UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Operador de medición en un cálculo lambda con control cuántico

Tesis de Licenciatura

Nicolás San Martín

Director de tesis: Alejandro Díaz-Caro
Codirector de tesis: Pablo E. Martínez López

Buenos Aires, 2023

Índice general

Resumen

En los últimos años se han desarrollado distintas extensiones al cálculo lambda buscando lenguajes de programación cuánticos siguiendo el modelo de "control cuántico". Este modelo, a diferencia del de "control clásico", describe las operaciones cuánticas de manera explícita, incorporando conceptos de la computación cuántica como el de las superposiciones al cálculo. Ejemplos de tales lenguajes son Lambda-$\mathcal{S}$ y Lambda-$\mathcal{S}_1$. El primero enfocado principalmente en incorporar la medición cuántica a los cálculos anteriores donde todas las operaciones son lineales. El segundo asegura que las superposiciones se mantienen en la esfera de módulo 1 haciendo que las operaciones sean isometrías, lo que es también un requisito para la computación cuántica. En esta tesis se define Lambda-$\mathcal{S}_1^\pi$, que es un cálculo que preserva la norma de las superposiciones, asegura que las operaciones son isometrías, y a la vez incorpora la medición cuántica. Se define el lenguaje, se prueban la propiedad de subject reduction, progreso, preservación de la normal y un resultado de expresividad.

Palabras clave: cálculo lambda, computación cuántica, medición cuántica, compuertas cuánticas

Capítulo 1

Introducción

En computación, para modelar lenguajes de programación secuencial se utiliza un formalismo conocido como lambda cálculo, que permite estudiar las propiedades de los programas. Este formalismo se basa en las nociones de función y de aplicación de una función a un argumento. Una función se construye *abstrayendo* sobre una variable en una expresión. La *aplicación*, en tanto, se obtiene sustituyendo en la expresión la variable abstraída por el argumento al cual se aplica.

En computación cuántica existen diversos enfoques para formalizar el estudio del comportamiento de los programas cuánticos. Muchos de ellos toman el lambda cálculo como base y buscan hacer extensiones que permitan capturar las características de la computación cuántica.

La computación cuántica tiene restricciones impuestas por el modelo físico. Las mismas son la imposibilidad de clonar un qubit, obteniendo una copia del mismo, y que toda superposición de estados que conforma un qubit se mantenga en la esfera unitaria. La primera se sigue de un resultado según el cual no existe en el modelo físico una transformación tal que realice una copia y la segunda es un postulado de dicho modelo.

Mientras que la computación clásica se basa en listas de *bits*, o dígitos binarios, y operaciones sobre los mismos, en computación cuántica el elemento básico es el bit cuántico, o *qubit*, con el que se forman sistemas de varios qubits y sobre los que se realizan operaciones. La cantidad de estados que puede admitir un bit es igual a dos y y las operaciones sobre un bit pueden ser la negación (o sea cambiar de estado), la identidad o setear alguno de los dos estados independientemente de aquel en el que esté. Esto es muy distinto en el caso de los qubits, ya que puede adoptar infinidad de estados. Por otra parte, el estado de un bit clásico puede conocerse mediante una simple lectura que no tenga como efecto ninguna alteración de su estado. En cambio, no existe una acción similar que permita conocer el estado de un qubit y que no tenga un efecto sobre el mismo.

Un programa cuántico describe la evolución de un sistema de qubits (o memoria cuántica) ya sea por medio de transformaciones unitarias o de mediciones. De este modo, pueden especificarse las transformaciones indicando qué compuertas cuánticas se van a realizar y así escribir un programa mediante un *circuito cuántico*. Sin embargo, también se han desarrollado diversos lenguajes de programación cuánticos que permiten programar con un mayor nivel de abstracción que mediante compuertas.

Estos lenguajes suelen clasificarse según se ejerza un control clásico o uno cuántico de los programas [5, 4]. El enfoque de *control clásico* incluye unos registros cuánticos sobre los que

pueden aplicarse operaciones unitarias, junto a registros clásicos que permiten guardar los resultados de las mediciones y estructuras de control clásico que permiten programar la evolución. En este modelo de "datos cuánticos, control clásico" la memoria cuántica se incorpora como una "caja negra". Como ejemplos de este tipo de lenguajes podemos mencionar quantum lambda calculus [15], Quipper [12], QWIRE [14] y QML [17].

El otro modelo de "datos cuánticos, control cuántico" apunta a desarrollar lenguajes puramente cuánticos que hagan explícita esa caja negra y que no escindan las estructuras de control de las operaciones cuánticas que se aplican sobre los datos. Uno de los primeros lenguajes que cuadran en este enfoque es Linear-algebraic lambda-calculus (o simplemente Lineal [1]), el cual es una extensión del calculo lambda no tipado y que permite *superponer* programas, aunque no permite llevar a cabo mediciones. Esta superposición es posible por la inclusión en el lenguaje de las combinaciones lineales de los términos de modo tal que si s y t son términos, entonces también lo es $u = \alpha \cdot s + \beta \cdot t$ donde $\alpha, \beta \in \mathbb{C}$.

Otros lenguajes basados en Lineal son Lambda-$\mathcal{S}$ [4, 5, 8] y Lambda-$\mathcal{S}_1$ [6, 10]. Como veremos, Lambda-$\mathcal{S}$ se enfoca en incorporar la medición al lenguaje mientras que Lambda-$\mathcal{S}_1$ carece de operador de medición pero asegura que las superposiciones se mantengan en la esfera unitaria para satisfacer los requerimientos teóricos de la computación cuántica.

Si bien un lenguaje de programación cuántico debe satisfacer la propiedad de no clonado, es decir, no debe permitir que un programa duplique un estado copiando un qubit arbitrario, no existe la misma restricción para vectores de la base computacional. La limitación surge de la imposibilidad de una transformación que dado un qubit en un estado desconocido, lo replique para dar lugar a dos en ese mismo estado. Sin embargo, sí existe una transformación que dado un qubit de la base computacional (como $\{|0\rangle, |1\rangle\}$) permita ese resultado, la cual puede implementarse utilizando la compuerta cuántica conocida como CNOT.

Así por ejemplo, si $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ es un qubit y $\lambda x.x \otimes x$ una función que mapea $\mathbb{C}^2 \to \mathbb{C}^4$, entonces no queremos que el lenguaje permita que la aplicación $(\lambda x.x \otimes x)|\psi\rangle$ evalúe a $|\psi\rangle \otimes |\psi\rangle$ y sin embargo sí permitimos que $(\lambda x.x \otimes x)|0\rangle$ evalúe a $|0\rangle \otimes |0\rangle$.

Basándose en el Linear lambda-calculus, un mecanismo para evitar esta duplicación es el de marcar ciertos "recursos lineales" los cuales no pueden ser duplicados, diferenciándolos de los "no-lineales" que sí (donde "lineal" se usa en este sentido distinto del de cumplir con las propiedades de *aditividad* y *homogeneidad* de las transformaciones lineales). Así, en el Lineal lambda-calculus un término no puede duplicarse a menos que su tipo esté marcado con ! (bang).

La estrategia utilizada tanto en Lambda-$\mathcal{S}$ como en Lambda-$\mathcal{S}_1$ es la de diferenciar entre los tipos básicos, que representan los vectores de la base computacional, de las superposiciones. Los primeros pueden duplicarse mientras que las segundas no. De este modo, una función podrá duplicar su argumento si éste es un término del tipo básico, pero no si es una superposición. Para esto se usa el sistema de tipado cuya regla Contr es la única que permite duplicar variables (siguiendo el enfoque de la lógica lineal [11]).

Otra forma de conseguir este resultado pero admitiendo la duplicación de los qubits $|0\rangle$ y $|1\rangle$ es la de *distribuir* las aplicaciones al ser aplicadas a una combinación lineal, pero reducirse si son aplicadas a un vector de la base, así:

$$(\lambda x.x \otimes x)(\alpha\,|0\rangle + \beta\,|1\rangle) \to^* \alpha(|0\rangle \otimes |0\rangle) + \beta(|1\rangle \otimes |1\rangle)$$

Este comportamiento es el adecuado para modelar las operaciones unitarias de la evolución cuántica, pero no para el caso de la medición. Y si queremos admitir funciones que se apliquen sobre una superposición en forma no lineal (es decir, que reduzcan reemplazando la superposición en lugar de distribuir primero) debemos garantizar que dicha función no va a duplicar dicha superposición. Lambda-$\mathcal{S}_1$ define las reglas de reescritura de forma tal que una aplicación únicamente reduce cuando el argumento no es una superposición. De esta forma, una abstracción aplicada a un término que sea una superposición deberá primero distribuirse linealmente para luego beta reducir. En cambio, Lambda-$\mathcal{S}$ incluye dos reglas de beta reducción según el tipo del parámetro sea o no una superposición, lo cual está codificado en el tipo de la abstracción: ésta puede esperar uno u otro tipo. Si lo que espera (el tipo del parámetro) es un vector de la base computacional, entonces se sigue la estrategia de *call-by-base*, es decir, reducirá si el argumento es $|0\rangle$ o $|1\rangle$, pero si no deberá aplicar primero las reglas de distribución lineal. En el otro caso, en cambio, la estrategia seguida es *call-by-name*, reemplazando la superposición (lo cual es seguro ya que el cuerpo de la abstracción, por las reglas de tipado como dijimos, no duplica dicha variable).

La propiedad de no clonado de qubits no es la única restricción que debe tenerse en cuenta para un lenguaje que busque ser aplicable a la computación cuántica. Es necesario de algún modo evitar operar en qubits cuya norma no sea unitaria tal como requieren los postulados. Este es el foco del lenguaje Lambda-$\mathcal{S}_1$ que adapta los tipos del lenguaje para dar con este resultado. Sin embargo, una limitación de este lenguaje es que no posee, como sí lo hace Lambda-$\mathcal{S}$, un operador de medición.

A. Díaz-Caro, M. Guillermo, A. Miquel, y B. Valiron. definen un modelo de realizabilidad que da lugar a una semántica que permite derivar el sistema de tipos, en lugar de partir de las reglas de tipado [7]. Es decir, se utiliza un predicado de realizabilidad [7, Def. IV.2] que se basa en interpretar cada tipo en un conjunto que garantiza las propiedades buscadas, y se demuestra que el tipado es válido si y solamente si se cumple ese predicado para el mismo término. Así, las reglas de tipado presentadas por Díaz-Caro y Malherbe son un subconjunto de los juicios de tipado derivables en el modelo se realizabilidad [10]. La ventaja de este enfoque es que el teorema de subject reduction está implicado por dicho modelo. En Lambda-$\mathcal{S}_1^\pi$, el lenguaje que presentaremos en esta tesis, por el contrario, tomamos las reglas de tipado (que son distintas) como punto de partida y probamos subject reduction.

1.1. Contenido de esta tesis

En esta tesis se presenta el cálculo Lambda-$\mathcal{S}_1^\pi$ que se basa en los cálculos Lambda-$\mathcal{S}$ y Lambda-$\mathcal{S}_1$, junto con demostraciones de algunas propiedades del mismo.

En el Capítulo 2, Preliminares, se exponen conceptos básicos de cálculo lambda, álgebra lineal y computación cuántica. Se presenta el cálculo lambda simplemente tipado con pares, incluyendo un teorema que tiene su análogo en Lambda-$\mathcal{S}_1^\pi$. Este teorema es *subject reduction*, que afirma que la reducción de un término conserva el tipo del mismo.

Luego viene una sección dedicada a la computación cuántica donde se definen los espacios vectoriales donde se modelan los sistemas cuánticos y su evolución y se introduce la notación *Dirac* que es la que se usa en la bibliografía. Además se presentan algunos algoritmos, a saber, la codificación superdensa y la teleportación cuántica, ya que éste último se usará asimismo como un ejemplo de aplicación para Lambda-$\mathcal{S}_1^\pi$.

En los capítulos 3 y 4 sobre Lambda-$\mathcal{S}$ y Lambda-$\mathcal{S}_1$ se explican dichos sistemas señalando aquellos aspectos relevantes para contextualizar Lambda-$\mathcal{S}_1^\pi$. En particular, se exponen sus respectivas gramáticas de términos, reglas de reescritura y de tipado. Luego, en el capítulo 5, se presenta Lambda-$\mathcal{S}_1^\pi$, indicando algunos aspectos que tiene en común o difiere con los precedentes, se prueba subject reduction y se muestra que expresa el cálculo simplemente tipado y la evolución de qubits.

Los tres lenguajes mencionados buscan modelar el cómputo cuántico con control cuántico. La contribución de Lambda-$\mathcal{S}_1^\pi$ es la de restringir los valores del cálculo a aquellos que tienen sentido desde el punto de vista de la teoría física, es decir los qubits, siguiendo a Lambda-$\mathcal{S}_1$ y a la vez incluir la operación de medición cuántica, como lo hace Lambda-$\mathcal{S}$. De este modo se diferencia del primero, que no cuenta con medición así como del segundo, que incluye valores que representan estados posibles en una computadora cuántica. Finalmente, en el capítulo 6 se presenta una conclusión del trabajo realizado.

Capítulo 2

Preliminares

2.1. Cálculo lambda

La siguiente sección introduce algunos conceptos del Cálculo lambda. Para una introducción más completa se pueden consultar las notas de P. Selinger [16].

2.1.1. Cálculo lambda simplemente tipado con pares

El *cálculo lambda* es un lenguaje formal cuyas expresiones se llaman *términos lambda* con reglas para manipular dichos términos.

Definición 2.1.1 (término)**.** Sea $\mathcal{V}$ un conjunto infinito denumerable de variables con $x, y \in \mathcal{V}$. Definimos el cojunto de términos del cálculo lambda con pares (denotado mediante Λ), mediante notación Backus-Naur así:

$$t = x \mid \lambda x.t \mid tt \mid (t,t) \mid \pi_1 t \mid \pi_2 t$$

Las expresiones como $\lambda x.t$ se llaman *abstracciones*, las que tienen la forma tt se llaman *aplicaciones*, (t,t) son *pares* y $\pi_i t$ *proyecciones*.

En un término lambda no todas las variables son interpretadas de la misma forma. El uso de una variable en el cuerpo de una función se puede referir al parámetro de dicha función, o a un elemento externo a la misma. Para formalizar esto, se utilizan las nociones de variables libres y ligadas. Una variable libre será considerada un elemento externo de la función, mientras que una variable ligada será el parámetro de dicha función. Se define formalmente la noción de variable libre, y se considera que toda variable que no esté libre, está ligada.

Definición 2.1.2 (variables libres)**.** Las ocurrencias de las variables en los términos pueden ser libres o ligadas. Informalmente, las variables en el cuerpo de una abstracción son ligadas por la variable que sigue a λ. Formalmente, definimos el conjunto de las variables libres de un término t, denotado $FV(t)$ en forma recursiva:

$$FV(x) = \{x\},$$

$$
\begin{aligned}
FV(tr) &= FV(t) \cup FV(r),\\
FV(\lambda x.t) &= FV(t) \setminus \{x\},\\
FV((t,r)) &= FV(t) \cup FV(r),\\
FV(\pi_i t) &= FV(t)
\end{aligned}
$$

De este modo en la expresión (x,x) ambas ocurrencias de las variables x son libres, mientras que en $\lambda x.(x,x)$ están ligadas. Esta última representa a la función que recibe un argumento y devuelve un par cuyos elementos son ambos ese mismo argumento. Visto de este modo, tal función no se diferencia esencialmente de $\lambda y.(y,y)$, que también devuelve lo mismo. La única diferencia entre ambas expresiones es el nombre de la única variable que aparece en ellas, o sea que estas expresiones son idénticas salvo en el nombre de la variable que hay en ellas. Para hablar de esta semejanza entre términos que no son idénticos se utiliza la noción de α-equivalencia, que se define formalmente a continuación, junto a la de renombre de variable.

Definición 2.1.3 (renombre de variable)**.** Dado un término t y las variables x, y, definimos el renombre de x por y en t, notado $t\{x := y\}$ así:

$$
\begin{aligned}
x\{x := y\} &\equiv y, && \\
z\{x := y\} &\equiv z && (\text{si } x \neq z),\\
(tr)\{x := y\} &\equiv (t\{x := y\})(r\{x := y\}), && \\
(\lambda x.t)\{x := y\} &\equiv \lambda y.t, && \\
(\lambda z.t)\{x := y\} &\equiv \lambda z.t\{x := y\} && (\text{si } y \neq z),\\
(t,r)\{x := y\} &\equiv (t\{x := y\}, r\{x := y\}), && \\
(\pi_i t)\{x := y\} &= \pi_i t\{x := y\} &&
\end{aligned}
$$

Definición 2.1.4 (α-equivalencia)**.** Llamamos *alfa-equivalencia* a la menor relación sobre términos lambda que satisface las reglas de la Tabla 2.1.

$$
\frac{}{t = t}\ (\text{refl}) \qquad \frac{t = t' \quad r = r'}{tr = t'r'}\ (\text{conj})
$$

$$
\frac{r = t}{t = r}\ (\text{symm}) \qquad \frac{t = t'}{\lambda x.t = \lambda x.t'}\ (\xi)
$$

$$
\frac{t = r \qquad r = s}{t = s}\ (\text{trans}) \qquad \frac{y \notin FV(t)}{\lambda x.t = \lambda y.t\{x := y\}}\ (\alpha)
$$

$$
\frac{t = t' \quad r = r'}{(t,r) = (t',r')} \qquad \frac{t = t'}{\pi_i t = \pi_i t'}
$$

Tabla 2.1: Reglas de alfa-equivalencia

Una operación importante en el lambda cálculo es la de *sustitución*, que reemplaza variables libres pero no por una variable con otro nombre, sino por un lambda término. Al hacerlo, no se reemplazan variables que estén ligadas. Es decir, en la expresión $(\lambda x.(x,y),x)$, si vamos a sustituir la x por t no queremos obtener $(\lambda x.(t,y),t)$ sino $(\lambda x.(x,y),t)$. Además, queremos evitar que alguna variable que esté libre en t resulte capturada en la expresión donde se sustituye. Por ejemplo, queremos evitar que si x está libre en t sea capturada al sustituir y por t en $(\lambda x.(x,y),x)$. Por el contrario, queremos que ambas variables sean diferenciadas, para lo cual lo que podemos hacer es renombrar la variable x que está ligada en $(\lambda x.(t,y),t)$ por otra que no se haya usado todavía. Se define formalmente la sustitución a continuación.

Definición 2.1.5 (Sustitución)**.** La *sustitución* de las ocurrencias libres de x por r en t, notada $t[x := r]$ se define de esta forma:

$$
\begin{aligned}
x[x := r] &\equiv r, && \\
y[x := r] &\equiv y, && \text{si } x \neq y, \\
(ts)[x := r] &\equiv (t[x := r])(s[x := r]) && \\
(\lambda x.t)[x := r] &\equiv \lambda x.t && \\
(\lambda y.t)[x := r] &\equiv \lambda y.t[x := r] && \text{si } x \neq y \text{ e } y \notin FV(r) \\
(\lambda y.t)[x := r] &\equiv \lambda y'.t\{y := y'\}[x := r] && \text{si } x \neq y, y \in FV(r) \text{ e } y' \text{ es nueva}, \\
(t,r)[x := y] &\equiv (t[x := y], r[x := y]), && \\
(\pi_i t)[x := y] &= \pi_i t[x := y] &&
\end{aligned}
$$

Con variable "nueva"nos referimos a una variable que no está presente ni en t ni en r.

Puede probarse que todo término es α*-equivalente* a otro en el cual las variables ligadas son diferentes entre ellas y también lo son de todas las variables libres. De este modo, cualquier demostración puede asumir sin pérdida de generalidad que las variables tienen distintos nombres en distintos términos, convención que se conoce como la convención de las variables de Barendregt. En esta tesis se usará dicha convención.

En el cálculo lambda simplemente tipado con pares se asocia un tipo a cierta clase de términos siguiendo ciertas reglas. Estos tipos pueden describirse de la siguiente forma:

$$A = \iota \mid A \to A \mid A \times A$$

donde ι denota un tipo básico o atómico. El tipo $A \to B$ es el de las funciones de A en B y el tipo $A \times B$ el de los pares (t,r) donde t es de tipo A y r de tipo B. En la Tabla 2.2 se presentan las reglas de tipado, las cuales permiten distinguir entre los términos que tienen tipo y los que no. Estas reglas se presentan mediante *juicios de tipado*, los cuales tienen 3 partes: el *contexto de tipado*, el término a tipar y el tipo dado. La notación $t : A$ significa t tiene el tipo A y los contextos de tipado consisten en un conjunto de asignaciones de términos a tipos, por ejemplo:

$$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \vdash t : A$$

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

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ \text{abs} \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash r : A}{\Gamma \vdash tr : B}\ \text{app}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash r : B}{\Gamma \vdash (t,r) : A \times B}\ \text{pair} \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_1 t : A}\ \pi_1 \qquad \frac{\Gamma \vdash t : A \times B}{\Gamma \vdash \pi_2 t : B}\ \pi_2$$

Tabla 2.2: Reglas de tipado

Este juicio de tipado afirma que asumiendo que las variables $x_1, \ldots, x_n$ tengan los tipos $A_1, \ldots, A_n$ respectivamente, el término t tiene tipo A,

Un concepto importante del lambda cálculo es el de *reducción*. Un término puede reducir a otro. Por ejemplo el término $(\lambda x.x)y$, o sea la aplicación de la función identidad a la variable y, reduce a y, el término $\pi_1(x,y)$, es decir la proyección del par (x,y) sobre la primera coordenada reduce a x. Hay términos, por otra parte, que no reducen, como por ejemplo x o (y,z) y también hay términos que al reducirlos dan lugar a términos que también se pueden reducir a su vez, como $(\lambda x.\pi_2(y,x))z$.

Definición 2.1.6 (reducción)**.** Definimos la *reducción* como la mínima relación $\to$ que satisface las reglas de la Tabla 2.3.

$$\begin{aligned}
(\lambda x.t)r &\longrightarrow t[x := r] && \beta \\
\pi_1(t,r) &\longrightarrow t && \pi_1 \\
\pi_2(t,r) &\longrightarrow r && \pi_2
\end{aligned}$$

$$\frac{t \longrightarrow t'}{\lambda x.t \longrightarrow \lambda x.t'} \qquad \frac{t \longrightarrow t'}{tr \longrightarrow t'r} \qquad \frac{r \longrightarrow r'}{tr \longrightarrow tr'}$$

$$\frac{t \longrightarrow t'}{(t,r) \longrightarrow (t',r)} \qquad \frac{r \longrightarrow r'}{(t,r) \longrightarrow (t,r')} \qquad \frac{t \longrightarrow t'}{\pi_1 t \longrightarrow \pi_1 t'} \qquad \frac{t \longrightarrow t'}{\pi_2 t \longrightarrow \pi_2 t'}$$

Tabla 2.3: Reglas de reducción

Se llama *redex* a cualquier expresión a la cual puede aplicarse alguna de las reglas de reducción β, π_1 o π_2 de la Tabla 2.3. Por ejemplo, el término $((\lambda x.x)(\lambda y.y))\lambda z.z$ no es un redex, pero su subtérmino $(\lambda x.x)(\lambda y.y)$ sí lo es.

Si $t \to r$ entonces decimos que t *reduce* a r y que r es su *reducto*. Si un término no se puede reducir, entonces decimos que se encuentra en *forma normal*.

Por otra parte, denotamos como $\rightarrow^*$ a la clausura transitiva y reflexiva de $\rightarrow$ y decimos que si $t \rightarrow^* t'$ y t' está en forma normal, t *evalúa* a t'.

2.1.2. Subject reduction

La redución en el cálculo lambda simplemente tipado tiene la propiedad de que todo término bien tipado reduce a un término bien tipado del mismo tipo. Esta propiedad recibe el nombre de *subject reduction*.

Antes que esta propiedad, probaremos el Lema de Sustitución, que afirma que un término conserva el tipo tras una sustitucíon.

Lema 2.1.1 (Lema de Sustitución)**.** Si $\Gamma, x : A \vdash t : B$ y $\Gamma \vdash r : A$, entonces $\Gamma \vdash t[x := r] : B$.

Demostración. Por inducción en t.

- Caso $t = x$. Aquí $A = B$ y por lo tanto $\Gamma \vdash t[x := r] : B$. Pues $t[x := r] = r$.
- Caso $t = y \neq x$. Aquí $y : B \in \Gamma$ y por var, $\Gamma \vdash y : B$. Y como $y = y[x := r] = t[x := r]$, entonces $\Gamma \vdash t[x := r] : B$.
- Caso $t = \lambda y . t'$. Por la regla de tipado abs, se cumple que $B = A' \rightarrow B'$ y que $\Gamma,\ x : A,\ y : A' \vdash t' : B'$.

  Además, por hipótesis inductiva, $\Gamma\ y : A' \vdash t'[x := r] : B'$ y por ende: $\Gamma \vdash \lambda y . t'[x := r] : A$.
- Caso $t = t'r'$. Aquí, por las reglas de tipado tiene que ser: $\Gamma, x : A \vdash t' : A' \rightarrow B$ y $\Gamma, x : A \vdash r' : A'$. Entonces, por hipótesis inductiva: $\Gamma \vdash t'[x := r] : A' \rightarrow B$ y $\Gamma \vdash r'[x := r] : A'$, de donde usando la regla app: $\Gamma \vdash (t'r')[x := r] : B$.
- Caso $t = (t', r')$. Por la regla pair debe ser $B = A' \times B'$, $\Gamma, x : A \vdash t' : A'$ y $\Gamma, x : A \vdash r' : B'$. Usando la hipótesis inductiva, obtenemos $\Gamma \vdash t'[x := r] : A'$ y $\Gamma \vdash r'[x := r] : B'$. Además $(t', r')[x : r] = (t'[x := r], r'[x := r]) = t[x := r]$. De modo que usando pair tenemos $\Gamma \vdash t[x := r] : B$.
- Caso $t = \pi_1 t'$. Por la regla de tipado π_1 tiene que ser $\Gamma, x : A \vdash t' : B \times B'$ mientras que por la hipótesis inductiva $\Gamma \vdash t'[x := r] : B \times B'$. Luego, usando π_1, $\Gamma \vdash (\pi_1 t')[x := r] : B$ y $\Gamma \vdash t[x := r] : B$.
- Caso $t = \pi_2 t'$. Análogo al caso anterior pero usando la regla π_2.

□

Teorema 2.1.2 (Subject reduction)**.** Si $\Gamma \vdash t : A$, y $t \rightarrow r$, entonces $\Gamma \vdash r : A$.

Demostración. Por inducción en la derivación del juicio $t \rightarrow r$.

