Universidad Autónoma del Estado de México

Facultad de Ingeniería
Doctorado en Ciencias de la Ingeniería

Lenguaje de programación cuántico QML con historial, reversibilidad y cálculo lambda con mediciones

M. Nely Plata Cesar

TESIS

Para obtener el grado de:

Doctora en Ciencias de la Ingeniería

Cómite de Tutores:

Director de Tesis	Dr. José Raymundo Marcial Romero
Co-Director de Tesis	Dr. José Antonio Hernández Servín
	Dr. César Bautista Ramos

Toluca, México, Junio de 2020

FACULTAD DE INGENIERÍA
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

Agradecimientos

Agradezco el apoyo del Consejo Nacional de Ciencia y Tecnología CONACYT para la realización de estos estudios, con número 306171. También a la Facultad de Ingeniería de la Universidad Autónoma del Estado de México.

Durante este proceso conserve, conocí y viví diversas experiencias y personas, de las cuales, las personas son lo más importante por reconocer aquí.

Quiero agradecer ilimitadamente a mi familia por estar siempre conmigo, quererme, apoyarme y por desearme lo mejor. Este trabajo sólo es una consecuencia de lo maravilloso que ustedes logran con su existencia y compañía, gracias en especial a Antonia, Antonio, Evelyn y César.

Agradezco a los profesores que durante está etapa me obsequiaron su tiempo, conocimientos y sobre todo consejos de vida que siempre guardaré; fueron y son el soporte para lograr sueños y proyectos de vida en la investigación.

Este trabajo de investigación se logró con el apoyo de Dr. J. Raymundo Marcial Romero, al cual le agradezco todo lo aportado en esta etapa, y que tenga presente que es un profesor admirable y excepcional para mi y muchos compañeros. Espero en un tiempo no muy lejano ser tan extraordinaria investigadora como usted.

Y al Dr. Alejandro Díaz-Caro, por su significante aporte, admirable conocimiento, hospitalidad y por permitirme ver su emoción y entusiasmo por el cómputo cuántico. También, a la Dra. Rosa María Valdovinos Rosas, por su guía constante durante el doctorado y en dudas existenciales, gracias.

A todos los amigos y compañeros con los que conviví, los cuales recordaré y espero preservar siempre. Esta etapa se sobrellevo mejor con su compañía y personalidad, éxito en esta etapa única, irrepetible y maravillosa que están o pasaron conmigo.

Deseo que todos estén y estemos bien, que tengan la mejor vida para ustedes, gracias!

Abstract

Research in quantum computing allows us to think about computing in a new and different way, having the ambition to solve problems more efficiently than with classical computing.

Now a days, conventional tasks cannot be practically developed in a quantum computer, however, it is possible to think and develop abstractions of them, for example through quantum programming languages, being the core of this research, in particular the Quantum Meta Language (QML).

We present a semantic model that incorporates reversibility via a history track for the quantum programming language QML, considering classical and quantum data, omitting measurements. With the history, the reversibility can be explicitly and naturally applied from the proposed rules. The language is worked first with classical data and extrapolated to quantum data.

Also, the QML language and quantum lambda calculus were linked, giving as product a syntax with quantum measurements, operational semantics, typing rules and some formal proofs.

Resumen

La investigación en el cómputo cuántico permite pensar de una forma nueva y diferente la computación, teniendo la ambición de solucionar problemas de manera más eficiente que con el cómputo clásico.

Por ahora, las tareas convencionales no se pueden desarrollar de manera práctica en una computadora cuántica, sin embargo, sí se puede pensar y desarrollar abstracciones de éstas, por ejemplo a través de lenguajes de programación cuánticos, siendo el núcleo de esta investigación, en particular el lenguaje Quantum Meta Language (QML).

Se presenta un modelo semántico que incorpora reversibilidad a través de una pista de historial para el lenguaje de programación cuántico QML, considerando datos clásicos y cuánticos, omitiendo mediciones. Con la pila de historial, la reversibilidad puede aplicarse explícita y naturalmente a partir de las reglas propuestas. El lenguaje se trabaja gradualmente, inicialmente con datos clásicos y un procedimiento similar al clásico se extrapola con datos cuánticos.

Además, se vincularon los lenguajes QML y cálculo lambda cuántico, dando como producto una sintaxis con mediciones cuánticas, semántica operacional, reglas de tipado y ciertas pruebas formales.

Índice general

The Quantum Universe has a quotation from me in every chapter - but it's a damn good book anyway.

— Richard Feynman

Introducción

Entre los años 1900 y 1920 el estudio de la naturaleza a pequeña escala, da origen a la *mecánica cuántica*, estableciendo un conjunto de reglas que permiten describir el comportamiento de ese mundo microscópico. La mecánica cuántica es diferente a la mecánica clásica ya que ésta última estudia los objetos de dimensiones convencionales y los fenómenos respectivos.

En el mundo cuántico se identificó la capacidad de estar en dos o más situaciones al mismo tiempo (superposición), sin embargo, el que un objeto esté en varias situaciones simultáneamente, genera otra novedad, y es que si algo observa tal superposición, ésta desaparece y se comporta de manera clásica. Por éstas y otras características, los atributos cuánticos resultan difíciles de comprender pero asombrosos de estudiar.

La mecánica cuántica se aplicó a diversas áreas y aproximadamente entre 1970 y 1980 se considera la idea de reunir a la física, ciencias de la computación y teoría de la información, dando origen al cómputo cuántico.

De esta nueva área surgen varios algoritmos, en los años 80's surge el teorema de no clonación cuántica, el cual determina que no es posible copiar datos cuánticos; posteriormente, David Deutsch (1985), Peter Shor (1994) y Grover (1995), evidenciaron que en complejidad computacional los algoritmos cuánticos resultan ser más eficientes que los clásicos. En 1982 Richard Feynman sugiere simular una computadora con las propiedades de la mecánica cuántica y alrededor del siglo XXI, empiezan los estudios acerca de la simulación de sistemas cuánticos.

Para el desarrollo computacional cuántico, se proponen varios niveles de abstracción, comenzando con la física cuántica teórica, seguido de circuitos cuánticos, modelos matemáticos y finalmente lenguajes de programación cuánticos.

