por lo tanto, finalmente se aplica a términos de base.
Definimos H_1 como la función que toma dos qubits y aplica H al primero:

$$\mathsf{H}_1 = \lambda x{:}\mathbb{B} \times \mathbb{B}\ ((\mathsf{H}\ (\mathit{head}\ x)) \times (\mathit{tail}\ x))$$

Similarmente, H_{both} aplica H a ambos qubits:

$$\mathsf{H}_{both} = \lambda x{:}\mathbb{B} \times \mathbb{B}\ ((\mathsf{H}\ (\mathit{head}\ x)) \times (\mathsf{H}\ (\mathit{tail}\ x)))$$

El oráculo U_f está definido por:

$$U_f|xy\rangle = |x, y \oplus f(x)\rangle$$

donde $\oplus$ es la suma modulo 2. Para implementarlo, necesitamos la compuerta *not*, la que puede ser implementada como sigue:

$$\mathsf{not} = \lambda x{:}\mathbb{B}\ (\mathsf{isZ}(x)?|0\rangle:|1\rangle)$$

Entonces, U_f es:

$$\mathsf{U}_f = \lambda x{:}\mathbb{B} \times \mathbb{B}\ ((\mathit{head}\ x) \times (\mathsf{isZ}((\mathit{tail}\ x))?(\mathsf{not}\ (f\ (\mathit{head}\ x))):(f\ (\mathit{head}\ x))))$$

donde f es una función dada de tipo $\mathbb{B} \Rightarrow \mathbb{B}$.
Finalmente, el algoritmo de Deutsch combina todas las definiciones previas:

$$\mathsf{Deutsch}_f = \pi_1\ (\Uparrow_r \mathsf{H}_1\ (\mathsf{U}_f \Uparrow_\ell \Uparrow_r \mathsf{H}_{both}\ (|0\rangle \times |1\rangle)))$$

Los casteos luego de las compuertas Hadamard se necesitan para desarrollar el término por completo para luego poder pasarlo a una abstracción que espera términos de base.
El término $\mathsf{Deutsch}_f$ se tipa como sigue:

$$\vdash \mathsf{Deutsch}_f : \mathbb{B} \times S(\mathbb{B})$$

Este término, en la función identidad, reduce así:

$$\mathsf{Deutsch}_{id} \longrightarrow^*_{(1)} \pi_1({}^1\!/\!_{\sqrt{2}}.|10\rangle - {}^1\!/\!_{\sqrt{2}}.|11\rangle) \xrightarrow{(\mathsf{proj})}_{(1)} |1\rangle \times ({}^1\!/\!_{\sqrt{2}}.|0\rangle - {}^1\!/\!_{\sqrt{2}}.|1\rangle)$$

Algoritmo de teleportación En el ejemplo precedente la aplicación de la medición tenía sólo un resultado posible: el primer qubit ya estaba en un estado de base antes de medirlo. Por lo tanto, introducimos un ejemplo un poco más complejo, la teleportación, donde la medición es usada como un operador que cambia el estado.
La compuerta *cnot* la implementamos como sigue:

$$\mathsf{cnot} = \lambda x : \mathbb{B} \times \mathbb{B}\ ((\mathit{head}\ x) \times (\mathsf{isZ}((\mathit{head}\ x))?(\mathsf{not}\ (\mathit{tail}\ x)) : (\mathit{tail}\ x)))$$

Definimos H_1^3 que aplica H al primer qubit en un sistema de tres qubits:

$$\mathsf{H}_1^3 = \lambda x{:}\mathbb{B} \times \mathbb{B} \times \mathbb{B}\ ((\mathsf{H}\ (\mathit{head}\ x)) \times (\mathit{tail}\ x))$$

Notar que la única diferencia con H_1 es el tipo de la variable. También, necesitamos aplicar *cnot* a los dos primeros qubits de un sistema de tres qubits, por lo que definimos cnot_{12}^3:

$$\lambda x{:}\mathbb{B} \times \mathbb{B} \times \mathbb{B}\ ((\mathsf{cnot}\ (\mathit{head}\ x \times (\mathit{head}\ \mathit{tail}\ x))) \times (\mathit{tail}\ \mathit{tail}\ x))$$

La compuerta Z se implementa de manera similar a not:

$$\mathsf{Z} = \lambda x{:}\mathbb{B}\ (\mathsf{isZ}(x)?(-|1\rangle) : |0\rangle)$$

A la parte de Alice la definimos así:

$$\mathsf{Alice} = \lambda x{:}S(\mathbb{B}) \times S(\mathbb{B} \times \mathbb{B})\ (\pi_2(\Uparrow_r \mathsf{H}_1^3\ (\mathsf{cnot}_{12}^3(\Uparrow_\ell \Uparrow_r x))))$$

Notar que antes de pasar a cnot_{12}^3 el parámetro de tipo $S(\mathbb{B}) \times S(\mathbb{B} \times \mathbb{B})$, necesitamos desarrollar el término por completo usando dos casteos, y de nuevo luego de la compuerta Hadamard.
El lado de Bob del algoritmo aplicará ciertas compuertas basado en los bits que reciba de Alice. Por lo tanto, para cualquier $\vdash \mathsf{U} : \mathbb{B} \Rightarrow S(\mathbb{B})$ o $\vdash \mathsf{U} : \mathbb{B} \Rightarrow \mathbb{B}$, definimos $\mathsf{U}^{(b)}$ como la función que aplica U o no dependiendo del bit b:

$$\mathsf{U}^{(b)} = (\lambda x{:}\mathbb{B}\ \lambda y{:}\mathbb{B}\ (\mathsf{isZ}(x)?\mathsf{U}y : y))\ b$$