- Caso $t = (\lambda x\,.\,t')r' \to t'[x := r']$. De las reglas de tipado se sigue que $\Gamma,\ x : A' \vdash t' : A$:

$$\dfrac{\dfrac{\Gamma, x : A' \vdash t' : A}{\Gamma \vdash \lambda x\,.\,t' : A' \to A}\ \textsf{abs} \qquad \Gamma \vdash r' : A'}{\Gamma \vdash (\lambda x\,.\,t')r' : A}\ \textsf{app}$$

  Luego aplicamos el lema 2.1.1 y como $\Gamma, x : A' \vdash t' : A$, entonces $\Gamma \vdash t'[x := r'] : A$

- Caso $\pi_1(r, r') \to r$. Por las reglas de tipado π_1 y pair:

$$\dfrac{\dfrac{\Gamma \vdash r : A \quad \Gamma \vdash r' : B}{\Gamma \vdash (r, r') : A \times B}\ \textsf{pair}}{\Gamma \vdash \pi_1(r, r') : A}\ \pi_1$$

  De manera que $\Gamma \vdash r : A$

- Caso $\pi_2(r', r) \to r$. Similar al caso anterior, usando π_2 en vez de π_1.

- Caso $\dfrac{t \to t'}{(t, r) \to (t', r)}$. Por hipótesis inductiva, si $\Gamma \vdash t : B$, entonces $\Gamma \vdash t' : B$. Luego, en el árbol de derivación de $\Gamma \vdash (t, r) : A$ reemplazamos t por t'.

- Caso $\dfrac{r \to r'}{(t, r) \to (t, r')}$. Similar al anterior.

- Caso $\dfrac{t \to t'}{tr \to t'r}$. Por hipótesis inductiva, si $\Gamma \vdash t : B$, entonces $\Gamma \vdash t' : B$. Luego, en el árbol de derivación de $\Gamma \vdash tr : A$ reemplazamos t por t'.

- Caso $\dfrac{r \to r'}{tr \to tr'}$. Similar al caso anterior, pero reemplazando r en vez de t.

- Caso $\dfrac{t \to t'}{\lambda x\,.\,t \to \lambda x\,.\,t'}$.
  Por las reglas de tipado tenemos:

$$\dfrac{\Gamma, x : A' \vdash t : B}{\Gamma \vdash \lambda x\,.\,t : A' \to B}\ \textsf{abs}$$

  Además, por hipótesis inductiva, si $\Gamma, x : A' \vdash t : B$ entonces $\Gamma, x : A' \vdash t' : B$. De donde, aplicando abs:

$$\dfrac{\Gamma, x : A' \vdash t' : B}{\Gamma \vdash \lambda x\,.\,t' : A' \to B}\ \textsf{abs}$$

□

2.2. Computación cuántica

Incluimos en esta introducción algunos conceptos básicos de álgebra lineal que permiten modelar los *qubits* y las operaciones que pueden hacerse con una computadora cuántica, para profundizar en los temas de esta sección puede consultarse el libro de M. Nielsen e I. Chuang [13]. El álgebra lineal describe los espacios vectoriales, que son conjuntos de objetos llamados *vectores* y cuentan además con la operación de *suma* de vectores, operación que cumple con las leyes de asociatividad y conmutatividad, posee un elemento neutro (la *identidad aditiva*) y también un inverso aditivo para cada elemento, y la operación de *multiplicación escalar* (entre un vector y un *escalar*, que es el modo de referirse a los elementos de un cuerpo como los número reales o como los complejos en el contexto de álgebra lineal), que cumple con la distributividad tanto del escalar con respecto a la suma de vectores como del vector respecto a la suma de escalares, también con la asociatividad y tiene un elemento neutro, la *identidad multiplicativa*. Ambas operaciones son cerradas con respecto al conjunto de vectores que conforman el espacio vectorial.

En esta tesis se van a considerar vectores que consisten en listas finitas de números complejos, ya que son los relevantes para la aplicación del caso, y los escalares serán los números complejos.

Representamos los vectores ya sea como una *fila*:

$$[x_1, \ldots, x_n]$$

o como una *columna*:

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

La multiplicación escalar (o *producto por un escalar*) la definimos del siguiente modo:

$$a \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} ax_1 \\ \vdots \\ ax_n \end{bmatrix}$$

Y la suma de vectores, así:

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

2.2.1. Espacio de Hilbert

Los espacios de Hilbert son espacios vectoriales que incluyen una operación llamada *producto interno* (notada $\langle \cdot, \cdot \rangle$), tienen una *norma* y son completos (en el sentido de que toda sucesión de Cauchy de puntos en el espacio tiene límite en él también, ver más adelante).

El producto interno es una función que va de los pares de vectores a los escalares. Por ejemplo en el caso particular del espacio vectorial conformado por los pares de números reales podemos definir el producto interno como la suma del producto de sus coordenadas, es decir

$$\left\langle \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right\rangle = x_1 y_1 + x_2 y_2.$$

En uno sobre números complejos sin embargo, la operación así definida *no es* producto interno ya que existen $\vec{x} \neq 0$ tales que $\langle \vec{x}, \vec{x} \rangle = 0$, lo cual contradice una de las propiedades que lo definen. En tal espacio se usa el producto escalar pero conjugando el segundo vector del producto.

De forma general se define del siguiente modo:

Definición 2.2.1 (Producto interno)**.** Si V es un espacio vectorial sobre un cuerpo $\mathbb{K}$, el *producto interno* definido sobre V es una función $\langle , \rangle : V \times V \to \mathbb{K}$ tal que para $\vec{u}, \vec{v}, \vec{w} \in V$ y $a, b \in \mathbb{K}$ si cumple:

1. $\langle \vec{u}, \vec{u} \rangle \geq 0$
2. $\langle \vec{u}, \vec{u} \rangle = 0 \iff \vec{u} = \vec{0}$
3. $\langle \vec{w}, a\vec{u} + b\vec{v} \rangle = a\langle \vec{w}, \vec{u} \rangle + b\langle \vec{w}, \vec{v} \rangle$
4. $\langle a\vec{u} + b\vec{v}, \vec{w} \rangle = a^* \langle \vec{u}, \vec{w} \rangle + b^* \langle \vec{v}, \vec{w} \rangle$
5. $\langle \vec{u}, \vec{v} \rangle = \langle \vec{v}, \vec{u} \rangle^*$

Si el producto interno entre dos vectores $\vec{v}$ y $\vec{w}$ es cero, es decir, si $\langle \vec{v}, \vec{w} \rangle = 0$, entonces decimos que estos son *ortogonales* y lo notamos $\vec{v} \perp \vec{w}$.

Se llama *espacio pre-hilbert* a un espacio vectorial con producto interno. En un espacio pre-hilbert podemos definir la norma $\| \cdot \| : V \to \mathbb{R}$ así:

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

Para definir espacios de Hilbert de dimensión infinita se requiere una propiedad adicional, a saber, que dada la norma $\| \cdot \|$, toda sucesión $\{\vec{v}_k\}$ tal que $\|\vec{v}_k - \vec{v}_m\| \to 0$ si $k, m \to \infty$ (llamada *sucesión de Cauchy*), converge con dicha norma.

La computación cuántica involucra espacios vectoriales sobre $\mathbb{C}^n$ de dimensión finita y en tales espacios vectoriales un espacio pre-hilbert es también un espacio de Hilbert.

Dado un conjunto de vectores $\{\vec{x}_1, \ldots, \vec{x}_n\}$ se llama *combinación lineal* de esos vectores a la suma de la multiplicación de cada uno de ellos por algún escalar, es decir a

$$a_1\vec{x}_1 + \cdots + a_n\vec{x}_n$$

.

Si tal conjunto está incluido en un espacio vectorial, entonces las propiedades de los espacios vectoriales nos garantizan que cualquier combinación lineal suya también esté incluido en el mismo. O sea que el conjunto de todas estas combinaciones lineales conforman un espacio vectorial. Al espacio que resulta de tomar todas esas combinaciones lineales se lo llama *espacio generado* mientras que a ese conjunto de vectores de lo llama *conjunto generador*.

Definición 2.2.2 (Conjunto generador)**.** Dado un conjunto (generador) $S = \{\vec{v}_1, \dots \vec{v}_n\}$ de vectores en un espacio vectorial V sobre un cuerpo $\mathbb{K}$, el conjunto generado por S (*span*) es $\{\sum_{i=1}^{n} a_i \vec{v}_i \mid a_i \in \mathbb{K} \wedge \vec{v}_i \in S\}$, es decir el conjunto de todas las combinaciones lineales de vectores en S. Para denotar el conjunto generado por $\{\vec{v}_1, \dots \vec{v}_n\}$ escribimos también $\mathsf{span}\{\vec{v}_1, \dots \vec{v}_n\}$.

Claramente, dado un vector $\vec{x} \in \mathsf{span}\{S\}$ con $S = \{v_1, \dots, v_n\}$, entonces existen escalares $\alpha_1, \dots, \alpha_n$ tales que $\vec{x} = \sum_{i=1}^{n} \alpha_i v_i$.

Si sacando de un conjunto generador un vector el espacio resultante es el mismo, esto significa que existe una combinación lineal de los vectores del conjunto resultante que equivale al vector sacado. Se dice entonces que los vectores de tal conjunto son *linealmente dependientes*. En general se llama *linealmente dependientes* a los vectores $\vec{x}_1, \dots, \vec{x}_n$ si existen n escalares $\alpha_i, \cdots, \alpha_n$, alguno de los cuales es distinto de cero, tales que $\alpha_1 \vec{x}_1 + \cdots + \alpha_n \vec{x}_n = \vec{0}$, donde $\vec{0}$ es el vector nulo.

Si los vectores del conjunto generador S son linealmente independientes y generan el espacio V, entonces S es una *base* de V. De este modo, una base es un conjunto generador al cual si le sacamos cualquier vector el conjunto resultante es más chico.

Otra propiedad importante que un conjunto de vectores puede tener es la ortonormalidad en la cual la norma de cada uno de ellos es igual a uno y todos son ortogonales entre sí:

Definición 2.2.3 (Conjunto ortonormal)**.** El conjunto $V = \{\vec{v}_1, \dots, \vec{v}_n\}$ es un conjunto ortonormal si se cumple:

- $\langle \vec{v}_i, \vec{v}_j \rangle = 0$ si $i \neq j$
- $\|\vec{v}_i\| = \langle \vec{v}_i, \vec{v}_i \rangle = 1$.

Si sólo se cumple la primera propiedad entonces el conjunto se llama *ortogonal*.

Definición 2.2.4 (Operador lineal)**.** Dados V, W espacios vectoriales sobre un mismo cuerpo un *operador lineal* es una función $A : V \to W$ tal que se cumple;

$$A\left(\sum_i \alpha_i \vec{v}_i\right) = \sum_i \alpha_i A \vec{v}_i$$

Decimos que un operador lineal está definido en V si es una función de V en V. En este caso un operador importante es la identidad, notada como I, tal que $I\vec{v} = \vec{v}$ para todo $\vec{v} \in V$. Usualmente se representan los operadores lineales mediante matrices.

Si dado un operador A existe un operador B tal que $AB = BA = I$, entonces se lo llama *inverso* de A y se lo nota A^{-1}. No todo operador tiene un inverso.

2.2.2. Producto tensorial

En ciertas aplicaciones, como es el caso de la computación cuántica, nos interesa formar espacios vectoriales tomando espacios más pequeños y poniéndolos uno junto al otro. Por ejemplo, tomamos dos vectores de dos espacios y aplicamos transformaciones al par. Si esos vectores tienen dimensiones n y m respectivamente, vamos a obtener un espacio de dimensión nm que combine los n vectores del primer espacio con los m del segundo. Para esto definimos una operación $\otimes$ que se llama *producto tensorial*.

Definición 2.2.5 (Producto tensorial)**.** Dados los espacios vectoriales V y W generados por las bases B y C, definimos el espacio generado por $\{\vec{b}_i \otimes \vec{c}_j \mid \vec{b}_i \in B, \vec{c}_j \in C\}$ como el *producto tensorial* entre V y W o $V \otimes W$, donde $\vec{b}_i \otimes \vec{c}_j$ es el par $(\vec{b}_i, \vec{c}_j)$ y la operación $\otimes$ se extiende a los vectores de V y W bilinealmente, es decir:

$$\left(\sum_i \beta_i \vec{b}_i\right) \otimes \left(\sum_j \gamma_j \vec{c}_j\right) = \sum_{i,j} \beta_i \gamma_j (\vec{b}_i \otimes \vec{c}_j)$$

Como $V \otimes W$ es un espacio vectorial, no es lo mismo que tomar simplemente todos los pares cuyo primer elemento está en V y el segundo en W.

Definición 2.2.6 (Producto cartesiano de subconjuntos de espacios vectoriales)**.** Sean los espacios vectoriales V y W y sean $S \subset V$ y $T \subset W$. Definimos *producto cartesiano* entre S y T:

$$S \times T = \{(\vec{s}, \vec{t}) \mid \vec{s} \in S,\ \vec{t} \in T\} \simeq \{\vec{s} \otimes \vec{t} \mid \vec{s} \in S,\ \vec{t} \in T\}$$

Notar entonces que $S \times T \neq S \otimes T$. Por ejemplo si $S = T = \mathsf{span}\{\vec{i}, \vec{j}\}$, donde $\{\vec{i}, \vec{j}\}$ es la base canónica de $\mathbb{C}^2$, y $\vec{v} = \vec{i} \otimes \vec{i} + \vec{j} \otimes \vec{j}$ entonces $\vec{v} \in S \otimes T$ pero $\vec{v} \notin S \times T$.

Definición 2.2.7 (Producto tensorial entre matrices)**.** Sean $P \in \mathbb{C}^{n \times m}$ y $Q \in \mathbb{C}^{n' \times m'}$ dos matrices, el producto tensorial entre ambas se define $P \otimes Q \in \mathbb{C}^{nn' \times mm'}$ como:

$$P \otimes Q = \begin{bmatrix} p_{1,1}Q & \cdots & p_{1,m}Q \\ \vdots & \ddots & \vdots \\ p_{n,1}Q & \cdots & p_{n,m}Q \end{bmatrix}$$

donde $p_{i,j}$ son las componentes de P. En particular, si $P \in \mathbb{C}^{n \times 1}$ y $Q \in \mathbb{C}^{m \times 1}$,

$$\begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} \otimes \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix} = \begin{bmatrix} p_1 \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix} \\ \vdots \\ p_n \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix} \end{bmatrix} = \begin{bmatrix} p_1 q_1 \\ p_1 q_2 \\ \vdots \\ p_1 q_m \\ p_2 q_1 \\ \vdots \\ p_n q_m \end{bmatrix}$$

Ejemplo 2.2.1 ($\mathbb{C}^2 \otimes \mathbb{C}^2$)**.** Un ejemplo relevante de espacio vectorial para la computación cuántica es el siguiente.

Sea $S = \mathsf{span}\left\{\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\} = \mathbb{C}^2$.

$$S \otimes S = \mathsf{span}\left\{\begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\} = \mathsf{span}\left\{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}\right\}.$$

Podemos ver que $S \times S \subsetneq S \otimes S$. Sea $\vec{v} = [\alpha^*, 0, 0, \beta^*]^T$ con $\alpha, \beta \neq 0$. Claramente, $\vec{v} \in S \otimes S$ ya que

$$\vec{v} = \alpha \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Sin embargo, supongamos que $\vec{v} = \vec{v}_1 \otimes \vec{v}_2$, $\vec{v}_1 = [a, b] \in S \times S$ y $\vec{v}_2 = [c, d] \in S \times S$. Entonces

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}$$

de modo que

$$\begin{aligned} ac &= \alpha \\ ad &= 0 \\ bc &= 0 \\ bd &= \beta \end{aligned}$$

pero de $ac = \alpha$ y $db = \beta$ se sigue que $a \neq 0$ y $d \neq 0$ pero $ad = 0$ y por lo tanto no tiene solución.

2.2.3. Notación bra-ket

La notación bra-ket (o notación de Dirac) fue introducida por Paul Dirac y es utilizada usualmente para describir los estados cuánticos.

Definición 2.2.8 (Ket)**.** Llamamos *ket* (notado $|\cdot\rangle$) a un vector columna, es decir un vector con la forma

$$|\psi\rangle = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$$

donde $\alpha_i \in \mathbb{C}$.

Definición 2.2.9 (Bra)**.** Llamamos *bra* (y lo notamos $\langle\cdot|$) a un vector fila.Es decir un vector con la forma

$$\langle\phi| = [\beta^*, \ldots, \beta^*]$$

donde $\beta_i \in \mathbb{C}$.Dado un *ket* $|\phi\rangle$, su bra $\langle\phi|$ se define como el traspuesto conjugado de $|\phi\rangle$.

Usamos esta notación en la que un vector *bra* es el traspuesto conjugado del correspondiente *ket* ya que de este modo podemos escribir el producto interno $\langle |\phi\rangle, |\psi\rangle\rangle$ como

$$\langle\phi|\psi\rangle = [\beta^*, \ldots, \beta^*] \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}$$

También definimos

$$|0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$|+\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad |-\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

También, dados los vectores $|\psi\rangle$ y $|\phi\rangle$ vamos a escribir $|\psi\phi\rangle$ (o $|\psi\rangle|\phi\rangle$) en vez de $|\psi\rangle \otimes |\psi\rangle$. Así, por ejemplo

$$|0\rangle \otimes |1\rangle = |0\rangle|1\rangle = |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Definición 2.2.10 (delta de Kronecker)**.** Vamos a utilizar la *delta de Kronecker* $\delta_{i,j}$ definida de este modo:

$$\delta_{i,j} = \begin{cases} 1 & \text{si } i \neq j \\ 0 & \text{si no} \end{cases} \tag{2.1}$$

Esta notación resulta útil para representar los productos de vectores ortonormales.

Definición 2.2.11 (base ortonormal)**.** Sea V un espacio de Hilbert. Una base $B = \{|u_i\rangle\}_i$ de V se dice *ortonormal* si el conjunto de vectores que la conforman es ortonormal, es decir si
$\langle u_i|u_j\rangle = \delta_{i,j}$, para $|u_i\rangle, |u_j\rangle \in B$

En lo que sigue adoptamos la convención de utilizar $|u_i\rangle$ para denotar el iésimo vector de la base ortonormal del espacio del que se trate.

Dado un ket $|\psi\rangle = \sum_{k=1}^{n} \alpha_k |u_k\rangle$ (vimos que siempre existe una tal combinación lineal si $\{|u_i\rangle\}$ es una base que genera un espacio al que pertenece $|\psi\rangle$), tenemos que $\langle u_i|\psi\rangle = \langle u_i| \sum_{k=1}^{n} \alpha_k |u_k\rangle = \sum_{k=1}^{n} \alpha_k \langle u_i|u_k\rangle = \alpha_i$. Es decir que multiplicando el vector por uno de los de una base ortonormal obtenemos como resultado la coordenada correspondiente al mismo en la representación de $|\psi\rangle$ respecto de dicha base.

Si $P_i = |u_i\rangle\langle u_i|$ entonces $P_i|\psi\rangle = |u_i\rangle\langle u_i|\psi\rangle = \alpha_i|u_i\rangle$. Es decir P_i proyecta el vector $|\psi\rangle$ sobre $|u_i\rangle$. Además $P^2 = |u\rangle\langle u|u\rangle\langle u| = |u\rangle 1 \langle u| = P$. Un operador P tal que $P^2 = P$ se llama proyector.

Teorema 2.2.1. Sea V un espacio vectorial y $B = \{u_i, \ldots, u_n\}$ una base ortonormal, entonces

$$\sum_{i=1}^{n} |u_i\rangle\langle u_i| = I$$

Demostración.

$$\left(\sum_{i=1}^{n} |u_i\rangle\langle u_i|\right)|\psi\rangle = \left(\sum_{i=1}^{n} |u_i\rangle\langle u_i|\right) \sum_{k=1}^{n} \alpha_k |u_k\rangle = \sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_k |u_i\rangle\langle u_i|u_k\rangle = \sum_{k=1}^{n} \alpha_k |u_k\rangle = |\psi\rangle$$

□

Sea $A \in \mathbb{C}^{n\times n}$ con $\alpha_{i,j}$ sus componentes y sea $B = \{u_1, \ldots, u_n\}$ una base ortonormal de $\mathbb{C}^n$. Entonces

$$A = \left(\sum_{i=1}^{n} |u_i\rangle \langle u_i|\right) A \left(\sum_{k=1}^{n} |u_k\rangle \langle u_k|\right) = \sum_{i=1}^{n}\sum_{k=1}^{n} |u_i\rangle \langle u_i|A|u_k\rangle \langle u_k| = \sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_{i,k} |u_i\rangle \langle u_k|.$$

dado que $\langle u_i|A|u_k\rangle = \alpha_{i,k}$.

Como vimos, un operador lineal es una función lineal que mapea de un espacio a otro, y puede representarse como una matriz. Una matriz se puede *trasponer*, lo que significa intercambiar sus coordenadas de manera tal que si trasponemos la matriz A, su traspuesta (notada A^T) es tal que $A_{ij} = (A^\mathsf{T})_{ji}$. Si la traspuesta de una matriz es también su inversa, es decir si $A^\mathsf{T} = A^{-1}$ entonces la matriz es *ortogonal*. Multiplicando a ambos lados por A obtenemos $AA^\mathsf{T} = I$ y $A^\mathsf{T}A = I$, de modo que $A = (A^\mathsf{T})^{-1}$ y por ende A^{-1} es también ortogonal pues $(A^\mathsf{T})^\mathsf{T} = A$.

Una característica de este tipo de matrices, en un espacio vectorial sobre números reales, es que su aplicación no modifica la norma de los vectores, o sea que si $Q \in \mathbb{R}^n$ es una matriz ortogonal y $\vec{v} \in \mathbb{R}^n$, entonces $\|Q\vec{v}\| = \|v\|$, de modo que en este caso las transformaciones ortogonales *preservan la longitud*. Pero también ocurre que $\langle Q\vec{x}, Q\vec{y}\rangle = \langle \vec{x}, \vec{y}\rangle$, de manera que también preservan los ángulos entre los vectores.

Si además de trasponer una matriz conjugamos sus coordenadas obtenemos la matriz *adjunta*.

Definición 2.2.12 (Adjunto)**.** El adjunto de un operador A, notado $A^\dagger$, es el traspuesto conjugado de A. Es decir las componentes de $A^\dagger$ son $\alpha^*_{ij} = \langle u_j|A|u_i\rangle^* = \langle u_i|A^\dagger|u_j\rangle$

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

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

Un operador A se dice *hermítico* si $A = A^\dagger$.

Definición 2.2.13 (Operador unitario)**.** Sea A un operador. Si $A^\dagger = A^{-1}$, entonces es un operador *unitario*.

El operador unitario es el análogo en el caso complejo, al operador ortogonal en el real.

Propiedades. Sea U un operador unitario, entonces:

- $\langle U\phi|U\psi\rangle = \langle \phi|U^\dagger U|\psi\rangle = \langle \phi|\psi\rangle$
- U^{-1} es un operador unitario.

- Si $\{|\psi_1\rangle, \ldots |\psi_n\rangle\}$ es una base ortonormal, entonces $\{U|\psi_1\rangle, \ldots U|\psi_n\rangle\}$ también.

Se llama *espacio métrico* a un conjunto M, junto a una noción de *distancia* $d : M \times M \to \mathbb{R}$ entre sus elementos que cumple:

- $d(x,x) = 0$,
- si $x \neq y$, $d(x,y) > 0$,
- $d(x,y) = d(y,x)$
- $d(x,z) \leq d(x,y) + d(y,z)$

Por ejemplo, dado una espacio vectorial en el que se define una norma, podemos decir que se trata de un espacio métrico en el que la distancia es la norma de la diferencia, es decir si $v, w \in V$, $d(v,w) = \|v - w\|$.

Definición 2.2.14 (Isometría)**.** Se llama *isometría* a un operador que mapea elementos de un espacio métrico en otro que preserva la distancia, o sea dados los espacios métricos V y W, $f : V \to W$ es una isometría si $d_V(v,u) = d_W(f(v), f(u))$ para todos $v, u \in V$.

Si U es un operador unitario en un espacio vectorial V con norma $\|\cdot\|$, entonces para cualesquiera $v, w \in V$, $\|v - w\|^2 = \langle v - w | v - w \rangle = \langle U(v - w) | U(v - w) \rangle = \langle Uv - Uw | Uv - Uw \rangle = \|Uv - Uw\|^2$, de manera que U es una isometría.

Definición 2.2.15 (Proyector)**.** Un operador $P = |\phi\rangle \langle\phi|$ es denominado *proyector* ya que proyecta cualquier vector $|\psi\rangle$ sobre el vector $|\phi\rangle$.

$$P|\psi\rangle = |\phi\rangle \langle\phi| |\psi\rangle = a|\phi\rangle$$

Ejemplo 2.2.2. Sea $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ y $\{|0\rangle, |1\rangle\}$ la base canónica de $\mathbb{C}^2$. Luego

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

y α es la proyección de $|\psi\rangle$ sobre $|0\rangle$.

2.2.4. Qubits

La computación cuántica opera sobre sistemas de qubits que transforma de acuerdo a un programa. A diferencia de los bits clásicos, los estados de los qubits no son finitos ni tampoco las operaciones que pueden aplicárseles, y sus estados no pueden ser medidos sin ser modificados de acuerdo a un modelo probabilístico.

Un *qubit* (o bit cuántico) es representado con un vector en $\mathbb{C}^2$ en la esfera unitaria (es decir con *norma* $|\cdot|$ igual a 1). Por ende, cualquier qubit puede representarse mediante una combinación lineal de los vectores de la base canónica de $\mathbb{C}^2$, $|0\rangle$ y $|1\rangle$. Esto recibe el nombre de *superposición* puesto que puede pensarse que se superponen los estados $|0\rangle$ y $|1\rangle$ (en contraste con los *bits* de la computación clásica, donde el estado puede ser 0 o 1 en forma exclusiva). Es decir que el estado que puede adquirir un qubit es cualquier combinación lineal de los vectores de una base de $\mathbb{C}^2$ siempre que su norma sea 1.

Definición 2.2.16 (Qubit)**.** Un qubit es un vector normalizado $|\psi\rangle$ en el espacio de Hilbert $\mathbb{C}^2$. De este modo, los qubits pueden representarse como $\alpha|0\rangle + \beta|1\rangle$ donde $\{|0\rangle, |1\rangle\}$ es la base canónica de $\mathbb{C}^2$ y $|\alpha|^2 + |\beta|^2 = 1$.

Definición 2.2.17 (Sistema de n-qubits)**.** Un sistema de n-qubits es un vector normalizado en el espacio $\mathbb{C}^{2^n}$.

El espacio vectorial $\mathbb{C}^{2^n}$ es el producto tensorial $\bigotimes_{i=1}^{n} \mathbb{C}^2$. Vamos a escribir equivalentemente $|b_1 b_2 \dots b_n\rangle$ o $|b_1\rangle \otimes |b_2\rangle \otimes \cdots \otimes |b_n\rangle$ o incluso $|k\rangle_n$ donde $b_i \in \{0,1\}$ y $k = \sum_{i=1}^{n} b_i 2^{n-i}$ (es decir k es la notación decimal del número binario de n dígitos $b_1 \dots b_n$) de modo que $\{|i\rangle_n\}_{0 \le i \le 2^n}$ es la base canónica de $\mathbb{C}^{2^n}$.