La importancia de trabajar con lenguajes de programación cuánticos es que son una herramienta directa para interactuar el humano con una máquina, propiciando nuevas formas de procesar la información y dando pauta a una manera diferente de pensar la computación, esperando en un futuro obtener beneficios con respecto al cómputo clásico.

Por lo tanto, esta investigación se centra en el lenguaje de programación cuántico Quantum Meta Language (QML), el cuál considera datos clásicos y cuánticos,

y es el primer lenguaje que propone control cuántico. Este lenguaje se conforma de su sintaxis, modelo operacional basado en circuitos cuánticos, teoría ecuacional y una primera implementación en Haskell.

Se abordarán dos principales áreas de trabajo: la reversibilidad computacional y la cohesión con cálculo lambda cuántico. Primero, debido a la carencia de una computadora cuántica, en este trabajo se propone la modificación de la semántica de tal lenguaje que verifica y da la posibilidad de ejecutar explícitamente reversibilidad.

La reversibilidad en lenguajes de programación resulta desafiante, dado que tendrá ventajas significativas el saber qué sucede en cada paso de la ejecución de un programa, el recuperar o descartar datos (sino son necesarios), el poder regresar de donde se comenzó, etc. Para lograr esto, se determina la implementación de un historial de operaciones en conjunto con reglas para ejecutar reversibilidad, omitiendo mediciones cuánticas.

Si se amplia el panorama de reversibilidad en éste u otros lenguajes, implica por ahora, simular adecuadamente QML en una máquina clásica, y a futuro, implementarlo en una computadora cuántica y ver funcionando esta propiedad explícitamente.

También en lenguajes de programación se pueden incorporar y analizar propiedades de estos a través de otros modelos, como es el caso con cálculo lambda cuántico. Cálculo lambda cuántico λ, es un modelo utilizado en diversas disciplinas debido a su sobria definición, la cual permite explorar propiedades de las matemáticas y también aplicarlo a lenguajes de programación para modelarlos y posteriormente analizarlos y/o refinarlos.

Esta investigación vincula QML y cálculo lambda cuántico, complementando y refinando su sintaxis, a la cual se anexan mediciones cuánticas; implicando un modelo operacional semejante a cálculo lambda, con reglas de equivalencia y verificación de propiedades formales.

En síntesis, el objetivo se generaliza en incorporar un historial de operaciones al modelo semántico de QML y las reglas que garantizan reversibilidad computacional, omitiendo mediciones. También, trabajar con cálculo lambda cuántico, generando un lenguaje más intuitivo con datos y control cuántico, considerando mediciones cuánticas.

Esto nos permite establecer una hipótesis: la implementación formal de un historial de cálculos, reversibilidad y contribución de cálculo lambda cuántico con lenguaje de programación cuántico QML, resultará en un lenguaje coherente con los axiomas del cómputo cuántico, esperando ventajas con respecto a los lenguajes de programación clásico. Posibilitando por ahora, analizar teóricamente las propiedades cuánticas e implementar el lenguaje QML en una máquina clásica, y a futuro, en una computadora cuántica de manera adecuada.

Organización. La tesis cuenta con 6 capítulos y tiene la siguiente organización. En el capítulo 1, se encuentran los fundamentos de álgebra lineal y bases del cómputo cuántico, en el capítulo 2 se encuentra información de la diversidad de lenguajes de programación cuánticos y características; el capítulo 3 aborda la definición (de la literatura) del lenguaje QML; en el capítulo 4 y 5, se encuentran los resultados referentes a reversibilidad computacional y la asociación con cálculo lambda cuántico, respectivamente; en el capitulo 6, están conclusiones y trabajo a futuro; finalizando con bibliografía y apéndices.

If your model contradicts quantum mechanics, abandon it!

— Richard P. Feynman

1 Marco Teórico y Estado del Arte

Para abstraer el funcionamiento de la física cuántica, se recurre a conceptos algebraicos y cierta simbología que permiten comprenderla, esta información se encontrará en este capítulo, comenzando con definiciones algebraicas y posteriormente la formalización de la mecánica cuántica.

1.1 Nociones algebraicas

Para conocer y comprender cómo funciona la orbe de la mecánica cuántica, en particular para estudiar los postulados de ésta, se requieren de nociones algebraicas básicas, como vectores, matrices, producto interno, producto tensor, entre otros. A continuación, se tratará de resumir y guiar a través de definiciones y ejemplos, los conceptos mencionados [41].

1.1.1 Espacio vectorial

El álgebra lineal, estudia espacios vectoriales y operaciones lineales entre estos. Entonces se comenzará con saber qué es un espacio vectorial, considerando el conjunto de los números complejos $\mathbb{C}$, se procede a definir el espacio vectorial sobre $\mathbb{C}$ [26, 51].

Definición 1.1 (Espacio vectorial)**.** Un espacio vectorial V, es un conjunto de n-tuplas $(v_1, \ldots, v_n)$, donde los componentes v_i corresponden a valores complejos. Los elementos de un espacio vectorial se denominan vectores y usualmente se denotan como u, v, w. Un espacio vectorial debe satisfacer lo siguiente:

1. A cada par de vectores u, v de V, corresponde un tercer vector h de V, llamado *adición* o *suma* de u y v, escrito como $h = u + v$. Satisfaciendo:

   - Conmutatividad de la adición, $u + v = v + u$.
   - Asociatividad de la adición, $u + (v + w) = (u + v) + w$.

2. Hay un vector (único) 0 en V, llamado *vector cero*, con la propiedad de: $0 + u = u + 0 = u$.

3. Para cada $v \in V$ hay un vector (único), denotado por $-v \in V$ y llamado *inverso* de v, con la propiedad de $v + (-v) = 0$.

4. *(Propiedades de multiplicación escalar).* Cada número complejo $\kappa_1, \kappa_2 \in \mathbb{C}$, denominados *escalares*, y vectores $u, v \in V$, satisfacen: $\kappa_1(u+v) = \kappa_1 u + \kappa_1 v$; $(\kappa_1 + \kappa_2)u = \kappa_1 u + \kappa_2 u$; $(\kappa_1 \kappa_2)u = \kappa_1(\kappa_2 u)$.

   El elemento neutro del producto $1 \in \mathbb{C}$ satisface que $1u = u$.