Bob se implementa como sigue:

$$\mathsf{Bob} = \lambda x{:}\mathbb{B} \times \mathbb{B} \times \mathbb{B}\ (\mathsf{Z}^{(\mathit{head}\ x)}(\mathsf{not}^{(\mathit{head}\ \mathit{tail}\ x)}\ (\mathit{tail}\ \mathit{tail}\ x)))$$

El estado de Bell lo definimos directamente:

$$\beta_{00} = ({}^1/_{\sqrt{2}}.|00\rangle + {}^1/_{\sqrt{2}}.|11\rangle)$$

Finalmente, la teleportación se define por:

$$\mathsf{Telep} = \lambda q{:}S(\mathbb{B})\ (\mathsf{Bob}(\Uparrow_\ell \mathsf{Alice}\ (q \times \beta_{00})))$$

Este tipo tiene el tipo esperado $S(\mathbb{B}) \Rightarrow S(\mathbb{B})$, y aplicado a cualquier superposición $(\alpha.|0\rangle + \beta.|1\rangle)$ reduce, como es de esperarse, a $(\alpha.|0\rangle + \beta.|1\rangle)$.

Bibliografía

Pablo Arrighi, Alejandro Díaz-Caro, y Benoît Valiron. The vectorial lambda-calculus. *Information and Computation*, 254(1):105–139, 2017.

Pablo Arrighi y Gilles Dowek. Linear-algebraic lambda-calculus: higher-order, encodings, and confluence. En *Proceedings of RTA-2008*, (editado por Andrei Voronkov), tomo 5117 de *Lecture Notes in Computer Science*, págs. 17–31. Springer, 2008.

Pablo Arrighi y Gilles Dowek. Lineal: A linear-algebraic lambda-calculus. *Logical Methods in Computer Science*, 13(1:8), 2017. doi:10.23638/LMCS-13(1:8)2017.

Charles Bennett y Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. En *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, págs. 175–179. 1984.

Charles Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, y William Wootters. Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channels. *Physical Review Letters*, 70(13):1895–1899, 1993.

Charles Bennett y Stephen Wiesner. Communication via one- and two-particle operators on Einstein–Podolsky–Rosen states. *Physical Review Letters*, 69(20):2881–2884, 1992.

Charles H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.

Garret Birkhoff y John von Neumann. The logic of quantum mechanics. *Annals of Mathematics*, 37(4):823–843, 1936.

Julian Brown. *The Quest for the Quantum Computer*. Touchstone, 2001.

David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

David Deutsch y Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 439(1907):553–558, 1992.

Roberto Di Cosmo y Dale Miller. Linear logic. *The Stanford Encyclopedia of Philosophy*, Winter Edition, 2016. Edward N. Zalta (ed.). `https://plato.stanford.edu/archives/win2016/entries/logic-linear`.

Alejandro Díaz-Caro y Gilles Dowek. Typing quantum superpositions and measurement. En *Theory and Practice of Natural Computing (TPNC 2017)*, (editado por Carlos Martín-Vide, Roman Neruda, y Miguel A. Vega-Rodríguez), tomo 10687 de *Lecture Notes in Computer Science*, págs. 281–293. Springer, Cham, 2017.

Paul A. M. Dirac. A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35(03):416–418, 1939.

Gilles Dowek y Jean-Jacques Lévy. *Introduction to the theory of programming languages*. Undergraduate topics in computer science. Springer, 2011.

Albert Einstein, Boris Podolsky, y Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, 44(10):777–780, 1935.

Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieure*. Tesis Doctoral, Université Paris Diderot, París, Francia, 1972.

Jean-Yves Girard. Linear logic. *Theoretical Compututer Science*, 50:1–102, 1987.

Lov K. Grover. A fast quantum mechanical algorithm for database search. En *Proceedings of the 28th Annual ACM Symposium on Theory of computing*, STOC-96, págs. 212–219. ACM, 1996.

A. Kfoury, J. Tiuryn, y P. Urzyczyn. The undecidability of the semi-unification problem. En *Proceedings of STOC-1990*, págs. 468–476. 1990.

Michael Nielsen y Isaac Chuang. *Quantum Computation and Quantum Information. 10th Anniversary Edition*. Cambridge University Press., 2010.

Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977.

John Preskill. Quantum computing: pro and con. *Proceedings of the Royal Society of London A*, 454:469–486, 1998.

John C. Reynolds. Towards a theory of type structure. En *Programming Symposium: Proceedings of the Colloque sur la Programmation*, (editado por Bernard Robinet), tomo 19 de *Lecture Notes in Computer Science*, págs. 408–425. Springer, 1974.

Juan Pablo Rinaldi. *Demostrando normalización fuerte sobre una extensión cuántica del lambda cálculo*. Proyecto Fin de Carrera, Universidad Nacional de Rosario, Argentina, 2018.

Peter Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

Peter Selinger y Benoît Valiron. A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science*, 16(3):527–552, 2006.

Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

Morten H. Sørensen y Pawel Urzyczyn. *Lectures on the Curry-Howard Isomorphism*, tomo 149 de *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2006.

André van Tonder. A lambda calculus for quantum computation. *SIAM Journal on Computing*, 33:1109–1135, 2004.

Gilbert S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Transactions of the American Institute of Electrical Engineers*, XLV:295–301, 1926.

Joe B. Wells. Typability and type checking in the second-order lambda-calculus are equivalent and undecidable. En *Proceedings of LICS-1994*, págs. 176–185. IEEE Computer Society, 1994.

William K. Wootters y Wojciech .H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982.