Así como un qubit puede describirse con los coeficientes de la base canónica de $\mathbb{C}^2$, un sistema de n qubits puede describirse con los coeficientes en la base computacional correspondiente, por ejemplo si $|\psi\rangle \in \mathbb{C}^{2^n}$:

$$|\psi\rangle = \alpha_1 |0\rangle_n + \cdots + \alpha_n |2^n - 1\rangle_n$$

En analogía a las compuertas lógicas de la computación clásica, a las que operan sobre sistemas de qubits se les llama *compuertas cuánticas*. Estas compuertas son *operadores unitarios*. El hecho de que sea unitario es requerido ya que el resultado de una operación debe ser un estado normalizado, lo cual es mantenido en una operación unitaria ya que si $|\psi\rangle = \sum_{i=1}^{n} \alpha_i |i\rangle_n$, y U es unitaria, entonces $U|\psi\rangle = \sum_{i=1}^{n} \alpha_i U |i\rangle_n$ donde, como vimos, $\{U|1\rangle_n, \dots U|n\rangle_n\}$ es una base ortonormal.

Por último, para conocer el estado de un qubit es necesario medirlo y la medición es una operación que cambia el estado. Es decir, no podemos sencillamente identificar el estado y conocer los coeficientes que lo describen. Lo que podemos hacer es aplicarle una operación (esta vez no unitaria) que nos permita conocer el estado resultante (no el previo), aunque el estado resultante depende, en parte, del estado previo. Para ello necesitamos un conjunto de operadores $\{M_m\}_n$ que satisface $\sum_m M_m M_m^\dagger = I$. Dado este conjunto, el resultado de la medición es uno de los estados $\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}$, cada uno con probabilidad $p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle$. Nótese que el resultado no es otra cosa que el vector $M_m|\psi\rangle$ renormalizado.

Definición 2.2.18 (Operador de medición)**.** Un conjunto $\{M_1, \dots, M_k\}$ de proyectores es un *operador de medición* si se cumple:

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

Definición 2.2.19 (Evolución)**.** Dado un sistema representado por $|\psi\rangle$ se dice que el mismo *evoluciona* al sistema $|\phi\rangle$ si se realiza alguna de estas operaciones:

- Se premultiplica por una compuerta cuántica U:

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

- Se aplica un operador de medición $M = \{M_1, \dots, M_k\}$ de esta forma:

$$|\phi\rangle = \frac{M_i|\psi\rangle}{\sqrt{\langle\psi|M_i^\dagger M_i|\psi\rangle}}$$

para algún $i \in [1,\ldots,k]$, donde el valor de i tiene la probabilidad

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

Existe un caso especial relevante de la medición que recibe el nombre de *medición proyectiva respecto a la base computacional* donde $M_i = |u_i\rangle\langle u_i|$ y $\{|u_i\rangle\}_i$ son los vectores de la base computacional.

En este caso particular de la medición cuántica, el estado tras aplicar la medición coincide exactamente con uno de los vectores de la base $\{|u_i\rangle\}_i$, por ejemplo si se mide un qubit y la base computacional es $\{|0\rangle, |1\rangle\}$, será $|0\rangle$ o $|1\rangle$ aleatoriamente: la probabilidad de que sea el vector $|k\rangle$ es igual a $|\alpha_k|^2$, o sea, es el cuadrado del módulo del coeficiente que acompaña dicho vector de la base, al escribirlo como una combinación lineal de esa base. Y esta probabilidad está bien definida, porque como requerimos que todo qubit sea un vector normalizado, esa suma es siempre igual a 1.

Así, al medir un qubit $|\psi\rangle$, el estado resultante del mismo es aleatorio y depende de una función de probabilidad. Concretamente, como vamos a usar la medición respecto a la base computacional, base canónica), dado el operador de medición compuesto por n proyectores $M_i = |u_i\rangle\langle u_i|$, podemos escribir $|\psi\rangle$ como $\sum_{i=1}^{n} a_i|u_i\rangle$ y el resultado de aplicarlo será la proyección normalizada de $|\psi\rangle$ sobre $|u_i\rangle$ (es decir $|u_i\rangle\langle u_i|$) y la elección del proyector usado es una experiencia aleatoria donde la probabilidad de cada uno es $|a_i|^2$.

Como dijimos, a diferencia de la computación clásica que tiene una pequeña cantidad de operaciones posibles sobre un bit, cualquier operador unitario es una compuerta cuántica entre las cuales encontramos algunos ejemplos como la identidad I, la negación X, el cambio de fase Z y la compuerta Hadamard H, que se definen así:

- Identidad: $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- Negación: $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
- Cambio de fase: $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
- Hadamard: $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Así como existen compuertas clásicas para múltiples bits (como AND, OR, XOR, NAND) lo mismo ocurre para múltiples qubits. Estas compuertas pueden querer usarse para evolucionar todo el sistema de qubits, pero también puede fácilmente aplicarse una compuerta que modifica un qubit en particular del sistema de n. Para esto, dada la compuerta simple $U \in \mathbb{C}^2$, si se quiere modificar con ella únicamente el k-ésimo qubit del sistema, se premultiplica tensorialmente U $k-1$ veces por I y se posmultiplica tensorialmente $n-k-1$ veces por I, o sea:

$$\bigotimes_{1}^{k-1} I \otimes U \otimes \bigotimes_{1}^{n-k-1} I$$

Un importante ejemplo de compuerta para un sistema de 2 qubits es la llamada CNOT (*controlled* NOT). Esta compuerta recibe dos qubits, uno es el qubit de *control*, el otro es el *target*. Si el control es $|0\rangle$, entonces el target resulta inalterado. Si en cambio, el control es $|1\rangle$, es negado. Es decir:

$$\begin{aligned}\text{CNOT}\,|00\rangle &= |00\rangle\\ \text{CNOT}\,|01\rangle &= |01\rangle\\ \text{CNOT}\,|10\rangle &= |11\rangle\\ \text{CNOT}\,|11\rangle &= |10\rangle\end{aligned}$$

matricialmente (por bloques): $\text{CNOT} = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix}$

Una propiedad importante de la computación cuántica (y en virtud de la cual nuevamente difiere de la computación clásica) es la *imposibilidad* de copiar un un qubit. Es decir, no existe un programa cuántico que dado un estado cuántico arbitrario, pueda clonarlo resultando con dos qubits en ese mismo estado.

Teorema 2.2.2 (No-clonado)**.** No existe una compuerta cuántica U tal que para algún $|\phi\rangle \in \mathbb{C}^n$ se cumpla que para todo $|\psi\rangle \in \mathbb{C}^n$ tengamos que $U\,|\psi\phi\rangle = |\psi\psi\rangle$.

Demostración. Supongamos que existen la operación U y $|\phi\rangle$ tales que para estados arbitrarios $|\psi\rangle, |\psi'\rangle \in \mathbb{C}^n$ se cumplen

$$U\,|\psi\phi\rangle = |\psi\psi\rangle \text{ y } U\left|\psi'\phi\right\rangle = \left|\psi'\psi'\right\rangle$$

Por ende, $\langle U\psi\phi|U\psi'\phi\rangle = \langle\psi\psi|\psi'\psi'\rangle$. Pero entonces como:

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

Y además

$$\langle\psi\psi|\psi'\psi'\rangle = \langle\psi|\psi'\rangle\langle\psi|\psi'\rangle = \langle\psi|\psi'\rangle^2$$

De manera que $\langle\psi|\psi'\rangle = \langle\psi|\psi'\rangle^2$ y por lo tanto $\langle\psi|\psi'\rangle = 0$ o $\langle\psi|\psi'\rangle = 1$. Y esto es un absurdo porque si vale 0 significa que los vectores son ortogonales y si vale 1 que son el mismo, pero fueron sin embargo tomados arbitrariamente. □

2.2.5. Estados de Bell

Consideremos la operación siguiente: $\text{CNOT}(H \otimes I)$ Esta operación, aplicada a la base canónica de $\mathbb{C}^4$ nos da los estados:

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$
$$|\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$
$$|\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

donde los subíndices indican a qué vector se aplica la operación. Estos qubits (que forman una base ortonormal de $\mathbb{C}^4$) reciben el nombre de *estados de Bell*. Se trata de 4 *estados entrelazados* es decir estados que no pueden representarse como el producto tensorial de dos qubits. Dicho de otra manera, son estados que pertenecen a $(\mathbb{C}^2 \otimes \mathbb{C}^2) \setminus (\mathbb{C}^2 \times \mathbb{C}^2)$. Cuando un par de qubits está entrelazado las partículas que lo componen pueden, físicamente, separarse y medirse por separado. Sin embargo, el estado completo del sistema no se puede describir matemáticamente de manera independiente.

2.2.6. Codificación superdensa

Una aplicación de los estados de Bell recibe el nombre de *codificación superdensa* la cual permite a un emisor transmitir a un receptor dos bit clásicos transmitiendo un único qubit.

Llamando Alice y Bob (respectivamente) al emisor y el receptor, el procedimiento es así. Ambos preparan un estado entrelazado β_{00} quedándose Alice con el primer qubit del sistema y Bob con el segundo. Este estado es previo a la comunicación en sí misma, puede ser provisto por un tercero a ambos, puede considerarse como un canal de comunicación.

Alice quiere comunicar dos bits clásicos, es decir alguna de estas cuatro cadenas: $00, 01, 10, 11$. La idea, entonces, es que Bob reciba de Alice un qubit tal que al medir el par resultante en esa base (o puede también transformarlo a la base canónica y medirlo en ella), pueda mapear cada uno de estos estados a una de las cadenas que Alice quiere transmitir. Y ella lo consigue de esta forma: si quiere transmitir 00, entonces "no hace nada" a su qubit (es decir le aplica I), si quiere transmitir 01 le aplica Z, si quiere enviar 10 aplica el X y, por último, para enviar 11 le aplica XZ.

Esta transformación es equivalente a $Z^{b_1}X^{b_2}$ (donde $A^0 = I$ y $A^1 = A$ y $b_1 b_2$ es la cadena a transmitir). En efecto:

$$(II \otimes I)\,|\beta_{00}\rangle = |\beta_{00}\rangle$$
$$(IX \otimes I)\,|\beta_{00}\rangle = |\beta_{01}\rangle$$
$$(ZI \otimes I)\,|\beta_{00}\rangle = |\beta_{10}\rangle$$
$$(ZX \otimes I)\,|\beta_{00}\rangle = |\beta_{11}\rangle$$

2.2.7. Teleportación cuántica

Ahora veamos otra aplicación en la cual también se transmite información, sólo que en lugar de transmitir información clásica se transmite información cuántica, enviando únicamente bits clásicos.

Este problema tiene la dificultad de que conocer el estado del qubit supone conocer información infinita, y eso no puede hacerlo Alice.

Nuevamente, aquí Alice y Bob establecen un canal conservando cada uno uno de los bits de $|\beta_{00}\rangle$. Sea $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ el estado a teleportar. Y sea

$$|\psi_0\rangle = |\psi\rangle|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle))$$

donde los qubit $|\psi\rangle$ y el primero de $|\beta_{00}\rangle$ están en poder de Alice, mientras que Bob posee el segundo de $|\beta_{00}\rangle$.

En primer lugar, Alice aplica CNOT a su par obteniendo $|\psi_1\rangle$:

$$|\psi_1\rangle = (\text{CNOT} \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle))$$

Luego aplica Hadamard a su primer qubit

$$\begin{aligned}|\psi_2\rangle &= (H \otimes I \otimes I)|\psi_1\rangle = \frac{1}{2}(\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle)) \\ &= \frac{1}{2}(|00\rangle(\alpha|0\rangle + \beta 1) + |01\rangle(\alpha|1\rangle + \beta 0) + |10\rangle(\alpha|0\rangle + \beta 1) + |11\rangle(\alpha|1\rangle + \beta 0))\end{aligned}$$

Si ahora Alice realiza una medición (sobre los dos qubits que están en su poder), entonces naturalmente obtendrá alguno de los cuatro elementos de a base de $\mathbb{C}^4$. Si $|\psi_3\rangle_{xy}$ es el resultado tras la medición, entonces los cuatro posibles resultados son:

$$\begin{aligned}|\psi_3\rangle_{00} &= |00\rangle(\alpha|0\rangle + \beta|1\rangle) \\ |\psi_3\rangle_{01} &= |01\rangle(\alpha|1\rangle + \beta|0\rangle) \\ |\psi_3\rangle_{10} &= |10\rangle(\alpha|0\rangle - \beta|1\rangle) \\ |\psi_3\rangle_{11} &= |11\rangle(\alpha|1\rangle - \beta|0\rangle)\end{aligned}$$

Y como $|\psi\rangle = \alpha|0\rangle + \alpha|1\rangle$, conociendo el resultado de la medición es posible recuperar el estado de $|\psi\rangle$. Por ejemplo, si Bob recibe de Alice que el resultado fue 00, sabe que el qubit que está en su poder es exactamente $|\psi\rangle$. Si recibe 01, deberá aplicarle X, si recibe 10, Z y si en cambio recibe que la medición resultó en 11, tiene que aplicar XZ.

Es decir que deberá aplicar $Z^{b_1}X^{b_2}$, donde b_1b_2 son los bits recibidos.

Capítulo 3

Lambda-$\mathcal{S}$

Lambda-$\mathcal{S}$ [5, 6, 9], es una extensión del cálculo lambda con pares tipado que incluye combinaciones lineales de términos (siguiendo a Lineal [1, 2]) así como un operador de medición y otras construcciones. Dados dos términos t y u, Lambda-$\mathcal{S}$ incluye a $\alpha t + \beta u$, que representa una combinación lineal de los mismos. Del tipo de este tercer término se dice que es una superposición, lo que se representa mediante una S, que se aplica a algún tipo base A de forma que SA es el span de A.

Dentro de este cálculo, los tipos superpuestos no admiten duplicación de sus términos, ya que la superposición de qubits no admite la duplicación y este sistema es justamente un cálculo cuántico. Pero los tipos "básicos" que no están marcados por la S sí se pueden clonar, ya que esos tipos respresentan la base computacional cuyos qubits sí pueden duplicarse.

La duplicación en un cálculo lambda se efectúa mediante funciones como $\lambda x.x \otimes x$ y existen dos formas de evitarla. Por un lado está la utilización de *tipos lineales*, lo que implica un sistema en el cual tal expresión no está bien tipada. Los tipos lineales están basados en la lógica lineal de Girard [11]. En ella, se limita el "uso de recursos" restringiendo el uso de la regla de weakening y *contraction* para fórmulas con la modalidad ! (*bang*). En Lambda-$\mathcal{S}$ los tipos marcados con S como SA son tipos que no pueden duplicarse, de manera que los tipos como SA se comportan como las fórmulas de la lógica lineal que *no* están marcados con la modalidad ! mientras que los tipos básicos, que sí pueden duplicarse, lo hacen como aquellas con la modalidad !.

Otra forma diferente, que permite duplicar tipos *básicos* pero prohíbe hacerlo con superposiciones, consiste en definir las reglas de reducción de forma que esa función en lugar de aplicarse a una superposión reemplazando la misma en el cuerpo de la abstración (es decir en este ejemplo en $x \otimes x$), se distribuye primero entre los términos de la superposición. Es decir, si vamos a aplicar $\lambda x.x \otimes x$ al qubit $\alpha |0\rangle + \beta |1\rangle$, en lugar de

$$(\alpha |0\rangle + \beta |1\rangle) \otimes (\alpha |0\rangle + \beta |1\rangle)$$

el término reducido será en cambio

$$\alpha(\lambda x.x \otimes x) |0\rangle + \beta(\lambda x.x \otimes x) |1\rangle$$

y luego

$$\alpha(|0\rangle \otimes |0\rangle) + \beta(|1\rangle \otimes |1\rangle)$$

.

Como resultado, entonces, todos los tipos "superpuestos" son lineales (en el sentido de que no se pueden duplicar, como en la lógica lineal) mientras que los tipos básicos que representan la base computacional (es decir $\mathbb{B}^n$) sí pueden serlo. Y para que la duplicación de una variable de tipo $\mathbb{B}^n$ no traiga problemas, las reglas de beta reducción exigen que una abstracción que tome una variable de este tipo como argumento no reduce al ser aplicada a una superposición sino que, por un lado, el argumento debe estar reducido (a un término básico) y por otra parte, se debe distribuir la abstracción usando antes las reglas de distribución lineales.

Pero por otra parte, también se quiere incluir funciones que puedan tomar superposiciones, de forma de poder expresar por ejemplo términos como $\lambda x.\pi x$ donde πx represente la medición. Para estas situaciones es que se usan dos reglas diferentes de β-reducción una que espera superposiciones y otra que no. Esto se indica anotando el tipo de la variable sobre la que se está abstrayendo en una expresión lambda.

La gramática de tipos de Lambda-$\mathcal{S}$ está presentada en la Figura 3.1 y en la Figura 3.2 la de términos.

$$\begin{array}{lll} \Psi & := \mathbb{B}^n \mid S\Psi \mid \Psi \times \Psi & (\mathcal{Q}) \\ A & := \Psi \mid \Psi \Rightarrow A \mid SA & (\mathcal{T}) \end{array}$$

Figura 3.1: Gramática de tipos de Lambda-$\mathcal{S}$.

$$\begin{array}{lll} b & := x \mid \lambda x{:}\Psi.t \mid |0\rangle \mid |1\rangle \mid ?t{\cdot}t \mid b \times b & \text{Términos básicos (B)} \\ v & := b \mid (v+v) \mid \vec{0}_{SA} \mid \alpha.v \mid v \times v & \text{Valores (V)} \\ t & := v \mid tt \mid (t+t) \mid \pi_j t \mid \alpha.t \mid t \times t \mid \mathit{head}\, t \mid \mathit{tail}\, t \mid \Uparrow_r t \mid \Uparrow_\ell t & \text{Términos } (\Lambda) \\ p & := \{\mathtt{p}_1\}t_1 \parallel \cdots \parallel \{\mathtt{p}_n\}t_n & \text{Distribución (P)} \end{array}$$

donde $\alpha \in \mathbb{C}$, $\mathtt{p}_i \in [0,1] \subseteq \mathbb{R}$ y $\sum_i p_i = 1$

Figura 3.2: Gramática de términos de Lambda-$\mathcal{S}$.

Además de incluir, como el cálculo lambda presentado en los preliminares, variables, abstracciones, aplicaciones y pares, se agregan los términos $|0\rangle$ y $|1\rangle$ para representar sendos qubits, el operador ternario $(? \cdot)$ que es un condicional que se aplica a un booleano, la combinación lineal de términos, la proyección π_j que opera sobre los primeros j qubits, el vector nulo $\vec{0}_{SA}$ para cada tipo superpuesto SA y las funciones de *casteo* $\Uparrow_r$ y $\Uparrow_\ell$ que permiten transformar la superposición de un producto entre superposiciones en la superposición de un producto. Por último, la notación $\{\mathtt{p}_1\}t_1 \parallel \cdots \parallel \{\mathtt{p}_n\}t_n$ es una distribución que a cada t_i le asigna una probabilidad p_i.

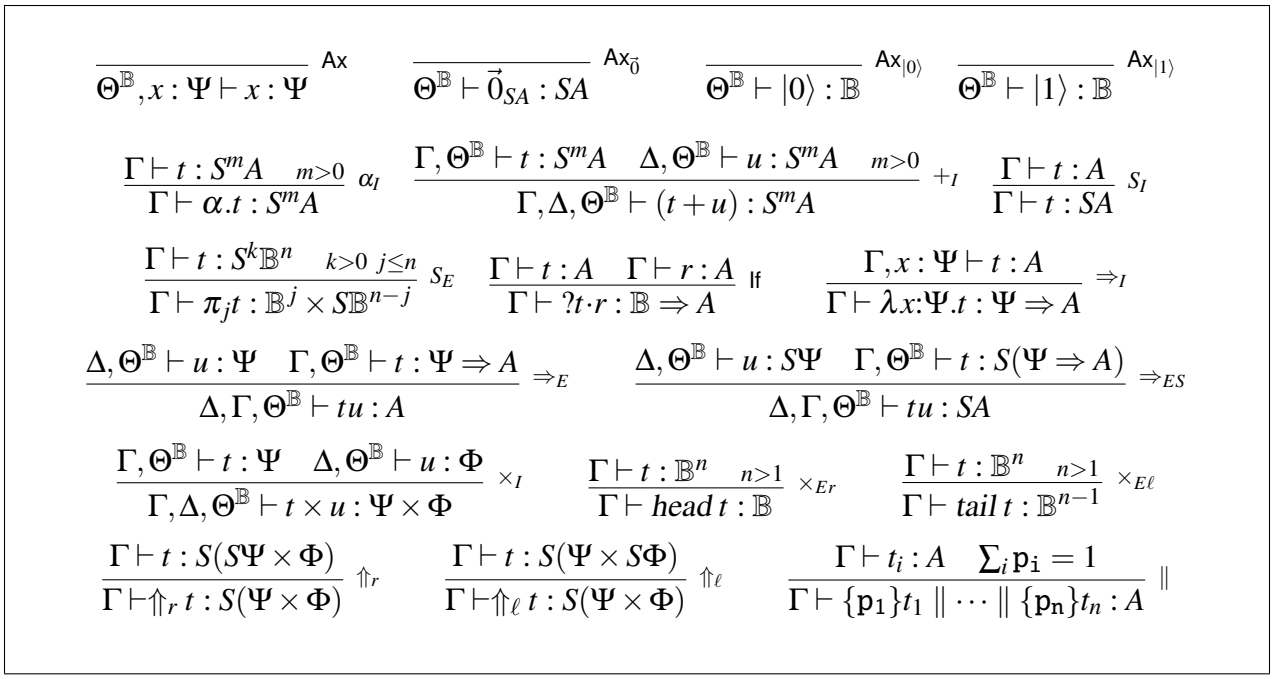

Figura 3.3: Reglas de tipado

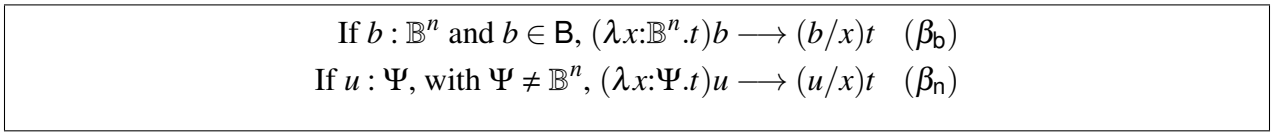

Figura 3.4: Reglas de reducción Beta

Las reglas de tipado están presentadas en la Figura 3.3.

La notación $\Theta^{\mathbb{B}}$ para contextos de tipado representa sólo aquellos en que los tipos asignados son todos de la forma $\mathbb{B}^n = \mathbb{B} \times \cdots \times \mathbb{B}$, es decir sin superposiciones. Además, por convención, cuando en una regla aparecen dos contextos de tipado sus dominios separados por ‘, ‘ se consideran disjuntos. De este modo, por ejemplo, en la regla $\times_I$, una variable que está en Γ no puede estar en Δ ni vice-versa. Por tanto, tal regla no permite duplicar una variable que aparezca en alguno de esos dos contextos, aunque sí lo permite para alguna que esté en $\Theta^{\mathbb{B}}$.

Las reglas de reducción se presentan agrupadas en las Figuras 3.4 a 3.11, en las cuales se usa la notación $t : A$ para decir que $\Gamma \vdash t : A$ para algún Γ y $t \not: A$ para decir que no existe tal Γ. En total son 50.

Hay dos reglas de reducción beta: (β_{b}) y (β_{n}). La segunda sigue la estrategia *call-by-name*. En este caso, el argumento es de tipo superpuesto y se reduce primero sustituyendo el párametro en el cuerpo de la abstracción. La segunda sigue la estrategia *call-by-value*. Acá, se espera un argumento de tipo $\mathbb{B}^n$, o sea no superpuesto, pero además que sea un término básico. Por lo tanto, primero debe reducirse el argumento y la reducción beta sólo será posible si el tipo es el esperado, por lo que si es superpuesto primero se tendrá que usar alguna de las reglas de la Figura 3.5 para distribuir linealmente la abstracción.