El espacio vectorial sobre los complejos $\mathbb{C}^n$, se tomará como base, siendo el espacio de todas las n-tuplas de números complejos. Algunas de las operaciones que involucran vectores y que son significativas para este trabajo se encuentran enseguida.

Definición 1.2. [Conjugado complejo] Sea $\kappa \in \mathbb{C}$, entonces su *conjugado complejo* consiste en cambiar el signo a la parte imaginaria del mismo, es decir, - por + e inversamente, se denota como $\overline{a}$.

Si consideramos un espacio con vectores complejos, entonces, para para tales vectores se menciona la operación de transpuesto conjugado.

Definición 1.3. [Vector transpuesto conjugado] Sea un vector $v \in \mathbb{C}^n$, el *transpuesto conjugado* de v consiste en cambiar el renglón por columna y aplicar a cada $\kappa \in v$ su conjugado complejo, se denota como v^*.

Ejemplo 1.1. Sea el vector $v = \begin{pmatrix} 2 - 6i \\ 2 \\ 5 + 2i \end{pmatrix}$, su transpuesto conjugado es $v^* = (2 + 6i,\ 2,\ 5 - 2i)$.

Con respecto a matrices, el cómputo cuántico considera a estas como el objeto que le permite plasmar instrucciones, entonces, para realizar operaciones con matrices se requiere de las definiciones siguientes.

Definición 1.4 (Matriz transpuesta)**.** Sea $A = [a_{ij}] \in \mathbb{C}^{m\times n}$ una matriz de m columnas y n renglones, su *transpuesta* consiste en cambiar las filas por columnas, es decir, $A^T = [a_{ji}] \in \mathbb{C}^{n\times m}$, se denota como A^T.

Definición 1.5 (Matriz transpuesta conjugada)**.** Sea una matriz $A = [a_{ij}] \in \mathbb{C}^{m\times n}$, la transpuesta conjugada es $A^* = [\overline{a_{ji}}] \in \mathbb{C}^{n\times m}$; se obtiene a partir de su transpuesta y después el conjugado complejo de cada valor (Definición 1.2), esto es $A^* = \overline{A^T}$.

Definición 1.6 (Matriz adjunta o hermitiana)**.** Una matriz hermitiana o adjunta, es aquella matriz A que satisface: $A^* = A$, es decir, si es igual a su transpuesta conjugada.

Definición 1.7 (Matriz normal)**.** Sea A una matriz, si $AA^* = A^*A$, entonces es normal.

Definición 1.8 (Matriz simétrica)**.** Si A es una matriz cuadrada que satisface $A = A^T$, entonces, es simétrica.

Definición 1.9 (Matriz unitaria)**.** Una matriz cuadrada U (con igual número de de filas y columnas) es unitaria si: $U^*U = UU^* = I$, donde, I es la matriz identidad.

Las definiciones previas, permitirán comprender operaciones de vectores y matrices, son importantes dado que en secciones posteriores, se asociarán con cálculos cuánticos.

Los conceptos algrebráicos en el ambiente cuántico, tiene una simbología estándar llamada *notación de Dirac*. Ésta se introducirá a partir de ahora para interpretar naturalmente las secciones posteriores.

1.1.2 Notación de Dirac

Para introducir gradualmente al contexto de la mecánica cuántica, se adopta la notación de Dirac $|\cdot\rangle$, que indica que un objeto $\cdot$ es un vector. En este estilo de notación, usaremos convencionalmente los símbolos ψ y φ para denotar vectores, resultando en $|\psi\rangle$ y $|\varphi\rangle$, respectivamente.

En la Tabla 1.1 se sintetizan los conceptos algebráicos y la notación de Dirac.

Los conceptos siguientes se definirán implementando la notación previa, comenzando con producto interno.

Notación	Descripción
κ^*	Complejo conjugado del número complejo κ.
	$(1+i)^* = 1-i$
$\|\psi\rangle$	Vector. También nombrado como ket.
$\langle\psi\|$	Transpuesto conjugado del vector $\|\psi\rangle$.
	Conocido como bra.
$\langle\varphi\|\psi\rangle$	Producto interno entre los vectors $\|\varphi\rangle$ y $\|\psi\rangle$.
$\|\varphi\rangle \otimes \|\psi\rangle$	Producto tensor de $\|\varphi\rangle$ y $\|\psi\rangle$.
$\|\varphi\rangle \|\psi\rangle$ ó $\|\varphi\psi\rangle$	Abreviación del producto tensor de $\|\varphi\rangle$ y $\|\psi\rangle$.
A^*	Complejo conjugado de la matriz A.
A^T	Transpuesto de la matriz A.
$A^\dagger$	Conjugado hermitiano o adjunto de la matriz A, $A^\dagger = (A^T)^*$
	$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^\dagger = \begin{bmatrix} a^* & c^* \\ b^* & d^* \end{bmatrix}$
$\langle\varphi\| A \|\psi\rangle$	Producto interno entre $\|\varphi\rangle$ y A$\|\psi\rangle$.
	Equivalente al producto interno entre $A^\dagger\|\varphi\rangle$ y $\|\psi\rangle$.

Tabla 1.1: Resumen de la notación de Dirac [41].

1.1.3 Producto interno

El producto interno entre vectores, es la función $(\cdot,\cdot) : V \times V \to \mathbb{C}$, donde V es un espacio vectorial. El producto interno, se denotará como $\langle\cdot|\cdot\rangle$, y toma dos vectores $|\psi\rangle$ y $|\varphi\rangle$, que producen un número complejo, satisfaciendo las siguientes condiciones.

Para todo $\kappa_i \in \mathbb{C}$ y $\psi, \varphi \in \mathbb{C}^n$:

1. Linealidad por la derecha: $\langle\psi| \sum_i \kappa_i\varphi_i\rangle = \sum_i \kappa_i \langle\psi|\varphi_i\rangle$.
2. Hermiticidad: $\langle\psi|\varphi\rangle = \langle\psi|\varphi\rangle^*$.
3. Definida positiva: $\langle\psi|\psi\rangle \geq 0$, con $|\psi\rangle \neq 0$ y $\langle\psi|\psi\rangle = 0$, si y sólo si $|\psi\rangle = 0$.

A partir de la información previa, se puede definir ortogonalidad.