En la regla (proj), presente en la Figura 3.10, $j \leq n$, y se usan las siguientes notaciones:

$$
\begin{array}{rl}
\text{If } t : \mathbb{B}^n \Rightarrow A,\ t(u+v) \longrightarrow (tu+tv) & (\mathsf{lin}_{\mathsf{r}}^{+}) \\
\text{If } t : \mathbb{B}^n \Rightarrow A,\ t(\alpha.u) \longrightarrow \alpha.tu & (\mathsf{lin}_{\mathsf{r}}^{\alpha}) \\
\text{If } t : \mathbb{B}^n \Rightarrow A,\ t\vec{0}_{S\mathbb{B}^n} \longrightarrow \vec{0}_{SA} & (\mathsf{lin}_{\mathsf{r}}^{0}) \\
(t+u)v \longrightarrow (tv+uv) & (\mathsf{lin}_{\ell}^{+}) \\
(\alpha.t)u \longrightarrow \alpha.tu & (\mathsf{lin}_{\ell}^{\alpha}) \\
\vec{0}_{S(\mathbb{B}^n \Rightarrow A)}t \longrightarrow \vec{0}_{SA} & (\mathsf{lin}_{\ell}^{0})
\end{array}
$$

Figura 3.5: Reglas de distribución lineal

$$
(?\, t \cdot r)\,|1\rangle \longrightarrow t \quad (\mathsf{if}_1) \qquad\qquad (?\, t \cdot r)\,|0\rangle \longrightarrow r \quad (\mathsf{if}_1)
$$

Figura 3.6: Reglas del operador ternario

$$
\begin{array}{rl}
\text{If } h \neq u \times v \text{ and } h \in \mathsf{B},\ \mathit{head}\,(h \times t) \longrightarrow h & (\mathsf{head}) \\
\text{If } h \neq u \times v \text{ and } h \in \mathsf{B},\ \mathit{tail}\,(h \times t) \longrightarrow t & (\mathsf{tail})
\end{array}
$$

Figura 3.7: Reglas para listas

$$
\begin{array}{rl}
(\vec{0}_{SA}+t) \longrightarrow t & (\mathsf{neutral}) \\
1.t \longrightarrow t & (\mathsf{unit}) \\
\text{If } \left\{ \begin{array}{c} t : A,\ \text{with } A \in \mathcal{B},\ \text{or} \\ t : SA,\ \text{and } t \not: A \end{array} \right\},\ 0.t \longrightarrow \vec{0}_{SA} & (\mathsf{zero}_{\alpha}) \\
\alpha.\vec{0}_{SA} \longrightarrow \vec{0}_{SA} & (\mathsf{zero}) \\
\alpha.(\beta.t) \longrightarrow (\alpha\beta).t & (\mathsf{prod}) \\
\alpha.(t+u) \longrightarrow (\alpha.t+\alpha.u) & (\alpha\mathsf{dist}) \\
(\alpha.t+\beta.t) \longrightarrow (\alpha+\beta).t & (\mathsf{fact}) \\
(\alpha.t+t) \longrightarrow (\alpha+1).t & (\mathsf{fact}^1) \\
(t+t) \longrightarrow 2.t & (\mathsf{fact}^2)
\end{array}
$$

Figura 3.8: Reglas que implementan los axiomas de los espacios vectoriales

$$
\begin{array}{rll}
\Uparrow_r (r+s)\times u \longrightarrow (\Uparrow_r r\times u+\Uparrow_r s\times u) & (\mathsf{dist}^{+}_{\mathsf{r}}) \\
\Uparrow_\ell u\times(r+s) \longrightarrow (\Uparrow_\ell u\times r+\Uparrow_\ell u\times s) & (\mathsf{dist}^{+}_{\ell}) \\
\Uparrow_r (\alpha.r)\times u \longrightarrow \alpha.\ \Uparrow_r r\times u & (\mathsf{dist}^{\alpha}_{\mathsf{r}}) \\
\Uparrow_\ell u\times(\alpha.r) \longrightarrow \alpha.\ \Uparrow_r u\times r & (\mathsf{dist}^{\alpha}_{\ell}) \\
\text{If } u \text{ has type } \Psi,\ \Uparrow_r \vec{0}_{S\Phi}\times u \longrightarrow \vec{0}_{S(\Phi\times\Psi)} & (\mathsf{dist}^{\mathsf{0}}_{\mathsf{r}}) \\
\text{If } u \text{ has type } \Psi,\ \Uparrow_\ell u\times\vec{0}_{S\Phi} \longrightarrow \vec{0}_{S(\Psi\times\Phi)} & (\mathsf{dist}^{\mathsf{0}}_{\ell}) \\
\Uparrow (t+u) \longrightarrow (\Uparrow t+\Uparrow u) & (\mathsf{dist}^{+}_{\Uparrow}) \\
\Uparrow (\alpha.t) \longrightarrow \alpha.\ \Uparrow t & (\mathsf{dist}^{\alpha}_{\Uparrow}) \\
\Uparrow_r \vec{0}_{S(S(S\Psi)\times\Phi)} \longrightarrow \Uparrow_r \vec{0}_{S(S\Psi\times\Phi)} & (\mathsf{dist}^{\mathsf{0}}_{\Uparrow_{\mathsf{r}}}) \\
\Uparrow_\ell \vec{0}_{S(\Psi\times S(S\Phi))} \longrightarrow \Uparrow_\ell \vec{0}_{S(\Psi\times S\Phi)} & (\mathsf{dist}^{\mathsf{0}}_{\Uparrow_\ell}) \\
\Uparrow_r \vec{0}_{S(S\mathbb{B}^n\times\Phi)} \longrightarrow \vec{0}_{S(\mathbb{B}^n\times\Phi)} & (\mathsf{neut}^{\Uparrow}_{\mathsf{0r}}) \\
\Uparrow_\ell \vec{0}_{S(\Psi\times S\mathbb{B}^n)} \longrightarrow \vec{0}_{S(\Psi\times\mathbb{B}^n)} & (\mathsf{neut}^{\Uparrow}_{\mathsf{0}\ell}) \\
\text{If } u\in \mathsf{B},\ \Uparrow_r u\times v \longrightarrow u\times v & (\mathsf{neut}^{\Uparrow}_{\mathsf{r}}) \\
\text{If } v\in \mathsf{B},\ \Uparrow_\ell u\times v \longrightarrow u\times v & (\mathsf{neut}^{\Uparrow}_{\ell})
\end{array}
$$

Figura 3.9: Reglas de casteo para $\Uparrow_r$ y $\Uparrow_\ell$

$$
\begin{array}{rl}
\pi_j(\sum_{i=1}^{m}[\alpha_i.]\prod_{h=1}^{n}|b_{hi}\rangle) \longrightarrow \Big\|_{k=0}^{2^j-1}\{\mathsf{p_k}\}(|k\rangle\times|\phi_k\rangle) & (\mathsf{proj}) \\
\pi_j\vec{0}_{S\mathbb{B}^n} \longrightarrow |0\rangle^{\times n} & (\mathsf{proj}_{\vec{0}})
\end{array}
$$

Figura 3.10: Reglas para la proyección

$[\alpha.]t$ puede ser tanto t como $\alpha.t$ (si α no está presente, entonces equivale a 1).

$|k\rangle = \left|b_1\ldots b_j\right\rangle$ donde $b_1\ldots b_j$ es la representación binaria de k

$$|\phi_k\rangle = \sum_{i\in T_k}\left(\frac{\alpha_i}{\sqrt{\sum_{r\in T_k}|\alpha_r|^2}}\right)\prod_{h=j+1}^{n}|b_{hi}\rangle$$

$$\mathsf{p_k} = \sum_{i\in T_k}\left(\frac{|\alpha_i|^2}{\sum_{r=1}^{m}|\alpha_r|^2}\right)$$

$$T_k = \{i\le m \mid \left|b_{1i}\ldots b_{ji}\right\rangle = |k\rangle\}$$

De este modo $|k\rangle\times|\phi_k\rangle$ es la k-ésima proyección normalizada del término.

A. Díaz-Caro, G. Dowek y J.P. Rinaldi probaron el teorema de *subject reduction* para términos

If $t \longrightarrow u$, then

$$
\begin{array}{ccc}
tv \longrightarrow uv & (\lambda x^{\mathbb{B}^n}.v)t \longrightarrow (\lambda x^{\mathbb{B}^n}.v)u & (t+v) \longrightarrow (u+v) \\
\alpha.t \longrightarrow \alpha.u & \pi_j t \longrightarrow \pi_j u & t \times v \longrightarrow u \times v \\
v \times t \longrightarrow v \times u & \Uparrow_r t \longrightarrow \Uparrow_r u & \Uparrow_\ell t \longrightarrow \Uparrow_\ell u \\
head\, t \longrightarrow head\, u & tail\, t \longrightarrow tail\, u & t?r{\cdot}s \longrightarrow u?r{\cdot}s \\
(\{\mathtt{p}_1\}t_1 \parallel \cdots \parallel \{\mathtt{p}_k\}t \parallel \cdots \parallel \{\mathtt{p}_n\}t_n) & \longrightarrow & (\{\mathtt{p}_1\}t_1 \parallel \cdots \parallel \{\mathtt{p}_k\}u \parallel \cdots \parallel \{\mathtt{p}_n\}t_n)
\end{array}
$$

Figura 3.11: Reglas contextuales

cerrados en Lambda-$\mathcal{S}$ [9, T.2.8] (ver también [6, T.5.12]) así como la normalización fuerte [9, T.2.9] (ver también [6, T.6.10]). A. Díaz-Caro y O. Malherbe probaron el teorema de progreso [9, T.2.10], es decir, que todo término de Lambda-$\mathcal{S}$ que no es un valor, reduce.

A continuación vemos la implementación del algoritmo de teleportación usando Lambda-$\mathcal{S}$.

Ejemplo 3.0.1 (Telep)**.** Para definir Telep primero definimos algunos términos auxiliares para simplificar la expresión.

$$
\begin{aligned}
\mathsf{not} &= (?\ |0\rangle \cdot |1\rangle) \\
\mathsf{cnot} &= \lambda x : \mathbb{B}^2 . (head\ x) \times (?\ \mathsf{not}\ (head\ x) \cdot (tail\ x)) \\
\mathsf{cnot}^3_{12} &= \lambda x : \mathbb{B}^3 . (\mathsf{cnot}\ (head\, x \times (tail\ head\, x)) \times (tail\ tail\, x) \\
\mathsf{H} &= (?\ |+\rangle \cdot |-\rangle) \\
\mathsf{H}^3_1 &= \lambda x : \mathbb{B}^3 . (\mathsf{H}\,(head\, x)) \times (tail\, x)) \\
\mathsf{Z} &= (?\ -|1\rangle \cdot |0\rangle) \\
\mathsf{AppIf} &= \lambda f : \mathbb{B} \to [S]\mathbb{B} . (?\ f \cdot I)
\end{aligned}
$$

not, cnot, H y Z corresponden a las implementaciones de las compuertas cuánticas ya vistas. cnot^3_{12} aplica cnot a los primeros dos qubits de una producto tensorial de 3 mientras que H^3_1 Hadamard al primero. AppIf recibe una función (correspondiente a una operación unitaria) y un qubit de la base y devuelve esa misma función o la identidad según cuál sea su valor. Es decir, se aplica o no un operador de acuerdo al valor del otro parámetro. Con estos términos definimos las partes de Telep:

$$\mathsf{Alice} = \lambda x : S\mathbb{B} \times S\mathbb{B}^2 . \pi_2(\Uparrow_r \mathsf{H}^3_1(\mathsf{cnot}^3_{12}\ \ \Uparrow_r \Uparrow_l x))$$

$$\mathsf{Bob} = \lambda x : \mathbb{B}^3 . (\mathsf{AppIf}\ \mathsf{Z}\ (head\ x))((\mathsf{AppIf}\ \mathsf{not}\ (head\ tail\ x))(tail\ tail\ x)))$$

Y por último:

$$\mathsf{Telep} = \lambda x : S\mathbb{B} . \mathsf{Bob}(\Uparrow_r \mathsf{Alice}(x \times (|\beta_{00}\rangle)))$$

Lambda-$\mathcal{S}$ permite definir funciones unitarias así como aplicar medición cuántica a los términos. Sin embargo no asegura que los términos de tipo $S\mathbb{B}$ correspondan a qubits, es decir, con norma igual a 1. Esto ocurre ya con el término $\vec{0}_{S\mathbb{B}}$, pero además al admitir funciones que no son unitarias permite aplicar transformaciones a qubits para dar lugar a estados que no lo son.

Por ejemplo el término $\lambda x : S\mathbb{B} \,.\, 2 \cdot x$, de tipo $S\mathbb{B} \to S\mathbb{B}$ aplicado a $|0\rangle$ resulta en $2 \cdot |0\rangle$, que no tiene norma 1 y aplicado a $|+\rangle$, $2 \cdot |0\rangle$. También se admiten funciones como $\lambda x : S\mathbb{B} \,.\, (?\ |+\rangle \cdot |+\rangle)x$ que aplicada a $|0\rangle$ o $|1\rangle$ dan $|+\rangle$, pero aplicada a $|+\rangle$ resulta en $|0\rangle + |1\rangle$, que tampoco tiene norma 1. Es decir que para restringir los términos de tipo $S\Psi$ adecuadamente también las funciones, de tipo $S\Psi \to S\Psi$, tienen que ser isometrías.

Capítulo 4

Lambda-$\mathcal{S}_1$

Lambda-$\mathcal{S}$ admite superposiciones cuya norma no es igual a uno. Por ejemplo, podemos tipar correctamente $(|0\rangle + |1\rangle)$ de este modo:

$$\dfrac{\dfrac{\dfrac{}{\vdash |0\rangle : \mathbb{B}}\,Ax_{|0\rangle}}{\vdash |0\rangle : S\mathbb{B}}\,S_I \qquad \dfrac{\dfrac{}{\vdash |1\rangle : \mathbb{B}}\,Ax_{|1\rangle}}{\vdash |1\rangle : S\mathbb{B}}\,S_I}{\vdash (|0\rangle + |1\rangle) : S\mathbb{B}}\,+_I$$

Y sin embargo su norma no equivale a uno. Lambda-$\mathcal{S}_1$ [7, 10], en cambio, busca limitar los términos a aquellos cuya norma sí valga uno, rechazando términos como ese. En contraste con Lambda-$\mathcal{S}$, por otra parte, comprende un número más reducido de reglas ya que muchos términos como $(\alpha + \beta) \cdot \vec{t}$ o como $t\vec{s}$ que tienen sus correspondientes en el otro sistema y que llevan a la introducción de numerosas reglas de reescritura, en Lambda-$\mathcal{S}_1$ no pertenecen a la gramática de términos. No obstante, se incluyen un conjunto de reglas de congruencia y notaciones para construcciones lineales. Así es como el primer término resulta congruente a $\alpha \cdot \vec{t} + \beta \cdot \vec{t}$ mientras que el segundo es una notación para $\sum_{l=1}^{q} \delta_l \cdot t s_l$ donde $\vec{s} = \sum_{l=1}^{q} s_l$.

Otra diferencia relevante con respecto a Lambda-$\mathcal{S}$ es que Lambda-$\mathcal{S}_1$ no incluye un operador para la medición y las funciones distribuyen linealmente siempre de modo que no pueden tomar un argumento de tipo superpuesto en la reducción beta.

Los términos de este lenguaje se presentan en la Tabla 4.1. Estos se dividen en *valores* y *términos*. Los valores son términos pero éstos últimos incluyen también a las aplicaciones, el operador de secuenciación (;), los match y let. Asimismo, ambos grupos se dividen entre valores (o términos) *puros* o *distribuciones*, que son las combinaciones lineales de los primeros.

Aquí presentaremos el cálculo de Lambda-$\mathcal{S}_1$ de [10], el cual es un subcálculo de aquél definido en [7].

Las reglas de congruencia están en la Tabla 4.2 y las notaciones para las construcciones lineales en 4.3

El conjunto de valores en el sistema Lambda-$\mathcal{S}_1$ no forma un espacio vectorial dado que su gramática no incluye ningún vector nulo. En [7] se utiliza el término *distribuciones* (en oposición a *combinaciones lineales* de la cual son una noción más restringida) para referirse al conjunto de términos de la gramática, junto a los que surgen de las reglas de congruencia.

$$
\begin{array}{llll}
\text{Valores puros} & v,w & ::= & x \mid \lambda x.\vec{s} \mid * \mid (v_1,v_2) \mid \mathtt{inl}(v) \mid \mathtt{inr}(v) \\
\text{Términos puros} & s,t & ::= & v \mid st \mid t;\vec{s} \mid \mathtt{let}\ (x_1,x_2) = t\ \mathtt{in}\ \vec{s} \mid \mathtt{match}\ t\ \{\mathtt{inl}(x_1) \mapsto \vec{s}_1 \mid \mathtt{inr}(x_2) \mapsto \vec{s}_2\} \\
\text{Distrib. de valores} & \vec{v},\vec{w} & ::= & v \mid \vec{v}+\vec{w} \mid \alpha\cdot\vec{v} \qquad (\alpha \in \mathbb{C}) \\
\text{Distrib. de términos} & \vec{s},\vec{t} & ::= & t \mid \vec{s}+\vec{t} \mid \alpha\cdot\vec{t} \qquad (\alpha \in \mathbb{C})
\end{array}
$$

Tabla 4.1: Gramática de términos

$$
\begin{array}{lllllllll}
\vec{t}_1+\vec{t}_2 & \equiv & \vec{t}_2+\vec{t}_1 & \quad (\vec{t}_1+\vec{t}_2)+\vec{t}_3 & \equiv & \vec{t}_1+(\vec{t}_2+\vec{t}_3) & \quad 1\cdot\vec{t} & \equiv & \vec{t} \\
\alpha\cdot(\beta\cdot\vec{t}) & \equiv & \alpha\beta\cdot\vec{t} & \quad (\alpha+\beta)\cdot\vec{t} & \equiv & \alpha\cdot\vec{t}+\beta\cdot\vec{t} & \quad \alpha\cdot(\vec{t}_1+\vec{t}_2) & \equiv & \alpha\cdot\vec{t}_1+\alpha\cdot\vec{t}_2
\end{array}
$$

Tabla 4.2: Reglas de congruencia en distribuciones de términos

En [10] se define la noción de *espacio de acción distribuida* (distributive action-space), a saber un semigrupo conmutativo equipado con una multiplicación escalar que cumple con la asociatividad así como con la distributividad del escalar respecto a la suma de vectores y de un vector respecto a la suma de escalares, y tiene un elemento neutro, 1.

En cuanto a la noción de *ortogonalidad*, en [10] se la define en base a lo que ahí se llama *pseudo producto interno* y se define del siguiente modo:

Definición 4.0.1 (Pseudo producto interno y ortogonalidad)**.** Sean $\vec{v} = \sum_{i=1}^{n} \alpha_i \cdot v_i$ y $\vec{w} = \sum_{j=1}^{m} \beta_j \cdot w_j$ dos distribuciones de valores en forma canónica. Entonces definimos el pseudo producto interno $(\cdot \mid \cdot) : \vec{\mathbb{V}} \times \vec{\mathbb{V}} \to \mathbb{C}$ como

$$(\vec{v} \mid \vec{w}) := \sum_{i=1}^{n} \sum_{j=1}^{m} \bar{\alpha}_i \beta_j \delta_{v_i,w_j}$$

donde δ_{v_i,w_j} es la delta de Kronecker. Escribimos $\vec{v} \perp \vec{w}$ si $(\vec{v} \mid \vec{w}) = 0$.

Los tipos son los producidos por la siguiente gramática:

$$A := \mathbb{U} \mid \sharp A \mid A + A \mid A \times A \mid A \to A$$

Esta gramática incluye el símbolo $\sharp$ que cumple la misma función que la de S en los tipos de Lambda-$\mathcal{S}$.

En la Tabla 4.5 se presentan las reglas de subtipado y las reglas de tipado están en la Tabla 4.6, donde se usa la notación:

$\Gamma \vdash (\Delta_1 \vdash \vec{v}_1 \perp \Delta_2 \vdash \vec{v}_2) : A$ para

$$\begin{cases} \Gamma, \Delta_1 \vdash \vec{v}_1 : A \\ \Gamma, \Delta_2 \vdash \vec{v}_2 : A \\ \theta_{\Gamma,\Delta_1}(\vec{v}_1) \perp \theta_{\Gamma,\Delta_2}(\vec{v}_2) \end{cases}$$

$$(\vec{v},\vec{w}) := \sum_{i=1}^{n}\sum_{j=1}^{k} \alpha_i\beta_j \cdot (v_i, w_j) \qquad \mathtt{inl}(\vec{v}) := \sum_{i=1}^{n} \alpha_i \cdot \mathtt{inl}(v_i) \qquad \mathtt{inr}(\vec{v}) := \sum_{i=1}^{n} \alpha_i \cdot \mathtt{inr}(v_i)$$

$$t\,\vec{s} := \sum_{\ell=1}^{q} \delta_\ell \cdot t s_\ell \qquad\qquad \vec{t};\vec{s} := \sum_{k=1}^{p} \gamma_k \cdot (t_k;\vec{s})$$

$$\mathtt{let}\ (x,y) = \vec{t}\ \mathtt{in}\ \vec{s} := \sum_{k=1}^{p} \gamma_k \cdot \big(\mathtt{let}\ (x,y) = t_k\ \mathtt{in}\ \vec{s}\big)$$

$$\mathtt{match}\,\vec{t}\,\{\mathtt{inl}(x_1) \mapsto \vec{s}_1 \mid \mathtt{inr}(x_2) \mapsto \vec{s}_2\} := \sum_{k=1}^{p} \gamma_k \cdot \big(\mathtt{match}\,t_k\,\{\mathtt{inl}(x_1) \mapsto \vec{s}_1 \mid \mathtt{inr}(x_2) \mapsto \vec{s}_2\}\big)$$

Donde $\vec{v} = \sum_{i=1}^{n} \alpha_i \cdot v_i$, $\vec{w} = \sum_{j=1}^{m} \beta_j \cdot w_j$, $\vec{t} = \sum_{k=1}^{p} \gamma_k \cdot t_k$, y $\vec{s} = \sum_{\ell=1}^{q} \delta_\ell \cdot s_\ell$.

Tabla 4.3: Notaciones para construcciones lineales

donde para cualquier contexto Γ, θ_Γ es una sustitución de variables por valores puros del mismo tipo. Si $\Delta_1 = \Delta_2 = \emptyset$, entonces podemos escribir solo $\Gamma \vdash (\vec{v}_1 \perp \vec{v}_2) : A$.

También se escribe $A^\flat$ para indicar que el tipo A no contiene ningún $\sharp$ (salvo tal vez a la derecha de una flecha). Así las reglas Weak y Contr permiten duplicar variables siempre que su tipo sea $A^\flat$.

A. Díaz-Caro y O. Malherbe prueban el teorema de progreso para Lambda-$\mathcal{S}_1$ [10, T.3.1] así como subject reduction [10, T.3.3] y normalización fuerte [10, Coro.3.4.1]. Este último se basa en el modelo de realizabilidad dado en [7].

En [10] también se incluye un teorema de *expresividad* que muestra que el cálculo lambda simplemente tipado extendido con pares y sumas está incluido en Lambda-$\mathcal{S}_1$ y que cualquier isometría puede definirse en el sistema. También se prueba el siguiente

Teorema 4.0.1. [10, T.3.12] Una abstracción $\lambda x.\vec{t}$ es un valor de tipo $\sharp\mathbb{B} \to \sharp\mathbb{B}$ si y solo si representa una isometría $U : \mathbb{C}^2 \to \mathbb{C}^2$.

A diferencia de lo que ocurre con Lambda-$\mathcal{S}$, la gramática de Lambda-$\mathcal{S}_1$ no incluye los términos $|0\rangle$ y $|1\rangle$. Sin embargo, los qubits de la base computacional pueden codificarse con los valores $\mathtt{inl}(*)$ y $\mathtt{inr}(*)$, representando respectivamente $|0\rangle$ y $|1\rangle$.

En Lambda-$\mathcal{S}_1$ pueden, como se vió en Lambda-$\mathcal{S}$, definirse cnot, Z, H: $\mathsf{Z} = \lambda x.\,\mathtt{match}\,x\,\{\mathtt{inl}(*) \mapsto -|1\rangle \mid \mathtt{inr}(*) \mapsto |0\rangle\}$, $\mathsf{H} = \lambda x.\,\mathtt{match}\,x\,\{\mathtt{inl}(*) \mapsto |+\rangle \mid \mathtt{inr}(*) \mapsto |-\rangle\}$ y

$$\begin{aligned}\mathsf{cnot} = \lambda x.\,&\mathtt{let}\ (x_0,x_1) = x\ \mathtt{in}\\ &\mathtt{match}\,x_0\,\{\mathtt{inl}(*) \mapsto \mathtt{match}\,x_1\,\{\mathtt{inl}(*) \mapsto |00\rangle \mid \mathtt{inr}(*) \mapsto |01\rangle\}\\ &\qquad \mid \mathtt{inr}(*) \mapsto \mathtt{match}\,x_1\,\{\mathtt{inl}(*) \mapsto |11\rangle \mid \mathtt{inr}(*) \mapsto |10\rangle\}\\ &\}\end{aligned}$$

De este modo, usando let podemos definir las funciones auxiliares necesarias para definir Telep. Sin embargo, debido a que no existe operador π ni nada que represente la medición, no podemos

$$(\lambda x.\vec{t})\, v \longrightarrow \vec{t}[x := v]$$

$$*;\vec{s} \longrightarrow \vec{s}$$

$$\texttt{let}\ (x,y) = (v,w)\ \texttt{in}\ \vec{s} \longrightarrow \vec{s}[x := v, y := w]$$

$$\texttt{match}\ \texttt{inl}(v)\ \{\texttt{inl}(x_1) \mapsto \vec{s}_1 \mid \texttt{inr}(x_2) \mapsto \vec{s}_2\} \longrightarrow \vec{s}_1[x_1 := v]$$

$$\texttt{match}\ \texttt{inr}(v)\ \{\texttt{inl}(x_1) \mapsto \vec{s}_1 \mid \texttt{inr}(x_2) \mapsto \vec{s}_2\} \longrightarrow \vec{s}_2[x_2 := v]$$

$$\frac{t \longrightarrow \vec{r}}{s\,t \longrightarrow s\,\vec{r}} \qquad \frac{t \longrightarrow \vec{r}}{t\,v \longrightarrow \vec{r}\,v} \qquad \frac{t \longrightarrow \vec{r}}{t;\vec{s} \longrightarrow \vec{r};\vec{s}} \qquad \frac{t \longrightarrow \vec{r}}{\texttt{let}\ (x,y) = t\ \texttt{in}\ \vec{s} \longrightarrow \texttt{let}\ (x,y) = \vec{r}\ \texttt{in}\ \vec{s}}$$

$$\frac{t \longrightarrow \vec{r}}{\texttt{match}\ t\ \{\texttt{inl}(x_1) \mapsto \vec{s}_1 \mid \texttt{inr}(x_2) \mapsto \vec{s}_2\} \longrightarrow \texttt{match}\ \vec{r}\ \{\texttt{inl}(x_1) \mapsto \vec{s}_1 \mid \texttt{inr}(x_2) \mapsto \vec{s}_2\}}$$

$$\frac{t \longrightarrow \vec{r}}{\alpha \cdot t + \vec{s} \longrightarrow \alpha \cdot \vec{r} + \vec{s}}$$

Tabla 4.4: Reglas de reescritura

$$\frac{}{A \leq A} \qquad \frac{}{A \leq \sharp A} \qquad \frac{}{\sharp\sharp A \leq \sharp A}$$

$$\frac{A \leq B \quad B \leq C}{A \leq C} \qquad \frac{A \leq A' \quad B \leq B'}{A \times B \leq A' \times B'} \qquad \frac{A \leq A' \quad B \leq B'}{A + B \leq A' + B'} \qquad \frac{A \leq A' \quad B \leq B'}{A' \to B \leq A \to B'}$$

Tabla 4.5: Subtipado

$$\frac{}{x : A \vdash x : A}\ \textsf{Ax} \qquad \frac{\Gamma, x : A \vdash \vec{t} : B}{\Gamma \vdash \lambda x.\vec{t} : A \to B}\ \textsf{Lam} \qquad \frac{\Gamma \vdash t : A \to B \quad \Delta \vdash \vec{s} : A}{\Gamma, \Delta \vdash t\vec{s} : B}\ \textsf{App}$$

$$\frac{}{\vdash * : \mathbb{U}}\ \textsf{Void} \qquad \frac{\Gamma \vdash t : \mathbb{U} \quad \Delta \vdash \vec{s} : A}{\Gamma, \Delta \vdash t; \vec{s} : A}\ \textsf{PureSeq} \qquad \frac{\Gamma \vdash \vec{t} : \sharp\mathbb{U} \quad \Delta \vdash \vec{s} : \sharp A}{\Gamma, \Delta \vdash \vec{t}; \vec{s} : \sharp A}\ \textsf{UnitarySeq}$$

$$\frac{\Gamma \vdash \vec{v} : A \quad \Delta \vdash \vec{w} : B}{\Gamma, \Delta \vdash (\vec{v}, \vec{w}) : A \times B}\ \textsf{Pair}$$

$$\frac{\Gamma \vdash t : A \times B \quad \Delta, x : A, y : B \vdash \vec{s} : C}{\Gamma, \Delta \vdash \mathsf{let}\ (x, y) = t\ \mathsf{in}\ \vec{s} : C}\ \textsf{PureLet} \qquad \frac{\Gamma \vdash \vec{t} : A \otimes B \quad \Delta, x : \sharp A, y : \sharp B \vdash \vec{s} : \sharp C}{\Gamma, \Delta \vdash \mathsf{let}\ (x, y) = \vec{t}\ \mathsf{in}\ \vec{s} : \sharp C}\ \textsf{UnitaryLet}$$

$$\frac{\Gamma \vdash v : A}{\Gamma \vdash \mathsf{inl}(v) : A + B}\ \textsf{InL} \qquad \frac{\Gamma \vdash v : B}{\Gamma \vdash \mathsf{inr}(v) : A + B}\ \textsf{InR}$$

$$\frac{\Gamma \vdash t : A + B \quad \Delta \vdash (x_1 : A \vdash \vec{v}_1 \perp x_2 : B \vdash \vec{v}_2) : C}{\Gamma, \Delta \vdash \mathsf{match}\ t\ \{\mathsf{inl}(x_1) \mapsto \vec{v}_1 \mid \mathsf{inr}(x_2) \mapsto \vec{v}_2\} : C}\ \textsf{PureMatch}$$

$$\frac{\Gamma \vdash \vec{t} : A \oplus B \quad \Delta \vdash (x_1 : \sharp A \vdash \vec{v}_1 \perp x_2 : \sharp B \vdash \vec{v}_2) : \sharp C}{\Gamma, \Delta \vdash \mathsf{match}\ \vec{t}\ \{\mathsf{inl}(x_1) \mapsto \vec{v}_1 \mid \mathsf{inr}(x_2) \mapsto \vec{v}_2\} : \sharp C}\ \textsf{UnitaryMatch}$$

$$\frac{^{(k \neq h)} \quad \vdash (\vec{v}_k \perp \vec{v}_h) : A \quad \sum_{j=1}^{m} |\alpha_j|^2 = 1 \quad m \geq 1 \quad A \neq B \to C}{\vdash \sum_{j=1}^{m} \alpha_j \cdot \vec{v}_j : \sharp A}\ \textsf{Sup}$$

$$\frac{\Gamma \vdash \vec{t} : A \quad A \leq B}{\Gamma \vdash \vec{t} : B}\ \leq \qquad \frac{\Gamma \vdash \vec{t} : A \quad \vec{t} \equiv \vec{r}}{\Gamma \vdash \vec{r} : A}\ \equiv$$

$$\frac{\Gamma \vdash \vec{t} : B \quad A^\flat}{\Gamma, x : A \vdash \vec{t} : B}\ \textsf{Weak} \qquad \frac{\Gamma, x : A, y : A \vdash \vec{t} : B \quad A^\flat}{\Gamma, x : A \vdash \vec{t}[y := x] : B}\ \textsf{Contr}$$

Tabla 4.6: Sistema de tipos

implementar ese algoritmo. En el próximo capítulo damos una implementación de Telep para ejemplificar con una aplicación la contribución de Lambda-$\mathcal{S}_1^\pi$.