Definición 1.10. Dos vectores $|\psi\rangle$ *y* $|\varphi\rangle$ son *ortogonales* entre sí, cuando satisfacen que su producto interno es cero, $\langle\psi|\varphi\rangle = 0$. La ortogonalidad de tales vectores, también se puede denotar como $\psi \perp \varphi$.

Para la mecánica cuántica, primordialmente los vectores de interés son aquellos que satisfacen que que su norma sea 1, esto corresponde a que:

Definición 1.11 (Norma de un vector)**.** Considerar el espacio vectorial $\mathbb{C}^n$, entonces, la norma se define como la función $\mathbb{C}^n \to \mathbb{R}$, y se denota como $\|\|$. Esto es, sea un vector $|\psi\rangle = \begin{pmatrix} \kappa_1 \\ \vdots \\ \kappa_n \end{pmatrix}$, su norma se calcula como: $\||\psi\rangle\| = \sqrt{|\kappa_1|^2 + \cdots + |\kappa_n|^2}$, donde $|\cdot|$ es el módulo de cada $\kappa_i \in \mathbb{C}$. Si $\kappa_i = a + bi \in \mathbb{C}$ entonces su *módulo* es: $|a+bi| = \sqrt{a^2+b^2}$. La norma también se define en términos del producto interno, que es: $\||\psi\rangle\| = \sqrt{\langle\psi|\psi\rangle}$.

Por lo tanto, en un espacio vectorial, un conjunto de vectores que son ortogonales entre sí y cada uno tienen norma 1, se dice que son vectores *ortonormales*.

Hasta este punto las operaciones conllevan elementos de un mismo espacio vectorial, sin embargo, también se requieren operaciones entre espacios, para lo cual se menciona el producto tensorial.

1.1.4 Producto tensorial

Posibilita formar o unir espacios vectoriales a partir de otros espacios vectoriales, esto es importante en el contexto cuántico, porque permite que interactúen diferentes sistemas cuánticos entre sí.

Definición 1.12 (Producto tensorial)**.** Sean V y W espacios vectoriales de dimensión m y n, respectivamente, el *producto tensor* o también llamado *producto de Kronecker* se denota por $V \otimes W$, resultando un espacio de dimensión m por n.

Los elementos de $V \otimes W$, son productos tensores $|\psi\rangle \otimes |\varphi\rangle$ formados por $|v\rangle$ de V y $|\varphi\rangle$ de W. El producto tensorial tiene las siguientes propiedades:

- Para cualquier escalar κ y vectores $|\psi\rangle \in V$, $|\varphi\rangle \in W$,

$$\kappa(|\psi\rangle \otimes |\varphi\rangle) = (\kappa\,|\psi\rangle) \otimes |\varphi\rangle = |\psi\rangle \otimes (\kappa\,|\varphi\rangle)$$

- Para vectores arbitrarios $|\phi\rangle, |\psi\rangle \in V$ y $|\varphi\rangle \in W$,

$$(|\phi\rangle + |\psi\rangle) \otimes |\varphi\rangle = |\phi\rangle \otimes |\varphi\rangle + |\psi\rangle \otimes |\varphi\rangle$$

- Para vectores arbitrarios $|\psi\rangle \in V$ y $|\varphi\rangle, |\varphi'\rangle \in W$,

$$|\psi\rangle \otimes (|\varphi\rangle + |\varphi'\rangle) = (|\psi\rangle \otimes \varphi + |\psi\rangle \otimes |\varphi'\rangle)$$

- El producto tensorial entre las matrices A, B y vectores $|\psi\rangle, |\varphi\rangle$, está dado por: $(A \otimes B)(|\psi\rangle \otimes |\varphi\rangle) = (A\,|\psi\rangle \otimes B\,|\varphi\rangle)$.

- Si A y B, son matrices, entonces, $(A \otimes B)^* = A^* \otimes B^*$.
- Si A y B son matrices unitarias, entonces $A \otimes B$ es unitario.

El producto tensorial entre dos matrices, da como resultado una matriz de bloque. Si se tiene la matriz $A \in \mathbb{C}^{m\times n}$, y $B \in \mathbb{C}^{s\times k}$, la operación se efectúa como:

$$A \otimes B = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{pmatrix} \otimes \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1k} \\ B_{21} & B_{22} & \cdots & B_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ B_{s1} & B_{s2} & \cdots & B_{sk} \end{pmatrix}$$

$$= \left(\begin{array}{cccc|c|cccc} A_{11}B_{11} & A_{11}B_{12} & \cdots & A_{11}B_{1k} & \cdots & A_{1n}B_{11} & A_{1n}B_{12} & \cdots & A_{1n}B_{1k} \\ A_{11}B_{21} & A_{11}B_{22} & \cdots & A_{11}B_{2k} & \cdots & A_{1n}B_{21} & A_{1n}B_{22} & \cdots & A_{1n}B_{2k} \\ \vdots & \vdots & \ddots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ A_{11}B_{s1} & A_{11}B_{s2} & \cdots & A_{11}B_{sk} & \cdots & A_{1n}B_{s1} & A_{1n}B_{s2} & \cdots & A_{1n}B_{sk} \\ \hline \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hline A_{m1}B_{11} & A_{m1}B_{12} & \cdots & A_{m1}B_{1k} & \cdots & A_{mn}B_{11} & A_{mn}B_{12} & \cdots & A_{mn}B_{1k} \\ A_{m1}B_{21} & A_{m1}B_{22} & \cdots & A_{m1}B_{2k} & \cdots & A_{mn}B_{21} & A_{mn}B_{22} & \cdots & A_{mn}B_{2k} \\ \vdots & \vdots & \ddots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ A_{m1}B_{s1} & A_{m1}B_{s2} & \cdots & A_{m1}B_{sk} & \cdots & A_{mn}B_{s1} & A_{mn}B_{s2} & \cdots & A_{mn}B_{sk} \end{array}\right)_{block}$$

$$= \begin{pmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & \ddots & \vdots & \\ A_{m1}B & A_{m2}B & \cdots & A_{mn}B \end{pmatrix}_{block},$$

Por otra parte, el producto tensorial entre vectores, se puede visualizar de la siguiente forma: Si se tienen los vectores $|\psi\rangle \in \mathbb{C}^n$ y $|\varphi\rangle \in \mathbb{C}^m$, entonces $|\psi\rangle \otimes |\varphi\rangle$ es:

$$|\psi\rangle \otimes |\varphi\rangle = \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_n \end{pmatrix} \otimes \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} \kappa_1\lambda_1 \\ \kappa_1\lambda_2 \\ \vdots \\ \kappa_1\lambda_m \\ \vdots \\ \kappa_n\lambda_1 \\ \kappa_n\lambda_2 \\ \vdots \\ \kappa_n\lambda_m \end{pmatrix} \in \mathbb{C}^{n\times m}$$

Por ejemplo, el producto tensorial de los vectores $(1,0)^T$ y $(0,1)^T$, es: $\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$. Y el producto tensorial de las matrices $A = \begin{pmatrix} 1 & 3 \\ 2 & 6 \end{pmatrix}$ y $B = \begin{pmatrix} 3 & 1 \\ 1 & 4 \end{pmatrix}$,

sería: $A \otimes B = \begin{pmatrix} 1 & 3 \\ 2 & 6 \end{pmatrix} \otimes \begin{pmatrix} 3 & 1 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 3 & 1 & 9 & 3 \\ 1 & 4 & 3 & 12 \\ 6 & 2 & 18 & 6 \\ 2 & 8 & 6 & 24 \end{pmatrix}$.

Notación 1.1. El producto tensorial $|\psi\rangle \otimes |\varphi\rangle$, se suele simplificar como $|\psi\rangle\,|\varphi\rangle$ ó $|\psi\varphi\rangle$.

Con las bases de álgebra lineal y familiarizados con la notación de Dirac, se procede a darle sentido al por qué fue importante mencionar tales definiciones, y esto es, abordar el formalismo de la mecánica cuántica.

1.2 Mecánica cuántica

Al estudiar el comportamiento cuántico, se notaron algunos de los sucesos más conocidos, como la capacidad de estar en dos o más situaciones simultáneamente, lo cual se llamo superposición de estados, resaltando que esto no sucedía en lo clásico; otra notable situación, era que si se está en superposición y un fenómeno de la naturaleza interfiere, entonces esta superposición deja de existir y se comporta como la naturaleza clásica.

Las propiedades cuánticas detectadas se concentran en cuatro postulados o axiomas que dan estructura a esta nueva teoría: estados puros, evolución, medición y estados compuestos, cada uno de estos, tiene un vínculo con álgebra lineal, la cual permite abrastraerlos. Estos proveen una conexión entre el mundo físico y el formalismo matemático de la mecánica cuántica.

Los axiomas se establecieron después de un largo proceso de prueba y error, implicando una considerable cantidad de conjeturas y falsificaciones por parte de los creadores de la teoría. No se sorprenda si la motivación para los postulados no siempre es clara; tales parecen sorprendentes incluso para los expertos.

La teoría cuántica al pasar de los años fue aplicada a diversas disciplinas, como la informática, dando una nueva manera de concebir y pensar la computación. La computación nos a instruido que el procesamiento de información es a través de bits, sin embargo, ahora se puede realizar con qubits (siendo la unidad de información de computadoras cuánticas).

A continuación se describirán los postulados cuánticos, los cuales en cualquier disciplina se definen igual, pero se interpretan de acuerdo al área de interés. Lo que se espera de las próximas secciones es obtener una buena comprensión de los postulados y cómo se conciben en un entorno computacional.

1.2.1 Postulados de la computación cuántica

Se comienza describiendo que la *computación cuántica* se define como el procesamiento de información que efectúa tareas aplicando los principios de la mecánica cuántica [41, 36].

Los postulados siguientes se presentan y conforman parte de un sistema cuántico. Un *sistema cuántico* es aquella porción de la naturaleza que satisface las propiedades cuánticas. El sistema que se considera es *aislado* o *cerrado*, es decir, que no existe ninguna interferencia con su exterior.

La importancia de establecer que el sistema cuántico sea cerrado es que considerando que cualquier factor de la naturaleza que interfiera con tal, podría colapsar el sistema y en consecuencia no llegar al estado final, es conveniente que el sistema sea cerrado.

Postulado 1. Estados del sistema

Un *estado puro* o simplemente *estado* de un sistema cuántico, es un vector $|\psi\rangle = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \in \mathbb{C}^n$, tal que $\||\psi\rangle\| = 1$, es decir, que el vector $|\psi\rangle$ tiene norma 1.

Los estados puros son los qubits, descritos con la notación de ket $|\cdot\rangle$. Recordando que el ket $|\psi\rangle$, es un vector y corresponde a la notación de Dirac (Tabla 1.1); el uso de ket o qubit se usará de manera equivalente a partir de ahora.

Los estados base en el cómputo cuántico son el ket cero $|0\rangle$ y ket uno $|1\rangle$, los cuales se asocian con el bit 0 y 1, respectivamente. Denotados como: $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Aquellos vectores o n-qubit se definen concretamente como: $\sum_{i=0}^{2^n-1} \lambda_i |\psi_i\rangle$ satisfaciendo: $\sum_{i=0}^{2^n-1} |\lambda_i|^2 = 1$, $\lambda_i \in \mathbb{C}$ y que cada tupla de vectores $|\psi_i\rangle$ sean ortogonales entre sí.

Computacionalmente, un qubit guarda información en la memoria de una computadora en un momento determinado.

Definición 1.13. Una combinación lineal $|\varphi\rangle = \sum_{i=1}^{n} \lambda_i |\psi_i\rangle$, es llamada *superposición de estados*, y los valores λ_i, *amplitudes de probabilidad*. Esto se interpreta como la propiedad de estar en varios estados al mismo tiempo, es decir, estar en $|\psi_1\rangle$, $|\psi_2\rangle$ hasta $|\psi_n\rangle$ simultáneamente, con un probabilidad asociada λ_i, respectivamente.

Los qubits que explícitamente no indican una amplitud de probabilidad, corresponde al valor 1, por ejemplo, $|0\rangle$, tiene probabilidad de 1.

Lo anterior corresponde al postulado de estados cuánticos; ahora se procede a describir el axioma de evolución.

Postulado 2. Evolución del sistema

Este postulado define cómo un sistema cuántico va evolucionando. Suponer que una computadora se encuentra en el estado $|\psi_1\rangle$ y se aplica una instrucción U, entonces cambia a un nuevo estado $|\psi_2\rangle$; de formalizar esto se encarga este postulado.