Capítulo 5

Lambda-$\mathcal{S}_1^\pi$

El cálculo Lambda-$\mathcal{S}_1^\pi$ es, como Lambda-$\mathcal{S}_1$, un lenguaje de control cuántico que opera en la esfera unitaria, pero incorpora, a la manera de Lambda-$\mathcal{S}$, un operador de medición.

A diferencia de Lambda-$\mathcal{S}$, las funciones no se distinguen según su parámetro sea o no una superposición. La reducción es call-by-pure-value, o sea que la reducción beta siempre reemplaza tomando un valor puro. Aplicar una función a una superposición significa, en realidad, que distribuimos una aplicación, lo cual surge de las notaciones para construcciones lineales (Tabla 5.3). Sin embargo, para ciertos programas necesitamos aplicar funciones que no se distribuyan linealmente respecto de su parámetro. Un ejemplo de esto surge al querer medir un qubit. Si el operador de medición se aplica a un parámetro y este es siempre un valor puro, entonces dicho operador carecería de utilidad, ya que medir un qubit de la base computacional dará como resultado siempre el mismo qubit. Para tales casos lo que hacemos es, siguiendo el trabajo [3], aplicar call-by-value en una abstracción que *empaqueta* una distribución, a la cual aplicamos un qubit descartable para desempaquetar.

5.1. Términos

En la Tabla 5.1 se presenta la gramática de términos de Lambda-$\mathcal{S}_1^\pi$. Esta gramática es similar a la de Lambda-$\mathcal{S}$, aunque tiene varias diferencias.

Por ejemplo Lambda-$\mathcal{S}_1^\pi$ no incluye entre sus términos a uno como $\vec{0}_{\sharp A}$, que representaría el vector nulo de tipo $\sharp A$, como lo hace Lambda-$\mathcal{S}$ con $\vec{0}_{SA}$. El vector nulo, sin embargo, puede representarse con términos como $0 \cdot |0\rangle$, si bien no se puede tipar. Tampoco incluye la función probabilística que da cuenta de los diferentes estados de una medición de manera conjunta. En cambio, recurre a un sistema de reescritura probabilístico en el cual la reducción del caso particular de la medición no es determinística y el resultado es uno solo, pero está asociado a una probabilidad. No tiene tampoco los términos *head* t ni *tail* t ya que se sirve del let para descomponer pares. Finalmente, no tiene los términos $\Uparrow_l t$ ni $\Uparrow_r t$ y utiliza las reglas de subtipado y no las reglas de *casteo*.

Los términos de Lambda-$\mathcal{S}_1^\pi$ pueden ser "puros" o "distribuciones". Las últimas son combinaciones lineales de términos puros. Usamos V para denotar el conjunto de los valores puros, $\vec{\mathsf{V}}$

$$
\begin{array}{rrcll}
\text{Pure values} & v,w & ::= & x \mid |0\rangle \mid |1\rangle \mid \lambda x\,.\,\vec{s} \mid ?\,\vec{s}\cdot\vec{t} \mid (v,w) & \\
\text{Pure terms} & s,t & ::= & v \mid st \mid (s,t) \mid \texttt{let}\ (x,y) = t\ \texttt{in}\ \vec{s} \mid \pi_l\ t & \\
\text{Value distrib.} & \vec{v},\vec{w} & ::= & v \mid \vec{v}+\vec{w} \mid \alpha\cdot\vec{v} & (\alpha\in\mathbb{C}) \\
\text{Term distrib.} & \vec{s},\vec{t} & ::= & t \mid \vec{s}+\vec{t} \mid \alpha\cdot\vec{t} & (\alpha\in\mathbb{C})
\end{array}
$$

Tabla 5.1: Gramática de térmimos

$$
\begin{array}{rclrclrcl}
\vec{t}_1+\vec{t}_2 & \equiv & \vec{t}_2+\vec{t}_1 & (\vec{t}_1+\vec{t}_2)+\vec{t}_3 & \equiv & \vec{t}_1+(\vec{t}_2+\vec{t}_3) & 0\cdot\vec{v}+\vec{t} & \equiv & \vec{t} \\
\alpha\cdot(\beta\cdot\vec{t}) & \equiv & \alpha\beta\cdot\vec{t} & (\alpha+\beta)\cdot\vec{t} & \equiv & \alpha\cdot\vec{t}+\beta\cdot\vec{t} & \alpha\cdot(\vec{t}_1+\vec{t}_2) & \equiv & \alpha\cdot\vec{t}_1+\alpha\cdot\vec{t}_2
\end{array}
$$

Tabla 5.2: Reglas de congruencia para distribución de términos

para el de las distribuciones de valores, Λ para los términos puros y $\vec{\Lambda}$ para términos distribuidos. Denotamos con $\mathcal{T}$ al conjunto de los tipos.

En la Tabla 5.2 se incluyen reglas de congruencia para distribuciones de términos. Estas reglas, similares a las presentes en Lambda-$\mathcal{S}_1$, permiten incluir en el lenguaje términos sin necesidad de complicar las reglas de reducción para cada uno de ellos, dado que son una notación de los términos a los que equivalen. Estas reglas se aplican tanto a términos como a subtérminos, de forma que permiten obtener las equivalencias $\sum_i \alpha_i(\sum_j \beta_j \vec{t_{ij}}) \equiv \sum_i\sum_j \alpha_i\beta_j\vec{t_{ij}}$ y $\sum_i\sum_j \alpha_{ij}\,|i\rangle \equiv \sum_i(\sum_j \alpha_{ij})\,|i\rangle$

Cabe mencionar que no incluímos la equivalencia $1\cdot\vec{t}\equiv\vec{t}$ ya que si bien ambos (usando las reglas de subtipado) pueden tiparse como superposiciones, incluirla daría lugar a $1\cdot|0\rangle \equiv |0\rangle$ pero mientras $|0\rangle$ puede tiparse como $\mathbb{B}$, $1\cdot|0\rangle$ no, lo cual no respetaría *subject reduction*.

También usamos las notaciones para construcciones lineales que están en la Tabla 5.3

$$
(\vec{t},\vec{s}) := \sum_{i=1}^{n}\sum_{j=1}^{k} \alpha_i\beta_j\cdot(t_i,s_j) \qquad\qquad t\vec{s} := \sum_{i=1}^{m}\alpha_i\cdot ts_i
$$

$$
\texttt{let}\ (x,y) = \vec{t}\ \texttt{in}\ \vec{s} := \sum_{k=1}^{p}\gamma_k\cdot\left(\texttt{let}\ (x,y) = t_k\ \texttt{in}\ \vec{s}\right)
$$

Donde $\vec{t} = \sum_{k=1}^{p}\gamma_k\cdot t_k$, y $\vec{s} = \sum_{\ell=1}^{q}\delta_\ell\cdot s_\ell$.

Tabla 5.3: Notaciones para construcciones lineales

Las reglas de reescritura se presentan en la Tabla 5.4, donde se usan las siguientes notaciones

$|k\rangle_l = |b_1\ldots b_l\rangle$ donde $b_1\ldots b_l$ es la representación binaria (con l bits) de k

$$(\lambda x.\vec{t})\, v \longrightarrow \vec{t}[x := v]$$
$$\texttt{let}\ (x,y) = (v,w)\ \texttt{in}\ \vec{s} \longrightarrow \vec{s}[x := v, y := w]$$
$$(?\, t \cdot r)|0\rangle \longrightarrow t$$
$$(?\, t \cdot r)|1\rangle \longrightarrow r$$
$$\pi_l \left(\sum_{i=1}^{m} \alpha_i \cdot \prod_{h=1}^{n} |b_{hi}\rangle \right) \longrightarrow_{p_k} |k\rangle_l \times |\phi_k\rangle$$

$$\frac{t \longrightarrow \vec{r}}{(t,\vec{s}) \longrightarrow (\vec{r},\vec{s})} \qquad \frac{t \longrightarrow \vec{r}}{(\vec{s},t) \longrightarrow (\vec{s},\vec{r})}$$
$$\frac{t \longrightarrow \vec{r}}{s\,t \longrightarrow s\,\vec{r}} \qquad \frac{t \longrightarrow \vec{r}}{t\,\vec{v} \longrightarrow r\,\vec{v}} \qquad \frac{t \longrightarrow \vec{r}}{\pi_l t \longrightarrow \pi_l \vec{r}}$$
$$\frac{t \longrightarrow \vec{r}}{\texttt{let}\ (x,y) = t\ \texttt{in}\ \vec{s} \longrightarrow \texttt{let}\ (x,y) = \vec{r}\ \texttt{in}\ \vec{s}} \qquad \frac{t \longrightarrow \vec{r}}{\alpha \cdot t + \vec{s} \longrightarrow \alpha \cdot \vec{r} + \vec{s}}$$

Tabla 5.4: Rewrite rules

$$|\phi_k\rangle_{n-l} = \sum_{i \in T_k} \left(\frac{\alpha_i}{\sqrt{\sum_{r \in T_k} \|\alpha_r\|^2}} \right) \prod_{h=l+1}^{n} |b_{hi}\rangle$$
$$\mathrm{p}_k = \sum_{i \in T_k} \|\alpha_i\|^2$$
$$T_k = \{ i \leq m \mid |b_{1i} \ldots b_{li}\rangle = |k\rangle_l \}$$

De este modo, $|k\rangle_l \times |\phi_k\rangle_{n-l}$ es la k-ésima proyección del término normalizada.

Las abstracciones permiten definir funciones que tomen por parámetro una distribución. Sin embargo, ante el natural problema de querer medir una distribución, no tiene sentido hacer algo como $\lambda x.\, \pi x$. Esta expresión no mide un qubit si no que es la identidad:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : \sharp\mathbb{B} \vdash x : \sharp\mathbb{B}}\,\mathsf{Ax}}{x : \sharp\mathbb{B} \vdash \pi x : \mathbb{B}}\,\pi}{x : \sharp\mathbb{B} \vdash \pi x : \sharp\mathbb{B}}\,\leq}{\vdash \lambda x.\, \pi x : \sharp\mathbb{B} \to \sharp\mathbb{B}}\,\mathsf{Lam}$$

Podemos aplicarla al vector $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, notado $|+\rangle$, usando

$$\frac{\vdash \lambda x.\, \pi x : \sharp\mathbb{B} \to \sharp\mathbb{B} \quad \vdash |+\rangle : \sharp\mathbb{B}}{\vdash (\lambda x.\, \pi x)|+\rangle : \sharp\mathbb{B}}\,\mathsf{UnitaryApp}$$

Pero $(\lambda x.\, \pi x)|+\rangle \longrightarrow^* |+\rangle$ ya que primero se distribuye linealmente. Como veremos en el Ejemplo 5.4.2, donde es necesario aplicar la medición sin distribuir primero el operador en el parámetro, es posible definir un valor que empaquete una distribución para esto.

Así como Lambda-$\mathcal{S}_1$ define el pseudo producto interno y la pseudo norma, también lo hace Lambda-$\mathcal{S}_1^\pi$. Sin embargo, mientras que el primero lo hace para valores cerrados, el segundo lo hace de fomrma más general, para términos abiertos.

Definición 5.1.1 (Pseudo producto interno y ortogonalidad)**.** Sean $\vec{v} = \sum_{i=1}^{n} \alpha_i \cdot v_i$ y $\vec{w} = \sum_{j=1}^{m} \beta_j \cdot w_j$ dos distribuciones de valores cerrados en forma canónica.

Luego, definimos el pseudo producto interno $(\cdot \mid \cdot) : \vec{\mathbb{V}} \times \vec{\mathbb{V}} \to \mathbb{C}$ como

$$(\vec{v} \mid \vec{w}) := \sum_{i=1}^{n} \sum_{j=1}^{m} \overline{\alpha}_i \beta_j \delta_{v_i, w_j}$$

donde δ_{v_i,w_j} es el delta de Kronecker, i.e. es 1 si $v_i = w_j$, y 0 si no. Decimos *pseudo* ya que no tenemos un espacio vectorial.

Vamos a definir la ortogonalidad entre dos términos de manera más general.

En primer lugar, para dos valores cerrados $\vdash \vec{v} : A$ y $\vdash \vec{w} : A$, decimos que son *ortogonales* (y lo notamos $\vec{v} \perp \vec{w}$) si $(\vec{v} \mid \vec{w}) = 0$.

Para dos términos cerrados $\vdash \vec{s} : A$ y $\vdash \vec{t} : A$ decimos que son *ortogonales* si para todo $\vec{v}$ y todo $\vec{w}$ tales que $\vec{s} \longrightarrow^* \vec{v}$ y $\vec{t} \longrightarrow^* \vec{w}$, se cumple: $(\vec{v} \mid \vec{w}) = 0$.

Por otra parte, dado un término $\vec{t}$, llamamos θ a una sustitución de las variables libres por valores cerrados, es decir $\theta(\vec{t}) = \vec{t}[x_1 := \theta(x_1)] \dots [x_n := \theta(x_n)]$ donde $FV(\vec{t}) = \{x_1, \dots, x_n\}$ y $\theta(x_1), \dots, \theta(x_n)$ son valores cerrados.

Decimos que θ es válida en Γ, y lo notamos $\theta \Vdash \Gamma$ si para toda $x : A \in \Gamma$, θ sustituye x por un valor cerrado de tipo A.

Ahora, dos términos $\Gamma, \Delta_s \vdash \vec{s} : A$ y $\Gamma, \Delta_t \vdash \vec{t} : A$ son ortogonales si para cualesquiera sustituciones θ_s y θ_t tales que $\theta_s \Vdash \Gamma, \Delta_s$ y $\theta_t \Vdash \Gamma, \Delta_t$, se cumple $(\theta_s(\vec{s}) \perp \theta_t(\vec{t}))$.

Escribimos $\Gamma \vdash (\Delta_s \vdash \vec{s} \perp \Delta_t \vdash \vec{t}) : A$ para decir que:

$$\begin{cases} \Gamma, \Delta_s \vdash \vec{s} : A \\ \Gamma, \Delta_t \vdash \vec{t} : A \\ \theta_s(\vec{s}) \perp \theta_t(\vec{t}) \qquad \forall \theta_s \Vdash \Gamma, \Delta_s \quad y \quad \forall \theta_t \Vdash \Gamma, \Delta_t. \end{cases}$$

Si $\Delta_s = \Delta_t = \emptyset$ entonces simplemente escribimos $\Gamma \vdash (\vec{s} \perp \vec{t}) : A$.

Definición 5.1.2 (Pseudo norma)**.** Sea $\vec{v}$ una distribución de valores. Definimos su pseudo norma $\| \vec{v} \|$ como

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

Nótese que para tener pseudo-norma un valor tiene que ser cerrado.

5.2. Tipos

Los tipos son los producidos por la siguiente gramática:

$$\mathcal{Q} := \mathbb{B} \mid \sharp\mathcal{Q} \mid \mathcal{Q} \times \mathcal{Q}$$

$$A := \mathcal{Q} \mid A \times A \mid A \to A$$

Escribiremos además $\mathcal{Q}_1$ para los tipos $\mathbb{B}$ y $\sharp\mathbb{B}$ y $\mathcal{Q}_n$ para cualquier tipo $\mathcal{Q}_m \times \mathcal{Q}_l$ si $m+l=n$.

Usamos la notación $A^\flat$ (A es bemol) para decir que A no contiene ningún # excepto, quizá, a la derecha de las flechas, y dado un contexto de tipado Γ usaremos la notación $\Gamma^\flat$ para decir que para cada tipo B contenido en él, $B^\flat$.

Las Tablas 5.5 y 5.6 representan las reglas de subtipado y tipado respectivamente.

$$\frac{}{A \leq A} \qquad \frac{}{\mathcal{Q} \leq \sharp\mathcal{Q}} \qquad \frac{}{\sharp\sharp\mathcal{Q} \leq \sharp\mathcal{Q}} \qquad \frac{}{\sharp\mathcal{Q}_n \times \sharp\mathcal{Q}_m \leq \sharp\mathcal{Q}_{n+m}}$$

$$\frac{A \leq B \quad B \leq C}{A \leq C} \qquad \frac{A \leq A' \quad B \leq B'}{A \times B \leq A' \times B'} \qquad \frac{A \leq A' \quad B \leq B'}{A' \to B \leq A \to B'}$$

Tabla 5.5: Subtipado

$$\frac{}{x : A \vdash x : A}\ \mathsf{Ax} \qquad \frac{}{\vdash |0\rangle : \mathbb{B}}\ \mathsf{Ax}_{|0\rangle} \qquad \frac{}{\vdash |1\rangle : \mathbb{B}}\ \mathsf{Ax}_{|1\rangle}$$

$$\frac{\Gamma, x : A \vdash \vec{t} : B}{\Gamma \vdash \lambda x.\vec{t} : A \to B}\ \mathsf{Lam} \qquad \frac{\Gamma \vdash (\vec{t} \perp \vec{s}) : A}{\Gamma \vdash\ ?\vec{t} \cdot \vec{s} : \mathbb{B} \to A}\ \mathsf{If} \qquad \frac{\Gamma \vdash (\vec{t} \perp \vec{s}) : \sharp A}{\Gamma \vdash\ ?\vec{t} \cdot \vec{s} : \sharp\mathbb{B} \to \sharp A}\ \mathsf{UnitaryIf}$$

$$\frac{\Gamma \vdash \vec{t} : A \quad \Delta \vdash \vec{s} : B}{\Gamma, \Delta \vdash (\vec{t}, \vec{s}) : A \times B}\ \mathsf{Pair}$$

$$\frac{\Gamma \vdash t : A \to B \quad \Delta \vdash \vec{s} : A \quad A^\flat}{\Gamma, \Delta \vdash t\vec{s} : B}\ \mathsf{App} \qquad \frac{\Gamma \vdash t : \sharp A \to \sharp B \quad \Delta \vdash \vec{s} : \sharp A}{\Gamma, \Delta \vdash t\vec{s} : \sharp B}\ \mathsf{UnitaryApp}$$

$$\frac{\Gamma \vdash t : A \times B \quad \Delta, x : A, y : B \vdash \vec{s} : C}{\Gamma, \Delta \vdash \mathtt{let}\ (x,y) = t\ \mathtt{in}\ \vec{s} : C}\ \mathsf{Let} \qquad \frac{\Gamma \vdash \vec{t} : \sharp(A \times B) \quad \Delta, x : \sharp A, y : \sharp B \vdash \vec{s} : \sharp C}{\Gamma, \Delta \vdash \mathtt{let}\ (x,y) = \vec{t}\ \mathtt{in}\ \vec{s} : \sharp C}\ \mathsf{UnitaryLet}$$

$$\frac{\Gamma \vdash \vec{t} : \sharp\mathbb{B}^n}{\Gamma \vdash \pi_l \vec{t} : \mathbb{B}^l \times \sharp\mathbb{B}^{n-l}}\ \pi \qquad \frac{(k \neq h)\ \ \Gamma \vdash (\vec{t}_k \perp \vec{t}_h) : \mathcal{Q} \quad \sum_{j=1}^m |\alpha_j|^2 = 1 \quad m \geq 1}{\Gamma \vdash \sum_{j=1}^m \alpha_j \cdot \vec{t}_j : \sharp\mathcal{Q}}\ \mathsf{Sup}$$

$$\frac{\Gamma \vdash \vec{t} : A \quad A \leq B}{\Gamma \vdash \vec{t} : B}\ \leq \qquad \frac{\Gamma \vdash \vec{t} : A \quad \vec{t} \equiv \vec{r}}{\Gamma \vdash \vec{r} : A}\ \equiv$$

$$\frac{\Gamma \vdash \vec{t} : B \quad A^\flat}{\Gamma, x : A \vdash \vec{t} : B}\ \mathsf{Weak} \qquad \frac{\Gamma, x : A, y : A \vdash \vec{t} : B \quad A^\flat}{\Gamma, x : A \vdash \vec{t}[y := x] : B}\ \mathsf{Contr}$$

Tabla 5.6: Sistema de tipos

Siguiendo a Lambda-$\mathcal{S}$ y a diferencia de Lambda-$\mathcal{S}_1$, las reglas de tipado de Lambda-$\mathcal{S}_1^\pi$ incluyen la regla If. Sin embargo, mientras que Lambda-$\mathcal{S}$ impone como única restricción que los posibles resultados sea del mismo tipo, Lambda-$\mathcal{S}_1^\pi$ además requiere que sean ortogonales entre sí. Es decir Lambda-$\mathcal{S}$ permite tipar $(?\ |+\rangle \cdot |1\rangle)\,|+\rangle$ ya que $\vdash |+\rangle : S\mathbb{B}$ y $?\ |+\rangle \cdot |1\rangle : S(\mathbb{B} \to S\mathbb{B})$ por las reglas If y $\mathsf{S_I}$ en la Tabla 3.3. Pero esto reduce a $\frac{1}{2}|0\rangle + \frac{1+\sqrt{2}}{2}|1\rangle$ que no tiene norma 1. Por otra parte, Lambda-$\mathcal{S}_1^\pi$ no permite tipar $?\ |+\rangle \cdot |1\rangle$ ya que $|+\rangle$ y $|1\rangle$ no son ortogonales.

Otra diferencia se da en la regla Sup ya que su contraparte en Lambda-$\mathcal{S}_1$ no admite contextos de tipado no vacíos. Esto por un lado incrementa la dificultad para una eventual implementación de realizar un chequeo de tipos de manera estática en Lambda-$\mathcal{S}_1^\pi$ ya que en general la verificación de ortogonalidad requiere reducir el término y para Lambda-$\mathcal{S}_1^\pi$ se tomó la decisión de dejar dicha prueba bajo la responsabilidad el programador, dándole más expresividad al lenguaje.

Nótese que si bien cualquier qubit puede representarse mediante una combinación de qubits de la base computacional, el sistema incluye (con las restricciones mencionadas) tipar combinaciones de qubits en general, dado que de otra forma la reducción rápidamente daría lugar a términos que no podrían tiparse.

La regla Pair permite tipar pares de superposiciones. Esto es así ya que si sólo admitiésemos las superposiciones de pares tipadas por Sup perderíamos la información de separabilidad Por ejemplo, no habría forma de tipar $(|+\rangle, |+\rangle)$ como $\sharp\mathbb{B} \times \sharp\mathbb{B}$ y sólo se podría tipar con $\sharp(\mathbb{B} \times \mathbb{B})$.

5.3. Propiedades

En adelante usaremos la notación $[\sharp]$ para indicar un $\sharp$ *consistentemente opcional*, por ejemplo, si se afirma un predicado sobre $[\sharp]A \to [\sharp]B$, entonces el predicado se afirma por un lado sobre $A \to B$ y por otro sobre $\sharp A \to \sharp B$.