Los estados de un sistema *evolucionan* a través de una transformación unitaria, aplicando ésta a un qubit y como resultado se tiene un nuevo estado. Formalmente:

- $|\psi_1\rangle \in \mathbb{C}^n$, estado actual del sistema.
- $U \in \mathbb{C}^{m\times n}$, una matriz unitaria.
- $|\psi_2\rangle = U\,|\psi_1\rangle$, el siguiente estado del sistema.

Las *compuertas* o *instrucciones* de una computadora cuántica están descritas en matrices unitarias, esto es, si se tiene determinada información en memoria (estado puro $|\psi_1\rangle$), la forma de cambiar (evolucionar) el contenido en memoria, es aplicar una matriz unitaria U al estado actual $|\psi_1\rangle$ obteniendo $|\psi_2\rangle = U\,|\psi_1\rangle$, donde $|\psi_2\rangle$ es lo que tiene en memoria la computadora después de aplicar la instrucción U.

Un ejemplo trivial, es tener el qubit $|0\rangle$, el cual se desea invertir a $-\,|0\rangle$, para esto se tendría que aplicar el operador unitario $M = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$, resultando en:

$$M\,|0\rangle = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} = -\,|0\rangle$$

Otro ejemplo es la compuerta u operación de Hadamard, ésta es relevante porque posibilita pasar del contexto clásico a una superposición de estados, por ejemplo:

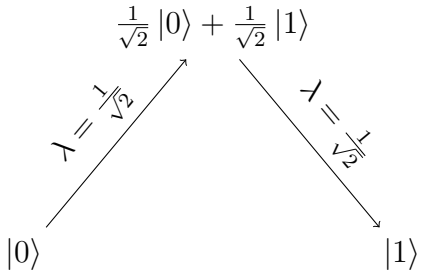
$$H\,|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle$$

Las instrucciones (matrices unitarias) se permiten aplicar las veces necesarias hasta llegar al estado deseado, cuándo se esté en un estado final, se requiere el obtener un resultado, para lo cual, se define el postulado de medición u observación.

Postulado 3. Medición u observación

La naturaleza cuántica indica que si se está en un estado y es interferido por algún fenómeno externo como luz, ruido, aire, observador, etc, éste colapsa a un estado clásico. El fenómeno externo que interfiere se le denomina *observador*, el *colapso* corresponde a dejar de estar en varios estados simultáneamente y sólo permanecer en uno (emite un resultado), y al acto previo al colapso se le llama *medir u observar*.

Para comprender esto, considerar que se tiene la superposición: $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$,



Si se observa el qubit $|\psi\rangle$, se tienen dos posibles salidas $|0\rangle$ o $|1\rangle$, con amplitud de probabilidad $\frac{1}{\sqrt{2}}$. Este postulado, describe el cómo modelar este colapso.

La medición es probabilística, ya que la naturaleza cuántica no permite determinar con exactitud a cuál estado va a colapsar el sistema, pero sí la probabilidad de que suceda cualquiera de los eventos posibles. Esto se formaliza a continuación [41, 62].

1. Cuando se mide un estado $|\varphi\rangle = \lambda_1|\psi_1\rangle + \lambda_2|\psi_2\rangle + \cdots + \lambda_n|\psi_n\rangle$, para todo $1 \leq m \leq n$ se asocia un operador de medición M_m. Todos los operadores de medición satisfacen la *ecuación de completez* o también conocida como *ecuación de normalización*.

$$M_{m_1}^* M_{m_1} + M_{m_2}^* M_{m_2} + \cdots + M_{m_k}^* M_{m_k} = I, \tag{1.1}$$

esta condición garantiza, que la suma de las probabilidades de observar los diferentes resultados sea igual a 1.

Ejemplo 1.2. Si se tiene un sistema en el estado $|\psi\rangle = \dfrac{1}{\sqrt{2}}|0\rangle + \dfrac{1}{\sqrt{2}}|1\rangle$. Al observar el sistema, se puede colapsar a dos eventos: al qubit $|0\rangle$ (el cual asociaremos al evento esperado m_1) o al qubit $|1\rangle$ (asociado con m_2).

Si sucede el colapso a $|0\rangle$, se le asocia la matriz $M_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, y al valor esperado $|1\rangle$, la matriz $M_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$.

Dadas estas matrices, se debe verificar que satisfacen la ecuación de completitud:

$$M_0^* M_0 + M_1^* M_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}^* + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}^* = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Por lo tanto, las matrices previas satisfacen la ecuación de completez y se continúa con el siguiente paso.

2. Si $|\psi\rangle$ es el estado del sistema antes de ser observado, una vez efectuada la medición, el sistema eventualmente colapsa a un estado:

$$\frac{M_{m_i} |\psi\rangle}{\| M_{m_i} |\psi\rangle \|}, \tag{1.2}$$

con probabilidad $P(m_i)$, donde: $P(m_i) = \langle\psi| M_{m_i}^* M_{m_i} |\psi\rangle$. La probabilidad se puede calcular también como $P(m_i) = \|\psi\|^2$.

Ejemplo 1.3. Retomando el qubit: $|\psi\rangle = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$ con operadores de medición $M_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ y $M_1 =, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. Ahora, si un observador mide el estado, no se puede determinar cuál de los dos eventos sucederá, sin embargo, para fines de este ejemplo, suponer que el colapso es a $|0\rangle$, entonces lo que sucede es (Ec. 1.2):

$$\frac{M_{m_i} |\psi\rangle}{\| M_{m_i} |\psi\rangle \|} = \frac{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}}{\sqrt{|\frac{1}{\sqrt{2}}|^2}} = \frac{\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix}}{\frac{1}{\sqrt{2}}} = |0\rangle$$

Con probabilidad $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$. Y analogamente sucedería con $|1\rangle$.

Se sigue con la descripción del último postulado: sistemas compuestos.

Postulado 4. Sistemas compuestos

Este postulado aborda la composición de espacios de estados a través del producto tensorial ($\otimes$). Suponer el estado $|\psi\rangle$ de un sistema cuántico A, y el estado $|\phi\rangle$ del sistema cuántico B, entonces $|\psi\rangle \otimes |\phi\rangle$ es el *estado del sistema cuántico compuesto* por A y B.

Ejemplo 1.4. Para establecer un estado compuesto, realizar el producto tensor de ambos. Considerar el qubit $|0\rangle$ del sistema cuántico A y el estado $|1\rangle$ del sistema B, el estado compuesto es:

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

Cabe resaltar la importancia del producto tensorial, si se tiene un estado puro compuesto $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_k\rangle$ y se desea evolucionar el sistema, entonces se efectúa el producto tensorial de matrices (dichas matrices tienen implícitas las instrucciones) de acuerdo al estado que se desea afectar, es decir, $M_1 \otimes M_2 \otimes \cdots \otimes M_k$, donde $M_i, 1 \leq i \leq k$ influirá al ket $|\psi_i\rangle$, respectivamente. Si algún estado $|\psi_i\rangle$ no se desea cambiar, entonces la instrucción M_i que se deberá aplicar es la matriz identidad. Esto se visualiza en la Figura 1.1.

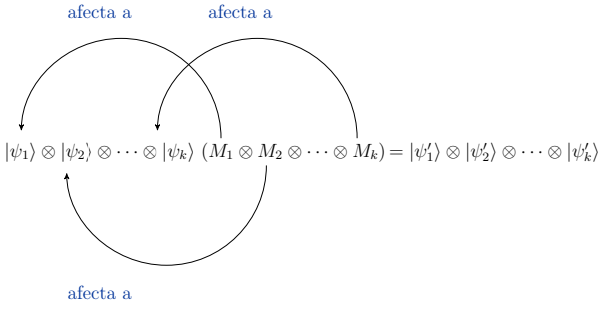


Figura 1.1: Producto tensor entre vectores y matrices.

Dados los cuatro postulados elementales, se procede a mencionar información acerca de *reversibilidad computacional*, siendo un tema vinculado con operaciones cuánticas y de interés para esta investigación.

1.2.2 Reversibilidad

La reversibilidad, es la propiedad de las operaciones cuánticas, tal que si se está en un estado $|\psi\rangle$, y se aplica la misma operación que originó tal salida, entonces se regresa al estado inmediato anterior [41]. Para esto, matemáticamente se recurre a la definición de matriz inversa o no singular.

Definición 1.14 (Matriz inversa)**.** Una matriz M es invertible (o no singular) si existe una matriz N tal que $MN = NM = I$. La matriz N si existe, es única y se denomina *matriz inversa de* M y se denotará por M^{-1} [41].

Esta definición se vincula con el concepto de matriz unitaria, ya que toda matriz unitaria, satisface la condición de tener inversa.

Por ejemplo, suponer que se está en el estado $|0\rangle$ y se aplica la matriz X o también llamada inversor:

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

Y si se aplica de nuevo la matriz X al qubit $|1\rangle$, se vuelve al estado original:

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

Entonces, desde el punto de vista de compuertas cuánticas, la reversibilidad es una propiedad inherente del cómputo cuántico. Esto, si se extiende a lenguajes de programación cuánticos, también deberá satisfacerse.

Algunas de las motivaciones en el cómputo reversible provienen de poder conocer qué sucede en cada cálculo, así como detectar errores intermedios y corrección de los mismos, poder ejecutar acciones biyectivas, incrementar la eficiencia de un proceso, descartar acciones, evitar o promover perdida de información (posiblemente innecesaria), etc.

Considerar que actualmente la implementación de algún lenguaje cuántico (por ausencia de computadora cuántica), implicaría que la reversibilidad no se pueda ejecutar naturalmente, por lo tanto, deberá ser modelada a través de alguna estrategia.

De la revisión de la literatura se mencionarán algunos de los resultados de nuestro interés. Landauer es pionero en esta área, quien aporta la idea de que una computadora puede efectuar la reversibilidad si se guarda su historial de cómputos [33, 42].

Otros investigadores sobresalientes son Abramsky y Bennett [3], mostrando que el cómputo clásico puede ser transformado en cómputo reversible [10]. Partiendo de tales autores, Van Tonder [58] retoma sus ideas, en particular de Bennett [10], para incorporar el historial de operaciones en cálculo lambda cuántico A.1.2, tal historial se encarga de plasmar los pasos realizados en ciertos procesos, como podrían ser las instrucciones ejecutadas durante un programa computacional; esta último trabajo será un sustento para la presente investigación.

Con estos antecedentes se procede a conocer investigaciones acerca de lenguajes cuánticos de programación, algunos simuladores; para posteriormente profundizar en el lenguaje QML.

I tried reading Hilbert. Only his papers published in mathematical periodicals were available at the time. Anybody who has tried those knows they are very hard reading.

— Alonzo Church

2

Lenguajes de programación cuánticos

Diversos autores han propuesto lenguajes de programación cuánticos, aplicando modelos matemáticos para su formalización, como lógica, teoría de categorías, cálculo lambda, entre otros. La implementación de éstos también se está investigando, dando pie a los simuladores cuánticos [37, 54, 57].

Un punto importante a considerar al desarrollar un lenguaje es sobre qué modelo de Hardware se implementará, es decir, las características físicas del dispositivo en el que funcionará dicho lenguaje. Los modelos más aceptados son: Modelo de circuito cuántico, el QRAM y la Máquina de Turing Cuántica; la mayoría de lenguajes se basa en el modelo de circuito y QRAM.

En la literatura existen lenguajes de programación cuánticos basados en los paradigmas imperativo, funcional, entre otros; a continuación se dará información acerca de estos.

Para proponer un lenguaje cuántico, idealmente se sugiere satisfacer los requerimientos o características de [37]:

Completez: Algún circuito cuántico, que incluya algún paradigma clásico.

Extensibilidad: Desarrollar algoritmos que requieran cálculos clásicos.

Separabilidad: Separar partes clásicas de cuánticas.

Expresividad: Proveer elementos de alto nivel para la codificación de algoritmos complejos.

Independencia: Ser independiente de la implementación física de una computadora cuántica, es decir, poder compilar un programa en diferentes computadoras con la misma arquitectura o distinta.

A continuación, se describen algunos de los lenguajes de programación cuánticos imperativos.

2.0.2 Lenguajes de programación cuánticos imperativos

El surgimiento de estos lenguajes se presenta con el precursor Knill, que en 1996 propone convenciones para plasmar algoritmos cuánticos en un pseudocódigo. Provee una sintaxis para describir qubits y operaciones con información clásica y cuántica, carece de estructura formal y representa sólo ciertas propiedades cuánticas [30, 54, 57].