Lema 5.3.1 (Generación)**.** Si $\Gamma \vdash \vec{t} : A$ entonces

- si $\vec{t} = x$ entonces
  - $\Gamma = x : B, \Delta^\flat$ y
  - $B \leq A$;
- si $\vec{t} = |0\rangle$ entonces
  - $\mathbb{B} \leq A$ y
  - $\Gamma^\flat$;
- si $\vec{t} = |1\rangle$ entonces
  - $\mathbb{B} \leq A$ y
  - $\Gamma^\flat$;
- si $\vec{t} = \lambda x.\vec{s}$ entonces

- $\Gamma, x : C \vdash \vec{s} : D$,
- $\Gamma^\flat$ y
- $C \to D \leq A$;

- si $\vec{t} =? \vec{r} \cdot \vec{s}$ entonces
  - $\Gamma \vdash (\vec{r} \perp \vec{s}) : C$ y
  - $[\sharp]\mathbb{B} \to [\sharp]C \leq A$;
- si $\vec{t} = (\vec{r}, \vec{s})$ entonces $\Gamma = \Gamma_1, \Gamma_2, \Gamma_3^\flat$ y existen dos casos.
  - Si $\vec{t}$ fue tipado con Pair:
    - $\Gamma_1, \Gamma_3 \vdash \vec{r} : B$,
    - $\Gamma_2, \Gamma_3 \vdash \vec{s} : C$,
    - $B \times C \leq A$;
  - Si fue tipado con Sup
    - $\vec{r} = \sum_{i=1}^{n_r} \alpha_i \cdot r_i$,
    - $\vec{s} = \sum_{i=1}^{n_s} \beta_i \cdot s_i$
    - $\vec{t} = \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} \alpha_i \beta_j \cdot (r_i, s_j)$,
    - $\sum_{i=1}^{n_r} \sum_{j=1}^{n_s} |\alpha_i \beta_j|^2 = 1$,
    - $(\forall_{i_1 \neq i_2 \vee j_1 \neq j_2}) \Gamma \vdash ((r_{i_1}, s_{j_1}) \perp (r_{i_2}, s_{j_2})) : \mathcal{Q}$,
    - $\sharp \mathcal{Q} \leq A$,
    - $n_r n_s \geq 1$,
- si $\vec{t} = r\vec{s}$,

  Existen tres casos.
  - Si $\vec{t}$ fue tipado con App
    - $\Gamma = \Gamma_1, \Gamma_2, \Gamma_3^\flat$,
    - $\Gamma_1, \Gamma_3 \vdash r : D \to E$,
    - $\Gamma_2, \Gamma_3 \vdash s : D$,
    - $E \leq A$;
  - Si $\vec{t}$ fue tipado con UnitaryApp
    - $\Gamma = \Gamma_1, \Gamma_2, \Gamma_3^\flat$,
    - $\Gamma_1, \Gamma_3 \vdash r : \sharp D \to \sharp E$,
    - $\Gamma_2, \Gamma_3 \vdash s : \sharp D$,
    - $\sharp E \leq A$;
  - Si fue tipado con Sup

- $\vec{s} = \sum_{i=1}^{n} \alpha_i \cdot s_i$
- $\vec{t} = \sum_{i=1}^{n} \alpha_i \cdot rs_i$,
- $\sum_{k=1}^{n} |\alpha_i|^2 = 1$,
- $(\forall_{i \neq j})\ \Gamma \vdash (rs_i \perp rs_j) : \mathcal{Q}$,
- $\sharp \mathcal{Q} \leq A$,
- $n \geq 1$,

- si $\vec{t} = \texttt{let}\ (x,y) = \vec{u}\ \texttt{in}\ \vec{s}$. Existen tres casos.
  - Si $\vec{t}$ fue tipado con Let
    - $\Gamma = \Gamma_1, \Gamma_2, \Gamma_3^\flat$,
    - $\vec{u}$ es un término puro, o sea u,
    - $\Gamma_1, \Gamma_3 \vdash u : D \times E$,
    - $\Gamma_2, \Gamma_3, x : D, y : E \vdash \vec{s} : F$,
    - $F \leq A$;
  - Si fue tipado con UnitaryLet
    - $\Gamma = \Gamma_1, \Gamma_2, \Gamma_3^\flat$,
    - $\Gamma_1, \Gamma_3 \vdash \vec{u} : \sharp(D \times E)$,
    - $\Gamma_2, \Gamma_3, x : \sharp D, y : \sharp E \vdash \vec{s} : \sharp F$,
    - $\sharp F \leq A$;
  - Si fue tipado con Sup
    - $\vec{u} = \sum_{k=1}^{n} \gamma_k \cdot u_k$
    - $\vec{t} = \sum_{k=1}^{n} \gamma_k \cdot (\texttt{let}\ (x,y) = u_k\ \texttt{in}\ \vec{s})$,
    - $\sum_{k=1}^{n} |\gamma_k|^2 = 1$,
    - $(\forall_{k \neq l})\ \Gamma \vdash (\texttt{let}\ (x,y) = u_k\ \texttt{in}\ s \perp \texttt{let}\ (x,y) = u_k\ \texttt{in}\ \vec{s}) : \mathcal{Q}$,
    - $\sharp \mathcal{Q} \leq A$,
    - $n \geq 1$,

- si $\vec{t} = \pi_l \vec{r}$ entonces
  - $\Gamma \vdash \vec{r} : \sharp \mathbb{B}^n$ y
  - $\mathbb{B}^l \times \sharp \mathbb{B}^{n-l} \leq A$.

- si $\vec{t} = \sum_i^n \alpha_i \cdot \vec{s}_i$ entonces
  - $\sum_{i=1}^{n} |\alpha_i|^2 = 1$,
  - $_{(i \neq j)}\ \Gamma \vdash (s_i \perp s_j) : D$,
  - $\sharp D \leq A$
  - $n \geq 1$,

Demostración. Usando las reglas Weak y $\leq$, siempre que haya una derivación de $\Gamma \vdash t : A$ existe una de $\Delta \vdash t : B$ donde $A \leq B$ y $(\Delta \backslash \Gamma)^\flat$, no habiendo otras reglas que cambien el tipo sin cambiar el término. De este modo, cada caso del Lema corresponde a una de las restantes reglas, con posibles usos adicionales de las reglas mencionadas.

□

A continuación probamos *subject reduction*, para lo cual necesitamos el Lema de Sustitución. En este último, el término que sustituye debe tener su contexto de tipado vacío, y en el primero también requerimos que el contexto de tipado sea vacío. Para términos con contexto de tipado no vacíos, en general, en realidad *subject reduction* es falso. Para verlo, considérese este ejemplo:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{z : \sharp\mathbb{B} \vdash z : \sharp\mathbb{B}}\,\mathsf{A_x}}{z : \sharp\mathbb{B} \vdash \pi z : \mathbb{B}}\,\pi \qquad \vdash |0\rangle : \mathbb{B}}{z : \sharp\mathbb{B} \vdash (\pi z, |0\rangle) : \mathbb{B} \times \mathbb{B}}\,\mathsf{Pair} \qquad \dfrac{\dfrac{}{y : \mathbb{B} \vdash y : \mathbb{B}}\,\mathsf{A_x}}{x : \mathbb{B}, y : \mathbb{B} \vdash y : \mathbb{B}}\,\mathsf{Weak}}{z : \sharp\mathbb{B} \vdash \mathtt{let}\ (x,y) = (\pi z, |0\rangle)\ \mathtt{in}\ y : \mathbb{B}}\,\mathsf{Let}$$

Pero

$$\mathtt{let}\ (x,y) = (\pi z, |0\rangle)\ \mathtt{in}\ y \longrightarrow^* |0\rangle$$

y sin embargo

$$z : \sharp\mathbb{B} \not\vdash |0\rangle : \mathbb{B}$$

.

Nótese que si fuese posible derivar $z : \sharp\mathbb{B} \vdash |0\rangle : \mathbb{B}$, entonces tendríamos la función $\vdash \lambda z.\ |0\rangle : \sharp\mathbb{B} \to \mathbb{B}$, aplicable a un qubit (con la regla App) y capaz de descartarlo, lo que rompe la unitariedad. Dado que no se permite reducir bajo lambda, esto no supone un problema.

A continuación vemos el Lema de Sustitución para el cual, y a fin de evitar complejizar innecesariamente la prueba, usaremos las notaciones

$\Delta' = \{x_i' : E_i \mid x_i : E_i \in \Delta\}$,
$t' = t[x_1 := x_1'] \ldots [x_n := x_n']$.

Lema 5.3.2 (Sustitución)**.** Si $\Gamma, x : C \vdash \vec{t} : A$ y $\vdash v : C$ entonces $\Gamma \vdash \vec{t}[x := v] : A$

Demostración. Inducción estructural en $\vec{t}$.

- $\vec{t} = x$

  Entonces por el Lema 5.3.1, $C \leq A$ y además $\Gamma^\flat$. De este modo, y dado que $x[x := v] = v$:

$$\dfrac{\dfrac{\vdash x[x := v] : C \quad C \leq A}{\vdash x[x := v] : A}\,\leq \qquad \Gamma^\flat}{\Gamma \vdash x[x := v] : A}\,\mathsf{Weak}$$

- $\vec{t} = y \neq x$

  Entonces por el Lema 5.3.1, $\Gamma = y : D, \Delta^\flat$ para $D \leq A$. Así, como $y = y[x := v]$,

$$\dfrac{\dfrac{\dfrac{}{y : D \vdash y[x := v] : D}\,\mathsf{A_x} \qquad \Delta^\flat}{\Gamma \vdash y[x := v] : D}\,\mathsf{Weak} \qquad D \leq A}{\Gamma \vdash y[x := v] : A}\,\leq$$

- $\vec{t} = |0\rangle$

  Entonces por el Lema 5.3.1, $\Gamma^\flat$ y $\mathbb{B} \leq A$. Así, como $|0\rangle = |0\rangle\,[x := v]$,

$$\dfrac{\dfrac{\dfrac{}{\vdash |0\rangle\,[x := v] : \mathbb{B}}\,\mathsf{A}_{|0\rangle} \qquad \Gamma^\flat}{\Gamma \vdash |0\rangle\,[x := v] : \mathbb{B}}\,\mathsf{Weak} \qquad \mathbb{B} \leq A}{\Gamma \vdash |0\rangle\,[x := v] : A}\,\leq$$

- $\vec{t} = |1\rangle$ Es análogo a $|0\rangle$.

- $\vec{t} = \lambda y\,.\,\vec{s}$

  Entonces por el Lema 5.3.1:

  - $\Gamma, x : C, y : D \vdash \vec{s} : E$,
  - $\Gamma^\flat$ y
  - $D \to E \leq A$.

  Luego, por hipótesis de inducción: $\Gamma, y : D \vdash \vec{s}\,[x := v] : E$. Y por ende:

$$\dfrac{\dfrac{\Gamma, y : D \vdash \vec{s}\,[x := v] : E \qquad \Gamma^\flat}{\Gamma \vdash \lambda y\,.\,\vec{s}\,[x := v] : D \to E}\,\mathsf{Lam} \qquad D \to E \leq A}{\Gamma \vdash \lambda y\,.\,\vec{s}\,[x := v] : A}\,\leq$$

  Y como por convención de Barendregt y no aparece libre en v, $\Gamma \vdash (\lambda y\,.\,\vec{s})[x := v] : A$

- $\vec{t} =? \vec{s} \cdot \vec{u}$

  Entonces por el Lema 5.3.1,

  - $\Gamma, x : C \vdash (\vec{s} \perp \vec{u}) : D$ y
  - $[\sharp]\mathbb{B} \to [\sharp]D \leq A$.

  Y $\Gamma, x : C \vdash (\vec{s} \perp \vec{u}) : D$ significa que:

  - $\Gamma, x : C \vdash \vec{s} : D$
  - $\Gamma, x : C \vdash \vec{u} : D$
  - $\theta(\vec{s}) \perp \theta(\vec{u}), \quad \forall \theta \Vdash \Gamma, x : C$

Por hipótesis de inducción, $\Gamma \vdash \vec{s}\,[x := v] : C$ y $\Gamma \vdash \vec{u}\,[x := v] : C$.

Sea $\theta' \Vdash \Gamma$. Como $\vdash \vec{u}\,[x := v] : C$, entonces $\theta' \cup [x := v] \Vdash \Gamma, x : C$ y por ende $\Gamma, x : C \vdash (\theta' \cup [x := v])(\vec{s}) \perp (\theta' \cup [x := v])(\vec{u}) : D$. Y como $(\theta' \cup [x := v])(\vec{s}) = \theta'(\vec{s}[x := v])$ y $(\theta' \cup [x := v])(\vec{u}) = \theta'(\vec{u}[x := v])$, entonces,

$$\dfrac{\dfrac{\Gamma \vdash (\vec{s}[x := v] \perp \vec{u}[x := v]) : C}{\Gamma \vdash (?\,\vec{s} \cdot \vec{u})[x := v] : [\sharp]\mathbb{B} \to [\sharp]C}\ [\text{Unitary}]\text{If} \qquad [\sharp]\mathbb{B} \to [\sharp]C \leq A}{\Gamma \vdash (?\,\vec{s} \cdot \vec{u})[x := v] : A}\ \leq$$

.

- $\vec{t} = (\vec{s}, \vec{u})$

  Entonces por el Lema 5.3.1, $\Gamma, x : C = \Delta_1, \Delta_2, \Delta_3^\flat$ y existen dos casos:

  - Si $\vec{t}$ fue tipado con la regla Pair.
    - $\Delta_1, \Delta_3 \vdash \vec{s} : D$,
    - $\Delta_2, \Delta_3 \vdash \vec{u} : E$,
    - $D \times E \leq A$.

    Supongamos que $x : C \in \Delta_1$ (o sea que $x : C \notin \Delta_2 \cup \Delta_3$).
    Sea $\Delta_1 = \Xi, x : C$. Entonces $\Xi, x : C, \Delta_3 \vdash \vec{s} : D$ y por hipótesis de inducción $\Xi, \Delta_3 \vdash \vec{s}\,[x := v] : D$ (notar que $(\vec{s}\,[x := v], \vec{u}) = (\vec{s}, \vec{u})[x := v]$)). Entonces

    $$\dfrac{\dfrac{\dfrac{\Xi, \Delta_3 \vdash \vec{s}\,[x := v] : D \quad \Delta_2, \Delta_3' \vdash \vec{u}' : E}{\Xi, \Delta_2, \Delta_3, \Delta_3' \vdash (\vec{s}\,[x := v], \vec{u}') : D \times E}\ \text{Pair}}{\Xi, \Delta_2, \Delta_3 \vdash (\vec{s}\,[x := v], \vec{u}) : D \times E}\ \text{Contr} \qquad D \times E \leq A}{\Gamma \vdash (\vec{s}, \vec{u})[x := v] : A}\ \leq$$

    Si $x : C \in \Delta_2$ el caso es análogo a $x : C \in \Delta_1$.
    Supongamos que $x : C \in \Delta_3$. Sea $\Delta_3 = \Xi, x : C$. Entonces

    - $\Delta_1, \Xi, x : C \vdash \vec{s} : D$
    - $\Delta_2, \Xi, x : C \vdash \vec{u} : E$

    Por hipótesis de inducción, tenemos

    - $\Delta_1, \Xi \vdash \vec{s}\,[x := v] : D$
    - $\Delta_2, \Xi \vdash \vec{u}\,[x := v] : E$

    Como Ξ' y $\vec{u}'$ son un mero renombre de las variables tenemos $\Delta_2, \Xi' \vdash \vec{u}' : E$ y entonces:

    $$\dfrac{\dfrac{\dfrac{\Delta_1, \Xi \vdash \vec{s}\,[x := v] : D \quad \Delta_2, \Xi' \vdash \vec{u}'\,[x := v] : E}{\Delta_1, \Delta_2, \Xi, \Xi' \vdash (\vec{s}, \vec{u}')[x := v] : D \times E}\ \text{Pair}}{\Gamma \vdash (\vec{s}, \vec{u})[x := v] : D \times E}\ \text{Contr} \qquad D \times E \leq A}{\Gamma \vdash (\vec{s}, \vec{u})[x := v] : A}\ \leq$$

- Si $\vec{t}$ fue tipado conla regla Sup.
  - $(\vec{s},\vec{u}) = \sum_{i=1}^{n}\sum_{j=1}^{k}\alpha_i\beta_j\cdot(s_i,u_j)$
  - $\sum_{i=1}^{n}\sum_{j=1}^{k}|\alpha_i\beta_j|^2 = 1$
  - $(\forall_{i\neq k\vee\neq h})\,\Delta_1,\Delta_2,\Delta_3 \vdash (s_i,u_j)\perp(s_k,u_h) : \mathcal{Q}$
  - $\sharp\mathcal{Q} \leq A$
  - $nk \geq 1$

  Por hipótesis de inducción, $(\forall_{i,j})\ \ \Gamma\vdash(s_i,u_j)[x:=v] : \mathcal{Q}$.

  Además, como $\vdash v : C$, entonces para cualquier $\theta' \Vdash \Gamma$, $\theta'\cup[x:=v] \Vdash \Gamma, x:C$ y por ende $\Delta_1,\Delta_2,\Delta_3 \vdash (\theta'\cup[x:=v])((s_i,u_j)) \perp (\theta'\cup[x:=v])((s_k,u_h)) : \mathcal{Q}$. Y como $(\theta'\cup[x:=v])((s_i,u_j)) = \theta'((s_i,u_j)[x:=v])$

$$\dfrac{\dfrac{(i\neq k\vee j\neq h)\ \ \Gamma\vdash((s_i,u_j)\perp(s_k,u_h)) : \mathcal{Q} \quad \sum_{i,j}|\alpha_i\beta_j| = 1}{\Gamma\vdash\sum_{i,j}\alpha_i\beta_j\cdot(s_i,u_j) : \sharp\mathcal{Q}}\ \textsf{Sup} \qquad \sharp\mathcal{Q}\leq A}{\Gamma\vdash\sum_{i,j}\alpha_i\beta_j\cdot(s_i,u_j) : A}\ \leq$$

- $\vec{t} = s\vec{u}$

  Por el Lema 5.3.1 Existen tres casos.

  - Si $\vec{t}$ fue tipado con App [o UnitaryApp]
    - $\Gamma = \Delta_1,\Delta_2,\Delta_3^\flat$,
    - $\Delta_2,\Delta_3 \vdash u : [\sharp]D$,
    - $\Delta_1,\Delta_3 \vdash s : [\sharp]D \to [\sharp]E$,
    - $[\sharp]E \leq A$;

    Si $x:C\in\Delta_1$. Sea $\Delta_1 = \Xi, x:C$, de modo que $\Xi, x:C,\Delta_3 \vdash s : [\sharp]D\to[\sharp]E$.

    Por hipótesis de inducción, $\Xi,\Delta_3\vdash s[x:=v] : [\sharp]D\to[\sharp]E$ y $\Delta_2,\Delta_3\vdash u[x:=v] : [\sharp]D$.

    Entonces:

$$\dfrac{\dfrac{\dfrac{\Xi,\Delta_3\vdash s[x:=v] : [\sharp]D\to[\sharp]E \quad \Delta_2,\Delta_3'\vdash u'[x:=v] : [\sharp]D}{\Xi,\Delta_2,\Delta_3,\Delta_3'\vdash su'[x:=v] : [\sharp]E}\ [\textsf{Unitary}]\textsf{App}}{\Xi,\Delta_2,\Delta_3\vdash su[x:=v] : [\sharp]E}\ \textsf{Contr} \qquad [\sharp]E\leq A}{\Gamma\vdash su[x:=v] : A}\ \leq$$

    Si $x:C\in\Delta_2$ es análogo.

    Si $x:C\in\Delta_3$ Sea $\Delta_3 = \Xi, x:C$. Entonces
    - $\Delta_1,\Xi,x:C\vdash s : [\sharp]D\to[\sharp]E$
    - $\Delta_2,\Xi,x:C\vdash u : [\sharp]D$

    Por hipótesis de inducción, tenemos

- $\Delta_1, \Xi \vdash s\ [x := v] : [\sharp]D \to [\sharp]E$
- $\Delta_2, \Xi \vdash u\ [x := v] : [\sharp]D$

y

$$\dfrac{\dfrac{\dfrac{\Delta_1, \Xi \vdash s\ [x := v] : [\sharp]D \to [\sharp]E \quad \Delta_2, \Xi' \vdash u'[x := v] : [\sharp]D}{\Delta_1, \Delta_2 \Xi, \Xi' \vdash su'[x := v] : [\sharp]E}\ \textsf{[Unitary]App}}{\Gamma \vdash su[x := v] : [\sharp]E}\ \textsf{Contr} \qquad [\sharp]E \leq A}{\Gamma \vdash su[x := v] : A}\ \leq$$

- Si fue tipado con Sup
  - $\vec{t} = \sum_{i=1}^{n} \alpha_i \cdot su_i$,
  - $\sum_{k=1}^{n} |\alpha_i|^2 = 1$,
  - $(\forall_{i \neq j})\ \ \Gamma \vdash (su_i \perp su_j) : \mathcal{Q}$,
  - $\sharp \mathcal{Q} \leq A$,
  - $n \geq 1$,
  - $\vec{u} = \sum_{i=1}^{n} \alpha_i \cdot u_i$

  Por hipótesis de inducción, $(\forall_i)\ \ \Gamma \vdash su_i[x := v] : \mathcal{Q}$.

  Además, como $\vdash [x := v] : C$, entonces para cualquier $\theta' \Vdash \Gamma$, $\theta' \cup [x := v] \Vdash \Gamma, x : C$ y por ende $\Delta_1, \Delta_2, \Delta_3 \vdash (\theta' \cup [x := v])(su_i) \perp (\theta' \cup [x := v])((su_k)) : \mathcal{Q}$. Y como $(\theta' \cup [x := v])(su_i) = \theta'(su_i[x := v])$

$$\dfrac{\dfrac{(i \neq k \vee j \neq h)\ \ \Gamma \vdash (su_i \perp su_k) : \mathcal{Q} \quad \sum_i |\alpha_i|^2 = 1}{\Gamma \vdash \sum_i \alpha_i \cdot su_i : \sharp \mathcal{Q}}\ \textsf{Sup} \qquad \sharp \mathcal{Q} \leq A}{\Gamma \vdash \sum_i \alpha_i \cdot su_i : A}\ \leq$$

- $\vec{t} = \texttt{let}\ (y_1, y_2) = \vec{u}\ \texttt{in}\ \vec{s}$

  Por Lema 5.3.1 existen tres casos:

  - Si $\vec{t}$ fue tipado con la regla Let [*o* UnitaryLet]:
    - $\Gamma, x : C = \Delta_1, \Delta_2, \Delta_3^\flat$,
    - $\Delta_1, \Delta_3 \vdash u : [\sharp]D \times E$,
    - $\Delta_2, \Delta_3, y_1 : [\sharp]D, y_2 : [\sharp]E \vdash \vec{s} : [\sharp]F$,
    - $[\sharp]F \leq A$;

    Si $x : C \in \Delta_1$, $\Delta_1 = \Xi, x : C$ y $\Xi, x : C, \Delta_3 \vdash u : [\sharp]D \times E$. Por hipótesis de inducción: $\Xi, \Delta_3 \vdash u[x := v] : [\sharp]D \times E$.

    Entonces:

$$\dfrac{\dfrac{\dfrac{\Xi,\Delta_3 \vdash u[x:=v] : [\sharp]D \times E \quad \Delta_2,\Delta_3',y_1 : [\sharp]D, y_2 : [\sharp]E \vdash \vec{s}'\,[x:=v] : [\sharp]F}{\Xi,\Delta_2,\Delta_3,\Delta_3' \vdash (\texttt{let}\ (y_1,y_2) = u\ \texttt{in}\ \vec{s}')[x:=v] : [\sharp]E}\ \text{[Unitary]Let}}{\Xi,\Delta_2,\Delta_3,\vdash (\texttt{let}\ (y_1,y_2) = u\ \texttt{in}\ \vec{s})[x:=v] : [\sharp]E}\ \text{Contr} \qquad [\sharp]E \leq A}{\Gamma \vdash (\texttt{let}\ (y_1,y_2) = u\ \texttt{in}\ \vec{s})[x:=v] : A}\ \leq$$

Si $x : C \in \Delta_2$ es análogo.

Si $x : C \in \Delta_3$ Sea $\Delta_3 = \Xi, x : C$. Por el Lema 5.3.1:

- $\Delta_1, \Xi \vdash u : [\sharp]D \times E$,
- $\Delta_2, \Xi, y_1 : [\sharp]D, y_2 : [\sharp]E \vdash \vec{s} : [\sharp]F$,

Por hipótesis de inducción, tenemos

- $\Delta_1, \Xi \vdash u[x = v] : [\sharp]D \times E$,
- $\Delta_2, \Xi, y_1 : [\sharp]D, y_2 : [\sharp]E \vdash \vec{s}\,[x = v] : [\sharp]F$,

Entonces

$$\dfrac{\dfrac{\dfrac{\Delta_1,\Xi \vdash u\,[x=v][\sharp]D \times E : \quad \Delta_2,\Xi',y_1 : [\sharp]D, y_2 : [\sharp]E \vdash \vec{s}'[x=v] : [\sharp]F}{\Delta_1,\Delta_2,\Xi,\Xi' \vdash (\texttt{let}\ (y_1,y_2) = u\ \texttt{in}\ \vec{s}')[x:=v] : A}\ \text{[Unitary]Let}}{\Gamma \vdash (\texttt{let}\ (y_1,y_2) = u\ \texttt{in}\ \vec{s})[x:=v] : [\sharp]E}\ \text{Contr} \qquad [\sharp]E \leq A}{\Gamma \vdash (\texttt{let}\ (y_1,y_2) = u\ \texttt{in}\ \vec{s})[x:=v] : A}\ \leq$$

- Si fue tipado con Sup
  - $\sum_{k=1}^{n} |\gamma_k|^2 = 1$,
  - $(\forall_{k \neq l})\ \ \Gamma, x : C \vdash (\texttt{let}\ (x,y) = u_k\ \texttt{in}\ \vec{s} \perp \texttt{let}\ (x,y) = u_k\ \texttt{in}\ \vec{s}) : \mathcal{Q}$,
  - $\sharp\mathcal{Q} \leq A$,
  - $n \geq 1$,
  - $\vec{u} = \sum_{k=1}^{n} \gamma_k \cdot u_k$

  Por hipótesis inductiva $(\forall_k)\ \ \Gamma \vdash \texttt{let}\ (x,y) = u_k\ \texttt{in}\ \vec{s}[x:=v] : \mathcal{Q}$,

  Además, sea $\theta' \Vdash \Gamma$ y nuevamente como $\vdash [x:=v] : C$ tenemos que para todo $i \neq j$ se cumple: $\Gamma \vdash (\theta'(\texttt{let}\ (x,y) = u_i\ \texttt{in}\ \vec{s}[x:=v]) \perp \theta'(\texttt{let}\ (x,y) = u_j\ \texttt{in}\ \vec{s}[x:=v])) : \mathcal{Q}$.

  Luego:

$$\dfrac{^{(i \neq j)}\ \ \Gamma \vdash (\texttt{let}\ (y_1,y_2) = u_i\ \texttt{in}\ \vec{s}[x:=v] \perp \texttt{let}\ (y_1,y_2) = u_j\ \texttt{in}\ \vec{s}[x:=v]) : \mathcal{Q} \quad \sum_{k=1}^{n} |\gamma_k|^2 = 1 \quad m \geq 1}{\Gamma \vdash \texttt{let}\ (y_1,y_2) = \vec{u}\ \texttt{in}\ \vec{s} : \sharp\mathcal{Q}}\ \text{Sup}$$

- $\vec{t} = \pi_l\ \vec{s}$

  Por el Lema 5.3.1 :

  - $\Gamma, x : C \vdash \vec{s} : \sharp\mathbb{B}^n$

- $\mathbb{B}^l \times \sharp\mathbb{B}^{n-l} \leq A$

Por hipótesis inductiva: $\Gamma \vdash \vec{s}[x := v] : \sharp\mathbb{B}^n$ y entonces:

$$\dfrac{\dfrac{\Gamma \vdash \vec{s}\,[x := v] : \sharp\mathbb{B}^n}{\Gamma \vdash (\pi_l \vec{s})[x := v] : \mathbb{B}^n \times \mathbb{B}^{n-l}}\ \pi \qquad \mathbb{B}^n \times \mathbb{B}^{n-l} \leq A}{\Gamma \vdash (\pi_l \vec{s})[x := v] : A}\ \leq$$

- $\vec{t} = \sum_{i=1}^{n} \alpha_i \cdot s_i$

Por Lema 5.3.1 :

- $\sum_{i=1}^{n} |\alpha_i|^2 = 1$
- $_{(i \neq j)}\ \Gamma \vdash (s_i \perp s_j) : D$
- $\sharp D \leq A$
- $n \geq 1$

Por hipótesis de inducción, se cumple $(\forall_i)\ \Gamma \vdash s_i[x := v] : D$. Además, sea $\theta' \Vdash \Gamma$ y nuevamente como $\vdash v : C$ tenemos que para todo $i \neq j$ se cumple: $\theta'(s_i[x := v]) \perp \theta'(s_j[x := v]) : D$

Luego:

$$\dfrac{_{(i \neq j)}\ \Gamma \vdash (s_i[x := v] \perp s_j[x := v]) : D \quad \sum |\alpha_i|^2 = 1 \quad m \geq 1}{\sum_{i=1}^{n} \alpha_i \cdot s_i[x := v] : \sharp D}\ \textsf{Sup}$$

□

Teorema 5.3.3 (Subject reduction)**.** Si $\vdash \vec{t} : A$ y $\vec{t} \longrightarrow \vec{r}$ entonces $\vdash \vec{r} : A$

Demostración. Prueba por inducción en la relación de reescritura.

- Caso $(\lambda x.\vec{t}\,)v \longrightarrow \vec{t}[x := v]$.

Por Lema 5.3.1 tenemos

$$\dfrac{\dfrac{\dfrac{x : D \vdash \vec{t} : E}{\vdash \lambda x.\vec{t}\ : D \to E}\ \textsf{Lam} \qquad \vdash v : D}{\vdash (\lambda x.\vec{t}\,)v : E}\ \textsf{App} \qquad E \leq A}{\vdash (\lambda x.\vec{t}\,)v : A}\ \leq$$

y este es el único caso ya que $v \in \mathsf{V}$. De este modo, por el Lema 5.3.2, usando $x : D \vdash \vec{t} : E$ y $\vdash v : D$ obtenemos $\vdash \vec{t}[x := v] : A$.

- Caso $\texttt{let}\ (x,y) = (v,w)\ \texttt{in}\ \vec{s} \longrightarrow \vec{s}[x := v, y := w]$

  Por Lema 5.3.1, obtenemos

$$\dfrac{\dfrac{\dfrac{\vdash v : D \quad \vdash w : E}{\vdash (v,w) : D \times E}\ \textsf{Pair} \qquad x : D, y : E \vdash \vec{s} : F}{\vdash \texttt{let}\ (x,y) = (v,w)\ \texttt{in}\ \vec{s} : F}\ \textsf{Let} \qquad F \leq A}{\vdash \texttt{let}\ (x,y) = (v,w)\ \texttt{in}\ \vec{s} : A}\ \leq$$

  De este modo, usando el Lema 5.3.2 dos veces, dado que $x : D, y : E \vdash \vec{s} : F$, $\vdash v : D$ y $\vdash w : E$, obtenemos $\vdash \vec{s}[x := v, y := w] : A$.

- Caso $(?\,\vec{t} \cdot \vec{r})\,|0\rangle \longrightarrow \vec{t}$

  Usando Lema 5.3.1 tenemos

$$\dfrac{\dfrac{\dfrac{\vdash (\vec{t} \perp \vec{r}) : C}{\vdash ?\,\vec{t} \cdot \vec{r} : \mathbb{B} \to C}\ \textsf{If} \qquad \vdash |0\rangle : \mathbb{B}}{\vdash (?\,\vec{t} \cdot \vec{r})\,|0\rangle : C}\ \textsf{App} \qquad C \leq A}{\vdash (?\,\vec{t} \cdot \vec{r})\,|0\rangle : A}\ \leq$$

  De este modo $\vdash \vec{t} : A$

- Caso $(?t \cdot r)\,|1\rangle$ (análogo al anterior).

- Caso $\pi_l \left(\sum_{i=1}^{m} \alpha_i \prod_{h=1}^{n} |b_{hi}\rangle \right) \longrightarrow_{p_k} |k\rangle_l \times |\phi_k\rangle$.

  Por Lema 5.3.1

$$\dfrac{\dfrac{(i \neq j) \quad \vdash \prod_{h=1}^{n} |b_{hi}\rangle \perp \prod_{h=1}^{n} \left|b_{hj}\right\rangle : \mathbb{B}^n \qquad \sum_{i=1}^{m} |\alpha_i|^2 = 1}{\vdash \sum_{i=1}^{m} \alpha_i \prod_{h=1}^{n} |b_{hi}\rangle : \sharp\mathbb{B}^n}\ \textsf{Sup}}{\vdash \pi_l(\sum_{i=1}^{m} \alpha_i \prod_{h=1}^{n} |b_{hi}\rangle) : \mathbb{B}^n \times \sharp\mathbb{B}^{n-l}}\ \pi$$

  donde $\mathbb{B}^n \times \sharp\mathbb{B}^{n-l} \leq A$.

  Claramente $|k\rangle_l = |b_1 \ldots b_l\rangle : \mathbb{B}^l$. Por otra parte, $\sum_{i \in T_k} \left| \dfrac{\alpha_i}{\sqrt{\sum_{r \in T_k} |\alpha_r|^2}} \right|^2 = \sum_{i \in T_k} \dfrac{|\alpha_i|^2}{\sum_{r \in T_k} |\alpha_r|^2} = \dfrac{\sum_{i \in T_k} |\alpha_i|^2}{\sum_{r \in T_k} |\alpha_r|^2} = 1$. Además, como $\prod_{h=1}^{n} |b_{hi}\rangle \perp \prod_{h=1}^{n} \left|b_{hj}\right\rangle$ y dado que $T_k = \{i \leq m \,|\, |b_{1i} \cdots 1_l\rangle = |k\rangle_l\}$ tiene que ser $\prod_{h=l+1}^{n} |b_{hi}\rangle \perp \prod_{h=l+1}^{n} \left|b_{hj}\right\rangle$ dado que si existieran $i, j \in T_k$ tales que $\prod_{h=l+1}^{n} |b_{hi}\rangle =$

$\prod_{h=l+1}^{n} |b_{hj}\rangle$ entonces tendríamos $|k\rangle_l \times \prod_{h=l+1}^{n} |b_{hi}\rangle = |k\rangle_l \times \prod_{h=l+1}^{n} |b_{hj}\rangle$ lo que es absurdo porque son ortogonales.

Entonces por Sup

$$\vdash \sum_{i \in T_k} \frac{\alpha_i}{\sqrt{\sum_{r \in T_k} |\alpha_r|^2}} \prod_{h=l+1}^{n} |b_{hi}\rangle : \sharp\mathbb{B}^{n-l}$$

De modo que $|k\rangle_l \times |\phi_k\rangle : \mathbb{B}^l \times \sharp\mathbb{B}^{n-l} \leq A$.

Ahora consideramos los casos inductivos.

- Caso $\dfrac{t \longrightarrow \vec{r}}{(t,\vec{s}) \longrightarrow (\vec{r},\vec{s})}$

  Por 5.3.1 tenemos dos casos. Si se tipó con Pair, $\vdash t : B$, $\vdash \vec{s} : C$, $B \times C \leq A$ y por hipótesis de inducción $\vdash \vec{r} : B$. Por lo tanto usando Pair obtenemos $\vdash (\vec{r},\vec{s}) : A$.

  Si en cambio se usó Sup tenemos

  - $(t,\vec{s}) = \sum_{i=1}^{n_t} \sum_{j=1}^{n_s} \alpha_i \beta_j \cdot (t_i, \vec{s}_j)$
  - $\sum_{i=1}^{n} \sum_{j=1}^{k} |\alpha_i \beta_j|^2 = 1$.
  - $(\forall_{i_1 \neq i_2 \vee j_1 \leq j_2}) \ \vdash ((t_{i_1}, \vec{s}_{j_1}) \perp (t_{i_2}, \vec{s}_{j_2})) : \mathcal{Q}$,
  - $t = \sum_{i=1}^{n_t} \alpha_i \cdot t_i$,
  - $\vec{s} = \sum_{i=1}^{n_s} \beta_i \cdot s_i$
  - $\sharp\mathcal{Q} \leq A$,
  - $n_r n_s \geq 1$,

  Además $\mathcal{Q} = \sharp\mathcal{Q}_{m_t} \times \sharp\mathcal{Q}_{m_s}$, donde $\vdash t : \sharp\mathcal{Q}_{m_t}$ y $\vdash \vec{s} : \sharp\mathcal{Q}_{m_s}$ (sabemos que son tipos con $\sharp$ porque $n_t n_s \geq 1$). Entonces, dado que $t \longrightarrow \vec{r}$, tenemos por hipótesis de inducción que $\vdash \vec{r} : \sharp\mathcal{Q}_{m_t}$ lo que implica que $\vec{r} = \sum_{i=1}^{n_r} \gamma_i \cdot \vec{r}_i$ con $\sum_{i=1}^{n_r} |\gamma_i|^2 = 1$ y $\vdash (\vec{r}_i \perp \vec{r}_j) : \mathcal{Q}_{m_t}$ para todo $i \neq j$. De este modo, $\sum_{i=1}^{n_r} \sum_{j=1}^{n_s} \gamma_i \beta_j \cdot (\vec{r}_i, \vec{s}_j) = (\vec{r},\vec{s})$.

  Además $1 = (\sum_{i=1}^{n_r} |\gamma_i|^2)(\sum_{j=1}^{n_s} |\beta_j|^2) = \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} |\gamma_j \beta_j|^2$ y también $\vdash ((\vec{r}_{i_1}, \vec{s}_{j_1}) \perp (\vec{r}_{i_2}, \vec{s}_{j_2})) : \sharp\mathcal{Q}_{m_t} \times \sharp\mathcal{Q}_{m_s} = \mathcal{Q}$ para todo $i_1 \neq i_2$ o $j_1 \neq j_2$.

  Por lo tanto, podemos usar la regla Sup y por ende $\vdash (\vec{r},\vec{s}) : A$

- Caso $(\vec{s},t) \longrightarrow (\vec{s},\vec{r})$

  Es análogo al anterior.

- Caso $\dfrac{t \longrightarrow \vec{r}}{st \longrightarrow s\vec{r}}$.

  Por 5.3.1 tenemos tres casos. Si se tipó con App [o UnitaryApp], tenemos $\vdash s : [\sharp]D \to [\sharp]E$, $\vdash t : [\sharp]D$ y $[\sharp]E \leq A$. Además, por hipótesis de inducción $\vdash \vec{r} : [\sharp]D$ Por lo tanto, usando las reglas [Unitary]App y $\leq$ obtenemos $\vdash s\vec{r} : A$

Si se tipó con Sup:

- $st = \sum_{i=1}^{n} \alpha_i \cdot st_i$,
- $\sum_{k=1}^{n} |\alpha_i|^2 = 1$,
- $(\forall_{i \neq j}) \ \vdash (st_i \perp st_j) : \mathcal{Q}$,
- $\sharp\mathcal{Q} \leq A$,
- $n \geq 1$,
- $t = \sum_{i=1}^{n} \alpha_i \cdot t_i$

y además $\vdash s : D \to E$ y $\vdash t_i : D$.

Como $t \longrightarrow \vec{r}$, $st_i \longrightarrow s\vec{r}_i$ y $\vec{r} = \sum_{i=1}^{n} \vec{\alpha}_i \cdot r_i$, $\vdash (s\vec{r}_i \perp s\vec{r}_j) : \mathcal{Q}$. Y $s\vec{r} = \sum_{i=1}^{n} \alpha_i \cdot s\vec{r}_i$. Luego, usando la regla Sup obtenemos $\vdash s\vec{r} : \sharp\mathcal{Q}$.

Si se da que $t \equiv \vec{r}$ en virtud de una de las reglas de la Tabla 5.2, entonces es trivial chequear que en cada caso si $t \equiv \vec{r}$ también $st \equiv s\vec{r}$.

- Caso $\dfrac{t \longrightarrow r}{t\vec{v} \longrightarrow r\vec{v}}$

  Por 5.3.1 tenemos tres casos. Si se tipó con [Unitary]App, tenemos $\vdash t : [\sharp]D \to [\sharp]E$, $\vdash \vec{v} : [\sharp]D$ y $[\sharp]E \leq A$. Además, por hipótesis de inducción $\vdash r : [\sharp]D \longrightarrow [\sharp]E$. Por lo tanto, usando las reglas [Unitary]App y $\leq$ obtenemos $\vdash r\vec{v} : A$.

  Si se tipó con Sup, entonces $t\vec{v} = \sum_{i=1}^{n} \alpha_i \cdot tv_i$ y $r\vec{v} = \sum_{i=1}^{n} \alpha_i \cdot rv_i$. Por hipótesis de inducción r preserva el tipo de t y por definición de $\perp$ tenemos $\vdash (rv_i \perp rv_j) : E$ para todo $i \neq j$. Por lo que $\vdash r\vec{v} : A$.

- Caso $\dfrac{t \longrightarrow \vec{r}}{\pi_l t \longrightarrow \pi_l \vec{r}}$

  Por 5.3.1 tenemos $\vdash t : \sharp\mathbb{B}^n$ y $\mathbb{B}^l \times \sharp\mathbb{B}^{n-l} \leq A$. Además, por hipótesis de inducción $\vdash \vec{r} : \sharp\mathbb{B}^n$. Por lo tanto, usando la regla π obtenemos $\vdash \pi_l \vec{r} : A$.

- Caso $\dfrac{t \longrightarrow \vec{r}}{\texttt{let}\ (x,y) = t\ \texttt{in}\ \vec{s} \longrightarrow \texttt{let}\ (x,y) = \vec{r}\ \texttt{in}\ \vec{s}}$.

  Por 5.3.1 existen tres casos. Si fue tipado con Let (o UnitaryLet), tenemos: $\vdash t : [\sharp]D \times E$, $x : [\sharp]D, y : [\sharp]E \vdash \vec{s} : [\sharp]F$ y $D \leq A$. Además, por hipótesis de inducción $\vdash r : [\sharp]D \times E$. Entonces por Let y $\leq$ tenemos: $\vdash \texttt{let}\ (x,y) = r\ \texttt{in}\ \vec{s} : A$

  Si fue tipado con Sup

  - $\texttt{let}\ (x,y) = t\ \texttt{in}\ \vec{s} = \sum_{k=1}^{n} \gamma_k \cdot (\texttt{let}\ (x,y) = t_k\ \texttt{in}\ \vec{s})$,
  - $\sum_{k=1}^{n} |\gamma_k|^2 = 1$,
  - $(\forall_{k \neq l}) \ \vdash (\texttt{let}\ (x,y) = t_k\ \texttt{in}\ s \perp \texttt{let}\ (x,y) = t_l\ \texttt{in}\ \vec{s}) : \mathcal{Q}$,
  - $\sharp\mathcal{Q} \leq A$,
  - $n \geq 1$,

- $t = \sum_{k=1}^{n} \gamma_k \cdot t_k$

  Y por los tanto $\vec{r} = \sum_{k=1}^{n} \gamma_k \cdot \vec{r}_k$ donde los $\vec{r}_k$ mantienen el tipo pot hipótesis de inducción y la ortogonalidad por defínción de $\perp$, por lo que $\vdash (\texttt{let}\ (x,y) = \vec{r}_k\ \texttt{in}\ s \perp \texttt{let}\ (x,y) = \vec{r}_l\ \texttt{in}\ \vec{s}) : \mathcal{Q}$, y por ende podemos usar Sup y tipar $\vdash \texttt{let}\ (x,y) = r\ \texttt{in}\ \vec{s} : A$

- Caso $\dfrac{t \longrightarrow \vec{r}}{\alpha \cdot t + \vec{s} \longrightarrow \alpha \cdot \vec{r} + \vec{s}}$

  Por el Lema 5.3.1 tenemos $\vec{s} = \sum_{i=1}^{m} \beta_i \cdot s_i$ donde $(\forall_{i \neq j}) \vdash (s_i \perp s_j) : D$, $(\forall_i) \vdash s_i \perp t : D$, $\sharp D \leq A$ y $\sum_{i=1}^{m} |\beta_i|^2 + |\alpha|^2 = 1$. Además, por hipótesis de inducción $\vdash \vec{r} : D$ y por definición de $\perp$, dado que $t \longrightarrow \vec{r}$, $(\forall_i) \vdash s_i \perp \vec{r} : D$.

  Por lo tanto, usando la regla Sup, $\vdash \alpha \cdot \vec{r} + \vec{s} : A$

□

Teorema 5.3.4 (Progreso)**.** Sea $\vdash \vec{t} : A$ luego si $\vec{t} \not\longrightarrow$ entonces $\vec{t} \in \vec{\mathsf{V}}$

Demostración. Por inducción en $\vec{t}$.

- Si $\vec{t} \in \vec{\mathsf{V}}$ no hay nada que probar.
- $\vec{t} = sr$. Como $\vec{t} \not\longrightarrow$, tampoco s ni r, entonces por hipótesis de inducción $s, r \in \vec{\mathsf{V}}$. Por Lema 5.3.1 $\vdash s : C \to A$ y $\vdash r : C$,

  Por lo que $s = \lambda x.\vec{u}$, $s = \lambda^\sharp x.\vec{u}$ o $s = ?\ q \cdot p$, pero eso es absurdo porque tanto $(\lambda x.\vec{u})r$ como $(\lambda^\sharp x.\vec{u})r$ y $(?\ q \cdot p)r$ reducen.
- $\vec{t} = (\vec{s}, \vec{r})$. Entonces $\vec{s} \not\longrightarrow$ y $\vec{r} \not\longrightarrow$ ya que de lo contrario, por las reglas de la Tabla 5.4, $\vec{t}$ reduciría. Luego, por hipótesis de inducción $\vec{s} \in \vec{\mathsf{V}}$ y $\vec{r} \in \vec{\mathsf{V}}$. Pero esto es absurdo ya que en tal caso $\vec{t} \in \vec{\mathsf{V}}$.
- $\vec{t} = \texttt{let}\ (x,y) = s\ \texttt{in}\ \vec{r}$. Como $\vec{t} \not\longrightarrow$, tampoco s, entonces por hipótesis de inducción $s \in \vec{\mathsf{V}}$. Por Lema 5.3.1 $\vdash s : C \times D$, por lo que $s = (q, p)$. Pero eso es absurdo porque $\texttt{let}\ (x,y) = (q,p)\ \texttt{in}\ \vec{r} \longrightarrow \vec{r}[x := q][y := p]$.
- $\vec{t} = \pi_l \vec{s}$. Como $\vec{t} \not\longrightarrow$, tampoco $\vec{s}$, Por Lema 5.3.1 $\vdash \vec{s} : \sharp\mathbb{B}^n$, por lo que $\vec{s} = \sum_i \alpha_i |i\rangle$, ya que por hipótesis de inducción $\vec{s} \in \vec{\mathsf{V}}$.

  Pero eso es absurdo porque $\pi_l \sum_i \alpha_i |i\rangle\, s$ sí reduce.
- $\vec{t} = \sum_i \alpha_i \vec{s}_i$. Entonces para cada i, $\vec{s}_i \not\longrightarrow$ ya que de lo contrario, por las reglas de la Tabla 5.4, $\vec{t}$ reduciría. Luego, por hipótesis de inducción $\vec{s}_i \in \vec{\mathsf{V}}$. Pero esto es absurdo ya que en tal caso $\vec{t} \in \vec{\mathsf{V}}$.

□

Teorema 5.3.5 (Norma)**.** Si $\vdash \vec{v} : A$ entonces $\|\vec{v}\| = 1$

Demostración. Si $\vec{v} = v \in \mathsf{V}$ (es un valor puro) entonces $\|v\| = (v \mid v) = \delta_{v,v} = 1$. Si es una distribución $\|\vec{v}\| = \sum_{j=1}^{m} \alpha_j \cdot v_j$, donde por el Lema 5.3.1 $\forall_{i \neq j} \vdash (\vec{v}_i \perp \vec{v}_j)$ y $\sum_{i=1}^{n} \|\alpha_i\|^2 = 1$.

De este modo, $|\vec{v}| = \sum_{j=1}^{m} \delta_{jj} \alpha_j^2 = \sum_{j=1}^{m} \alpha_j^2 = 1$

□

5.4. Expresividad

En esta sección vamos a ver que el lenguaje incluye, por un lado, el cálculo lambda simplemente tipado y extendido con pares y, por otra parte, también los elementos básicos de la computación cuántica, o sea: qubits, compuertas cuánticas y medición. Con respecto a los qubits, esto es simplemente porque la gramática de términos incluye los qubits y los pares. Como ejemplo de una aplicación también expresaremos el algoritmo de teleportación cuántica en términos de Lambda-$\mathcal{S}_1^\pi$, para lo cual introducimos además la notación case of que permite representar operadores más facilmente.

Definición 5.4.1 (case of)**.**

En primer lugar, definimos recursivamente la notación $\mathsf{curried\_case}_n(\vec{v}_0, \ldots, \vec{v}_{2^n-1})$ del siguiente modo:

$\mathsf{curried\_case}_1(\vec{v}_0, \vec{v}_1) = (?\, \vec{v}_0 \cdot \vec{v}_1)$

Y para $m > 1$:

$\mathsf{curried\_case}_m(\vec{v}_0, \ldots, \vec{v}_{2^n-1}) = (?\, \mathsf{curried\_case}_{\frac{m}{2}}(\vec{v}_0, \ldots, \vec{v}_{2^{\frac{m}{2}}-1}) \cdot \mathsf{curried\_case}_{\frac{m}{2}}(\vec{v}_{2^{\frac{m}{2}}}, \ldots, \vec{v}_{2^m-1}))$

Entonces

$$\begin{aligned}\mathsf{case}\ x\ \mathsf{of}\ \{|0\rangle_n \mapsto \vec{v}_0 \mid \cdots \mid |2^n-1\rangle_n \mapsto \vec{v}_{2^n-1}\} = {} & \mathtt{let}\ (x_0, x') = x\ \mathtt{in} \\ & \mathtt{let}\ (x_1, x'') = x_1\ \mathtt{in} \\ & \ldots \\ & \mathtt{let}\ (x_{2^n-2}, x_{2^n-1}) = x'^{\cdots'}\ \mathtt{in} \\ & \quad \mathsf{curried\_case}_n(\vec{v}_0, \ldots, \vec{v}_{2^n-1})\ x_1\ \ldots\ x_{2^n-1}\end{aligned}$$

Así, dado cualquier n, podemos definir la expresión que toma por parámetro el término $|k\rangle_n$ y devuelve $\vec{v}_k$ mediante $\lambda x.\, \mathsf{case}\ x\ \mathsf{of}\ \{|0\rangle_n \mapsto \vec{v}_0 \mid \cdots \mid |2^n-1\rangle_n \mapsto \vec{v}_{2^n-1}\}$. Como cualquier operación lineal queda definida dados los valores a los que evalúa en cada vector de la base, podemos por ende definir mediante esta notación las operaciones lineales.

Ejemplo 5.4.1. El término

$$\begin{aligned}\lambda x.\, \mathsf{case}\ x\ \mathsf{of}\ \{ & \\ & |0\rangle_3 \mapsto |0\rangle_3 \\ \mid\ & |1\rangle_3 \mapsto |1\rangle_3 \\ \mid\ & |2\rangle_3 \mapsto |2\rangle_3 \\ \mid\ & |3\rangle_3 \mapsto |3\rangle_3 \\ \mid\ & |4\rangle_3 \mapsto |4\rangle_3 \\ \mid\ & |5\rangle_3 \mapsto |5\rangle_3 \\ \mid\ & |6\rangle_3 \mapsto |6\rangle_3 \\ \mid\ & |7\rangle_3 \mapsto |7\rangle_3 \\ \} & \end{aligned}$$

representa la identidad en $\mathbb{C}^8$.

Definimos la siguiente notación para el Teorema de Expresividad.

Definición 5.4.2. Sean $b_i \in \{0,1\}$ y $\underline{k} = b_0 \dots b_m$ donde $b_0 \dots b_m$ es la representación binaria de $k \in \mathbb{N}$ con m bits. Entonces, dado un vector $|\underline{k}\rangle \in \mathbb{C}^{2^n}$, podemos codificarlo mediante el término $|k\rangle_n$, que también notamos $|\hat{\underline{k}}\rangle$.

Asimismo, dado cualquier vector $\vec{v} \in \mathbb{C}^{2^n}$, podemos escribirlo como $\sum_{i=1}^{2^n} \alpha\, |\underline{k}\rangle$, que codificamos $\sum_{i=1}^{2^n} \alpha |\hat{\underline{k}}\rangle$, y también notamos $\hat{\vec{v}}$.

Teorema 5.4.1 (Expresividad)**.**

1. El Cálculo Lambda Simplemente Tipado extendido con pares con call-by-value está incluido en Lambda-$\mathcal{S}_1^{\pi}$.

2. Si U es una isometría en $\mathbb{C}^{2^n}$, entonces existe un término $\hat{U}$ tal que $\vdash \hat{U} : \sharp\mathbb{B}^n \to \sharp\mathbb{B}^n$ y para todo $\vec{v} \in \mathbb{C}^{2^n}$, si $\vec{w} = U\vec{v}$, entonces $\hat{U}\hat{\vec{v}} \to^* \hat{\vec{w}}$.

3. Si $\vec{v} \in \mathbb{C}^{2^n}$ es un vector que representa un sistema de qubits entonces el término $\vdash \pi_l \hat{\vec{v}} : \mathbb{B}^l \times \sharp\mathbb{B}^{n-l}$ representa una medición del sistema.

Demostración.

- Los términos del cálculo lambda con pares están incluidos en la gramática de términos puros de Lambda-$\mathcal{S}_1^{\pi}$. El conjunto de tipos $A^\flat$ incluye los tipos simples. Los tipos simples junto con las reglas Ax, Lam, App, Pair, PureLet, Weak y Contr permite tipar el cálculo lambda extendido con pares.

- Sea U una isometría. Entonces, podemos definirla dando su comportamiento en la base de $\mathbb{C}^{2^n}$. Así, sea la base canónica $B = \{|0\rangle_n, \dots, |2^n - 1\rangle_n\}$ y para todo $|k\rangle_n \in B$ sea $U|k\rangle_n = (\beta_{0,k}, \dots, \beta_{2^n-1,k})$, o sea que $U = (\beta_{i,j})_{i,j}$.

  Definimos el término $\hat{U}$ usando la notación case introducida en el Ejemplo 5.4.1

$$\hat{U} = \lambda x \,.\, \mathsf{case}\ x\ \mathsf{of}\ \{|\hat{k}\rangle_n \mapsto \sum_{i=0}^{2^n-1} \beta_{ik} \cdot |i\rangle_n\}$$

  Sea $\vec{v} = (\alpha_0, \dots, \alpha_{2^n-1})^T = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle_n \in \mathbb{C}^{2^n}$ con $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$ y $\hat{\vec{v}} = \sum_{k=0}^{2^n-1} \alpha_k |\hat{k}\rangle_n$ con $\vdash \hat{\vec{v}} : \mathbb{B}^{\otimes n}$. Entonces tenemos

$$\hat{U}\hat{\vec{v}} \equiv \sum_{k=0}^{2^n-1} \alpha_k \cdot \hat{U}|\hat{k}\rangle_n \to^* \sum_{k=0}^{2^n-1} \alpha_k \sum_{i=0}^{2^n-1} \beta_{ik} \cdot |\hat{i}\rangle \to^* \sum_{k=0}^{2^n-1} \sum_{i=0}^{2^n-1} \alpha_k \beta_{ik} \cdot |\hat{i}\rangle \to^* \widehat{U\vec{v}}$$

- Este punto se sigue de la regla de reescritura para π.

□

Ejemplo 5.4.2 (Teleportación cuántica)**.** Veamos como ejemplo cómo implementar la teleportación cuántica descripta en los preliminares (Sección 2.2.7). Para ello necesitamos definir las funciones:

- Hadamard, H: $(?\ |+\rangle \cdot |-\rangle)$

$$\frac{\vdash (|+\rangle \perp |-\rangle) : \sharp\mathbb{B}}{\vdash (?\ |+\rangle \cdot |-\rangle) : \sharp\mathbb{B} \to \sharp\mathbb{B}}\ \textsf{UnitaryIf}$$

- Z: $(?\ -|1\rangle \cdot |0\rangle)$, donde $-|1\rangle = (-1)\cdot|1\rangle$.

$$\frac{\vdash (-|1\rangle \perp |0\rangle) : \sharp\mathbb{B}}{\vdash (?\ -|1\rangle \cdot |0\rangle) : \sharp\mathbb{B} \to \sharp\mathbb{B}}\ \textsf{UnitaryIf}$$

- X (o not): $(?\ |1\rangle \cdot |0\rangle)$,

$$\frac{\vdash (|1\rangle \perp |0\rangle) : \sharp\mathbb{B}}{\vdash (?\ |1\rangle \cdot |0\rangle) : \sharp\mathbb{B} \to \sharp\mathbb{B}}\ \textsf{UnitaryIf}$$

- curried_cnot $= (?\ (?\ |00\rangle \cdot |01\rangle) \cdot (?\ |11\rangle \cdot |10\rangle))$

$$\frac{\dfrac{\vdash |00\rangle \perp |01\rangle : \sharp\mathbb{B}^2}{\vdash (?\ |00\rangle \cdot |01\rangle) : \sharp\mathbb{B} \to \sharp\mathbb{B}^2} \quad \dfrac{\vdash |11\rangle \perp |10\rangle : \sharp\mathbb{B}^2}{\vdash (?\ |11\rangle \cdot |10\rangle) : \sharp\mathbb{B} \to \sharp\mathbb{B}^2}}{\vdash (?\ (?\ |00\rangle \cdot |01\rangle) \cdot (?\ |11\rangle \cdot |10\rangle)) : \sharp\mathbb{B} \to \sharp\mathbb{B} \to \sharp\mathbb{B}^2}\ \textsf{If}$$

Entonces definimos la parte correspondiente a Alice así:

$$\begin{aligned}\textsf{Alice} := (\lambda z.\, \pi_2((\lambda x.\, \texttt{let}\ (x_0, x_{12}) = x\ \texttt{in}\ (H x_1, x_{12}))\\ (\texttt{let}\ (x_0, x_1) = |\beta_{00}\rangle\ \texttt{in}\ (\textsf{curried\_cnot}\ \{z\} x_0, x_1)))\end{aligned}$$

donde $\{z\} := z\,|0\rangle$. En general, usaremos las notaciones $[t] := \lambda x.\, t$, $\{t\} := t\,|0\rangle$ y $[\sharp\mathbb{B}] := \mathbb{B} \to \sharp\mathbb{B}$. Al término $[t]$ lo llamamos *empaquetado*, al que *desempaquetamos* con $\{[t]\}$. Como no podemos medir distribuciones pasadas por parámetro, ya que la aplicación en tal caso se distribuye al reducir, haciendo que la abstracción donde se aplica el operador π se aplique a su vez sobre los qubits de la base, para programas como telep recurrimos a este tipo, $[\sharp\mathbb{B}]$, que permite parametrizar una distribución y aplicarle la medición.

$$\dfrac{\dfrac{x : \sharp\mathbb{B}^3 \vdash x : \sharp\mathbb{B}^3 \qquad \dfrac{\dfrac{\vdash H : \sharp\mathbb{B} \to \sharp\mathbb{B} \quad \dfrac{}{x_1 : \sharp\mathbb{B} \vdash x_1 : \sharp\mathbb{B}}\ \textsf{Ax}}{x_1 : \sharp\mathbb{B} \vdash H x_1 : \sharp\mathbb{B}}\ \textsf{App} \quad \dfrac{}{x_{23} : \sharp\mathbb{B}^2 \vdash x_{23} : \sharp\mathbb{B}^2}\ \textsf{Ax}}{x_1 : \sharp\mathbb{B}, x_{23} : \sharp\mathbb{B}^2 \vdash (H x_1, x_{23}) : \sharp\mathbb{B} \times \sharp\mathbb{B}^2}\ \textsf{Pair}}{x : \sharp\mathbb{B}^3 \vdash \texttt{let}\ (x_1, x_{23}) = x\ \texttt{in}\ (H x_1, x_{23}) : \sharp\mathbb{B} \times \sharp\mathbb{B}^2}\ \textsf{Let}}{\vdash \lambda x.\, \texttt{let}\ (x_1, x_{23}) = x\ \texttt{in}\ (H x_1, x_{23}) : \sharp\mathbb{B}^3 \to \sharp\mathbb{B} \times \sharp\mathbb{B}^2}\ \textsf{Lam} \qquad \dfrac{\sharp\mathbb{B} \times \sharp\mathbb{B}^2 \to \sharp\mathbb{B}^3}{\sharp\mathbb{B}^3 \to \sharp\mathbb{B} \times \sharp\mathbb{B}^2 \leq \sharp\mathbb{B}^3 \to \sharp\mathbb{B}^3}\ \leq}{\vdash \lambda x.\, \texttt{let}\ (x_1, x_{23}) = x\ \texttt{in}\ (H x_1, x_{23}) : \sharp\mathbb{B}^3 \to \sharp\mathbb{B}^3}\ \leq$$

$$\dfrac{\dfrac{\vdash |\beta_{00}\rangle : \sharp\mathbb{B}^2 \qquad \dfrac{\dfrac{x_1 : \sharp\mathbb{B} : x_1 : \sharp\mathbb{B} \qquad \dfrac{x_0 : \sharp\mathbb{B} \vdash x_0 : \sharp\mathbb{B} \qquad \dfrac{\vdash \text{curried\_cnot} : \sharp\mathbb{B} \to \sharp\mathbb{B} \to \sharp\mathbb{B}^2 \quad z : [\sharp\mathbb{B}] \vdash z : [\sharp\mathbb{B}]}{z : [\sharp\mathbb{B}] \vdash \text{curried\_cnot}\ \{z\} : \sharp\mathbb{B} \to \sharp\mathbb{B}^2}\ \text{App}}{z : [\sharp\mathbb{B}], x_0 : \sharp\mathbb{B} \vdash \text{curried\_cnot}\ \{z\} x_0 : \sharp\mathbb{B}^2}\ \text{App}}{z : [\sharp\mathbb{B}], x_0 : \sharp\mathbb{B}, x_1 : \sharp\mathbb{B} \vdash (\text{curried\_cnot}\ \{z\} x_0, x_1) : \sharp\mathbb{B}^2 \times \sharp\mathbb{B}}\ \text{Pair}}{z : [\sharp\mathbb{B}] \vdash \texttt{let}\ (x_0, x_1) = |\beta_{00}\rangle\ \texttt{in}\ (\text{curried\_cnot}\ \{z\} x_0, x_1) : \sharp\mathbb{B}^2 \times \sharp\mathbb{B}}\ \text{UnitaryLet} \qquad \sharp\mathbb{B}^2 \times \sharp\mathbb{B} \leq \sharp\mathbb{B}^3}{z : [\sharp\mathbb{B}] \vdash \texttt{let}\ (x_0, x_1) = |\beta_{00}\rangle\ \texttt{in}\ (\text{curried\_cnot}\ \{z\} x_0, x_1) : \sharp\mathbb{B}^3}\ \leq$$

Sea $\mathsf{alice\_hadamard} := \lambda x.\,\texttt{let}\ (x_1, x_{23}) = x\ \texttt{in}\ (H x_1, x_{23})$ y $\mathsf{alice\_cnot}(z) := \texttt{let}\ (x_0, x_1) = |\beta_{00}\rangle\ \texttt{in}\ (\text{curried\_cnot}\ \{z\} x_0, x_1)$ entonces $\mathsf{Alice} = \lambda z.\,\pi_2(\mathsf{alice\_hadamard}(\mathsf{alice\_cnot}(z)))$ y

$$\dfrac{\dfrac{\dfrac{\vdash \mathsf{alice\_hadamard} : \sharp\mathbb{B}^3 \to \sharp\mathbb{B}^3 \quad z : [\sharp\mathbb{B}] \vdash (\mathsf{alice\_cnot}(z)) : \sharp\mathbb{B}^3}{z : [\sharp\mathbb{B}] \vdash \mathsf{alice\_hadamard}(\mathsf{alice\_cnot}(z)) : \sharp\mathbb{B}^3}\ \text{App}}{z : [\sharp\mathbb{B}] \vdash \pi_2(\mathsf{alice\_hadamard}(\mathsf{alice\_cnot}(z))) : \mathbb{B}^2 \times \sharp\mathbb{B}}\ \pi}{\vdash \lambda z.\,\pi_2(\mathsf{alice\_hadamard}(\mathsf{alice\_cnot}(z))) : [\sharp\mathbb{B}] \to \mathbb{B}^2 \times \sharp\mathbb{B}}\ \text{Lam}$$

De modo que $\vdash \mathsf{Alice} : [\sharp\mathbb{B}] \to \mathbb{B}^2 \times \sharp\mathbb{B}$.
Para definir la parte que corresponde a Bob definimos primero:

- $\mathsf{Control\text{-}}Z = \begin{bmatrix} I_2 & 0 \\ 0 & Z \end{bmatrix}$
- SWAP_{23} es la matriz que *swapea* en un producto tensorial de tres qubits (o sea un vector en $\mathbb{C}^{2^3}$) el segundo con el tercero, o sea que $x \otimes z \otimes y = \mathsf{SWAP}_{23}\ x \otimes y \otimes z$.
- $\mathsf{CNOT} = \begin{bmatrix} I_2 & 0 \\ 0 & X \end{bmatrix}$
- $\mathsf{MBOB} = (I_2 \otimes \mathsf{Control\text{-}}Z)\mathsf{SWAP}_{23}(CNOT \otimes I_2)\mathsf{SWAP}_{23}$

Como MBOB es una matriz unitaria, entonces la representamos usando case definiendo su comportamiento en la base $\mathbb{C}^{2^3}$. Esta matriz es

$$\mathsf{MBOB} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

De manera que podemos definir Bob como (nótese que $-|3\rangle_3 = ((|0\rangle, |1\rangle), -|1\rangle)$ y $-|7\rangle_3 = ((|1\rangle, |1\rangle), -|1\rangle)$) :

$$\mathsf{Bob} = \lambda x.\,\mathsf{case}\ x\ \mathsf{of}\ \{$$

$$
\begin{aligned}
&|0\rangle_3 \mapsto |0\rangle_3 \\
|\ &|1\rangle_3 \mapsto |1\rangle_3 \\
|\ &|2\rangle_3 \mapsto |2\rangle_3 \\
|\ &|3\rangle_3 \mapsto -|3\rangle_3 \\
|\ &|4\rangle_3 \mapsto |5\rangle_3 \\
|\ &|5\rangle_3 \mapsto |4\rangle_3 \\
|\ &|6\rangle_3 \mapsto -|7\rangle_3 \\
|\ &|7\rangle_3 \mapsto |6\rangle_3 \\
&\}
\end{aligned}
$$

Vemos que $\vdash \mathsf{Bob} : \mathbb{B}^2 \times \sharp\mathbb{B} \to \mathbb{B}^2 \times \sharp\mathbb{B}$.
Definimos por último $\mathsf{trd} := \lambda x.\,\mathtt{let}\ (y,z) = x\ \mathtt{in}\ z$ con tipo:

$$
\dfrac{\dfrac{}{x : \mathbb{B}^2 \times \sharp\mathbb{B} \vdash x : \mathbb{B}^2 \times \sharp\mathbb{B}}\ \mathsf{A_x} \quad \dfrac{\dfrac{}{z : \sharp\mathbb{B} \vdash z : \sharp\mathbb{B}}\ \mathsf{A_x}}{y : \mathbb{B}^2, z : \sharp\mathbb{B} \vdash z : \sharp\mathbb{B}}\ \mathsf{Weak}}{\dfrac{x : \mathbb{B}^2 \times \sharp\mathbb{B} \vdash \mathtt{let}\ (y,z) = x\ \mathtt{in}\ z : \sharp\mathbb{B}}{\vdash \lambda x.\,\mathtt{let}\ (y,z) = x\ \mathtt{in}\ z : \mathbb{B}^2 \times \sharp\mathbb{B} \to \sharp\mathbb{B}}\ \mathsf{Lam}}\ \mathsf{Let}
$$

Luego, el algoritmo de telep es

$$\mathsf{telep} := \lambda z.\,\mathsf{trd}(\mathsf{Bob}(\mathsf{Alice}\ z))$$

y su tipo

$$
\dfrac{\dfrac{\vdash \mathsf{trd} : \mathbb{B}^2 \times \sharp\mathbb{B} \to \sharp\mathbb{B} \qquad \dfrac{\vdash \mathsf{Bob} : \mathbb{B}^2 \times \sharp\mathbb{B} \to \mathbb{B}^2 \times \sharp\mathbb{B} \qquad \dfrac{\vdash \mathsf{Alice} : [\sharp\mathbb{B}] \to \mathbb{B}^2 \times \sharp\mathbb{B} \quad z : [\sharp\mathbb{B}] \vdash z : [\sharp\mathbb{B}]}{z : [\sharp\mathbb{B}] \vdash \mathsf{Alice}\ z : \mathbb{B}^2 \times \sharp\mathbb{B}}\ \mathsf{App}}{z : [\sharp\mathbb{B}] \vdash \mathsf{Bob}(\mathsf{Alice}\ z) : \mathbb{B}^2 \times \sharp\mathbb{B}}\ \mathsf{App}}{z : [\sharp\mathbb{B}] \vdash \mathsf{trd}(\mathsf{Bob}(\mathsf{Alice}\ z)) : [\sharp\mathbb{B}] \to \sharp\mathbb{B}}\ \mathsf{App}}{\vdash \lambda z.\,\mathsf{trd}(\mathsf{Bob}(\mathsf{Alice}\ z)) : [\sharp\mathbb{B}] \to \sharp\mathbb{B}}\ \mathsf{Lam}
$$

Para utilizarlo, el parámetro se debe pasar “empaquetado”, ya que se debe realizar una medición.

Capítulo 6

Trabajo futuro y conclusiones

6.1. Trabajo futuro

Durante la elaboración de esta tesis se ensayaron diferentes enfoques para mantener las dos familias de tipos (lineales para qubits y no lineales para bits) junto con el comportamiento buscado en el cálculo: duplicación (y descarte) sólo para bits (no para qubits) así como mantener el cálculo en la esfera unitaria (es decir no admitir valores cuya norma no sea igual a 1).

Uno de los problemas es el de las funciones que operan sobre qubits. La regla de reducción, siguiendo la estrategia usada por Lambda-$\mathcal{S}_1$, determina que la aplicación de funciones opera únicamente con valores puros, no distribuciones, lo cual implica que no puede haber funciones que reciben qubits y, en particular, funciones capaces de medir un qubit. La función $\lambda x\,.\,\pi x$ por ejemplo aplicada a $\alpha\,|0\rangle + \beta\,|1\rangle$ es la identidad y no una medición. Este es el motivo por el cual en la implementación de la teleportación se recurrió a *empaquetar* el qubit. Así, para definir una función que mida un qubit, $\lambda x\,.\,\pi x$, se debe pedir al programador que pase el parámetro empaquetado como $\lambda x\,.\,|\psi\rangle$.

Una de las (infructuosas) estrategias ensayadas fue la de dotar al sistema con una forma diferente de abstracción, lo cual se basó en el camino tomando en Lambda-$\mathcal{S}$ que es el de incluir dos reglas de beta reducción diferentes, una para valores de tipo *bemol*, otra para superposiciones. La idea era entonces contar con, además de λ, una abstracción distinta (marcada con un signo, $\lambda^\sharp$) cuya correspondiente regla de reescritura sustituyera el parámetro incluso si era una distribución. Es decir, que permita un comportamiento como este: $(\lambda^\sharp z\,.\,\pi z)\,|\psi\rangle \longrightarrow \pi\,|\psi\rangle$. Pero también es necesario para ello que la aplicación de una función a una distribución no fuera una mera notación (ya que en tal caso estaríamos distribuyendo linealmente la función, que es lo que se buscaba evitar) sino que estuviera admitido por la gramática. Como consecuencia, es necesario agregar una regla para reducir $(\lambda x\,.\,\vec{t})\,\vec{s}$.

Es decir, el ensayo consistió en incorporar en la gramática el *pure value* $\lambda^\sharp z\,.\,\vec{t}$ así como el término distribuido: $\vec{st}$, eliminado la notación correspondiente a $\vec{st}$ de las notaciones para construcciones lineales; y agregando a las reglas de reescritura

$$\lambda^\sharp z\,.\,\vec{t}\vec{v} \longrightarrow \vec{t}\,[x := \vec{v}]$$

y

$$(\lambda x.\vec{t})\sum_{i=1}^{n}\alpha_i\cdot v_i \longrightarrow \sum_{i=1}^{n}\alpha_i\cdot(\lambda x.\vec{t})v_i,$$

así como una regla de tipado $\mathsf{App}^\sharp$ para poder tipar aplicaciones cuando las funciones están definidas para parámetros puros, a ser distribuida en una distribución:

$$\frac{\Gamma\vdash t : A\to B \quad \Delta\vdash \vec{s} : \sharp A}{\Gamma,\Delta\vdash t\vec{s} : \sharp B}\ \mathsf{App}^\sharp.$$

Todo esto condujo a un sistema que no se mantiene en la esfera unitaria. Por ejemplo, la función constante $\lambda x.\,|0\rangle$ podría definirse sobre bits (o sea de tipo $\mathbb{B}\to\mathbb{B}$) y su aplicación al qubit $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ tiparse con la regla $\mathsf{App}^\sharp$:

$$\frac{\vdash \lambda x.\,|0\rangle : \mathbb{B}\to\mathbb{B} \quad \vdash \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle : \sharp\mathbb{B}}{\vdash (\lambda x.\,|0\rangle)\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle : \sharp\mathbb{B}}\ \mathsf{App}^\sharp$$

Pero la reducción se comportaría así:

$$(\lambda x.\,|0\rangle)\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle \longrightarrow \frac{1}{\sqrt{2}}(\lambda x.\,|0\rangle)\,|0\rangle+\frac{1}{\sqrt{2}}(\lambda x.\,|0\rangle)\,|1\rangle \longrightarrow \frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|0\rangle \longrightarrow \frac{2}{\sqrt{2}}|0\rangle$$

y $|\frac{2}{\sqrt{2}}|^2 = 2 \neq 1$.

La regla de UnitaryApp que finalmente fue la que se incluyó en el sistema, similar a las de UnitaryIf y UnitaryLet, junto a su par la regla App evitan esto ya que sólo nos permiten aplicar una función cuando tanto el parámetro formal de la misma como el argumento al cual se aplica son ambos $\flat$ o en cambio ambos $\sharp$. Por ende, no tiene tipo el término $(\lambda x.\,|0\rangle)\,|+\rangle$ ya que como la x se descarta no puede ser $\sharp$. Por otra parte, también se evita aplicar una expresión como $\vdash \lambda z.\,\pi z : \sharp\mathbb{B}\to\mathbb{B}$ a una distribución como $|+\rangle$. Es decir, no puede tiparse $(\lambda z.\,\pi z)\,|+\rangle$, término que reduciría a $|+\rangle$ y que por ende no tiene tipo $\mathbb{B}$. Otro tanto ocurre con let, cuya versión unitaria también exige que el resultado sea una distribución ya que de otro modo (si fuera del mismo tipo que el cuerpo del let) tendríamos $\vdash \texttt{let}\ (x,y) = (|+\rangle, |-\rangle)\ \texttt{in}\ \pi(x,y) : \mathbb{B}^2$.

El camino adoptado por Lambda-$\mathcal{S}_1^\pi$ de empaquetar un parámetro cuando se trata de una distribución cuando se la quiere medir es una forma de manejar el problema de soportar tanto un lenguaje que se mantiene en la esfera unitaria y, a la vez, capaz de realizar mediciones. Pero esto impone algunas limitaciones a la hora es escribir programas dado que el programador tiene que tener cuidado cuando quiere medir distribuciones dentro de un programa, complejizándolos en comparación con uno que no requiera tales cuidados. Por ende, un tema de trabajo futuro es el dar con mejores alternativas para manejar este problema, ya sea mediante cambios en las reglas de tipado, reescritura y hasta gramática del lenguaje, como quizá formas más ergonómicas de escribir este tipo de funciones.

Otra mejora posible es la de extender el sistema de tipos para incluir suma de tipos (*sumtypes*) como lo hace Lambda-$\mathcal{S}_1$ dado que es una característica de los lenguajes que es bastante común debido a su aplicabilidad.

Finalmente, como trabajo a futuro puede considerarse también la implementación de un compilador para Lambda-$\mathcal{S}_1^\pi$ que genere código ejecutable en computadoras cuánticas basado en los SDK disponibles para ello como por ejemplo el paquete de *Python* `Qiskit`.

6.2. Conclusiones

En este trabajo introducimos Lambda-$\mathcal{S}_1^\pi$, un lenguaje basado en Lambda-$\mathcal{S}$ y Lambda-$\mathcal{S}_1$, y que comparte con éstos varios aspectos, como el de ofrecer la posibilidad de expresar algoritmos con control cuántico. Mientras que puede destacarse como aspecto relevante de Lambda-$\mathcal{S}$ el contar con un operador de medición, de Lambda-$\mathcal{S}_1$ puede destacarse que impone la unitariedad de las superposiciones. Lambda-$\mathcal{S}_1^\pi$ reúne ambos aspectos para que sus programas puedan efectivamente correr en un computadora cuántica, y por tanto rechazar valores de normas arbitrarias, como Lambda-$\mathcal{S}_1$, pero a la vez incluye la posibilidad de describir la medición como parte de sus programas, como Lambda-$\mathcal{S}$.

Se optó por seguir el estilo de Lambda-$\mathcal{S}_1$ de mantener un conjunto más acotado de reglas de reescritura reduciendo los términos a los cuales aplicarlas usando notaciones para las construcciones lineales y reglas de congruencia. Por otra parte, siguiendo en cambio a la manera de Lambda-$\mathcal{S}$, se incluyen los qubits de la base computacional como términos básicos, en lugar de definirlo como la suma de otros tipos básicos.

Bibliografía

[1] Pablo Arrighi and Gilles Dowek. Linear-algebraic λ-calculus: higher-order, encodings, and confluence. In Andrei Voronkov, editor, *Rewriting Techniques and Applications*, pages 17–31, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[2] Pablo Arrighi and Gilles Dowek. Lineal: a linear algebraic lambda calculus. In *Logical methods in computer science*, volume 13, pages 1–33, 2017.

[3] Ali Assaf, Alejandro Díaz-Caro, Simon Perdrix, Christine Tasson, and Benoît Valiron. Call-by-value, call-by-name and the vectorial behaviour of the algebraic λ-calculus. *Logical Methods in Computer Science*, 10(4:8), 2014.

[4] Alejandro Díaz-Caro. A quick overview on the quantum control approach to the lambda calculus. In Mauricio Ayala-Rincón and Eduardo Bonelli, editors, *Proceedings of the 16th Workshop on Logical and Semantic Frameworks with Applications (LSFA'21)*, volume 357 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–17. Open Publishing Association, 2021.

[5] Alejandro Díaz-Caro and Gilles Dowek. Typing quantum superpositions and measurement. In Carlos Martín-Vide, Roman Neruda, and Miguel A. Vega-Rodríguez, editors, *Theory and Practice of Natural Computing (TPNC 2017)*, volume 10687 of *Lecture Notes in Computer Science*, pages 281–293. Springer, Cham, 2017.

[6] Alejandro Díaz-Caro, Gilles Dowek, and Juan Pablo Rinaldi. Two linearities for quantum computing in the lambda calculus. *BioSystems*, 186:104012, 2019. Postproceedings of TPNC 2017.

[7] Alejandro Díaz-Caro, Mauricio Guillermo, Alexandre Miquel, and Benoît Valiron. Realizability in the unitary sphere. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019)*, pages 1–13, 2019.

[8] Alejandro Díaz-Caro and Octavio Malherbe. A concrete categorical semantics for Lambda-$\mathcal{S}$. In Beniamino Accattoli and Carlos Olarte, editors, *Proceedings of the 13th Workshop on Logical and Semantic Frameworks with Applications (LSFA'18)*, volume 344 of *Electronic Notes in Theoretical Computer Science*, pages 83–100. Elsevier, 2019.

[9] Alejandro Díaz-Caro and Octavio Malherbe. A concrete model for a linear algebraic lambda calculus. Draft at `arXiv:1806.09236`, 2020.

[10] Alejandro Díaz-Caro and Octavio Malherbe. Quantum control in the unitary sphere: Lambda-$\int_1$ and its categorical model. *Logical Methods in Computer Science*, 18(3:32), 2022.

[11] J.-Y. Girard. Linear logic. *Theoretical Compututer Science*, 50:1–102, 1987.

[12] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. *ACM SIGPLAN Notices (PLDI'13)*, 48(6):333–342, 2013.

[13] Michael Nielsen and Isaac Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2010.

[14] Jennifer Paykin, Robert Rand, and Steve Zdancewic. Qwire: A core language for quantum circuits. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, pages 846–858, New York, NY, USA, 2017. ACM.

[15] Peter Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[16] Peter Selinger. Lecture notes on the lambda calculus. *CoRR*, abs/0804.3434, 2008.

[17] Thorsten Altenkirch y Jonathan Grattage. A functional quantum programming language. In *Proceedings of LICS 2005*, pages 249–258. IEEE, 2005.