Entre los años 1998 - 2003, Ömer desarrolla el lenguaje QCL (Quantum Computation Language). Éste es considerado el prime r lenguaje de programación cuántico, utiliza sintaxis derivada de C, permite describir operadores definidos por el usuario, calcula la inversa de algunos operadores, maneja algunos condicionales con operadores y es implementado en un simulador [43, 54, 57, 21].

Bettelli, et. al. desarrolla un lenguaje basado en C++, maneja la construcción y optimización de operadores cuánticos (matrices) y aplica transformaciones como Hadamard, QFourier, QNot, etc; describe comportamientos en su simulador [11].

El lenguaje qGCL (quantum Guarded Command Language) propuesto por Sanders y Zuliani, proponen una semántica para su lenguaje, formalizando los lenguajes de programación [53, 54, 57]. Posteriormente Zuliani continua la investigación de programación no determinista y con estados mixtos [21].

LanQ, es propuesto por Mlnarik en 2007, el cual se caracteriza por soportar operaciones cuánticas y clásicas, provee una sintaxis completa formalizada (similar al lenguaje C), diseña una semántica operacional, elimina ciertos errores y cuenta con un simulador [38, 57]. Otro lenguaje es Quantum Imperative Language (QIL), el cual toma como referencia el trabajo de Abramsky [2], obteniendo como aporte su semántica operacional [47].

Existen más propuestas de este tipo de lenguajes, sin embargo, sólo se mencionarón algunos representativos. Enseguida se describen lenguajes funcionales.

2.0.3 Lenguajes de programación funcionales

Entre los primeros trabajos se encuentra el desarrollado por Maymin en 1996, proponiendo un cálculo lambda cuántico para investigar la cómputabilidad de algoritmos cuánticos [35]. Amr Sabry simula cálculos cuánticos usando un lenguaje funcional clásico (Haskell) [52, 57].

Mu y Bird en 2001, modelan en Haskell programación cuántica con registros cuánticos [40]. Derivado de la investigación de Sabry, que extiende tal modelo para incorporar la representación de estados entrelazados [52]. Del resultado de tales trabajos, se extienden investigaciones de Vizzotta y da Rocha Costa [60], Karczmarczuk [29] y Skibinski [56].

Otro aporte importante sucede en 2004 con Selinger, quien define QFC (Quantum Flow Charts). Representa programas a través de un diagrama de flujo, utiliza una sintaxis textual llamada QPL (Quantum Programming Language), manipula datos clásicos y cuánticos, define operaciones y mediciones, no verifica errores en tiempo de ejecución y posee una semántica denotacional completa. La importancia de éste, radica en su formalismo matemático, siendo sustento de otros lenguajes de programación cuánticos [55, 57].

Danos en 2004 estudia un modelo de un camino, que incluye notación de entrelazamiento, mediciones y correcciones locales, basada en patrones [14]. Posteriormente Danos y Kashefi, describen condiciones para patrones de medición para ejecutar de manera determinista [12]. Así como en colaboración, desarrollan un cálculo de mediciones de dos qubits, relevantes para la teleportación [13].

Quantum Meta Language (QML) es un lenguaje para cálculos clásicos y cuánticos desarrollado por Altenkirch y Grattage, su primer propuesta tiene características como: Modela operaciones reversibles e irreversibles, utiliza datos cuánticos y control clásico, emplea teoría de categorías y operacionalmente se basa en circuitos cuánticos, presenta una formalización matemática de sus propiedades y desarrolla un simulador (deficiente) en Haskell [24].

Posteriormente Altenkirch, Grattage, Vizzotto y Sabry continúan la investigación de QML, donde retoman sólo la sintaxis de éste y definen otra perspectiva del modelo operacional, definiendo una teoría ecuacional, con pruebas de completez y validez [4]. QML será el lenguaje que se abordará de manera puntual, incorporando una semántica para reversibilidad computacional, así como una colaboración con cálculo lambda cuántico, generando nuevas reglas de tipado, modelo operacional y pruebas convenientes. Siendo esta propuesta la base de la presente investigación.

Lampis, Ginis y Paapaspyrou trabajan en un lenguaje que nombran como nQML, basado en el lenguaje QML [4], el cual se caracteriza por ampliar a datos cuánticos y control cuántico, también ajusta la semántica denotacional [32].

En 2005 Perdrix establece un sistema que modela el entrelazamiento de estados cuánticos, éste se encuentra basado en cálculo lambda [46, 57, 21].

Arrighia y Dowekc, proveen un lenguaje que se extiende de cálculo lambda incorporando combinaciones lineales de términos lambda [7]. También se encuentran Lago, Masini y Zorzi quienes proveen una equivalencia computacional del cálculo λ no tipado con una clase sustituta de circuitos cuánticos [31]. Zorzi continua la investigación previa, con el paradigma de datos cuánticos y control clásico [63].

Van Tonder propone el cálculo lambda cuántico (λ_q) para operaciones cuánticas puras. El lenguaje λ aplica características cuánticas al cálculo lambda clásico. Las operaciones se ejecutan teniendo un historial de lo efectuado, no implementa mediciones [58, 59, 57, 54, 21].

Posteriormente, Díaz-Caro, Arrighi, Gadella y Grattage en 2011 continúan con esta investigación, agregando mediciones, y dando una estrategia para incorporar mediciones en otros lenguajes, ajustando sintaxis y ciertas reglas establecidas, así como modelar cálculos reversibles conservando su historial de operaciones [17].

Cálculo λ vectorial, desarrollado en 2017 por Arrighi, Díaz-Caro, Valiron, consiste en un sistema de tipos con aspectos algebráicos lineales de λ_q. Capaz de especificar estáticamente las combinaciones lineales de términos que se obtendrán al reducir los programas; también modela cómo codificar matrices y vectores en este cálculo [6]. La importancia de tal investigación radica en que continua con el desarrollo de cálculo λ_q y se pueden implementar dichos aportes a diversos lenguajes de programación cuánticos.

En otros lenguajes de programación cuánticos se encuentran: QSPEC desarrollado por Papanikolaou en 2004, el cual soporta procesos concurrentes [44]. Mauerer en 2005 propone el lenguaje cQPL el cual está basado en la investigación de Selinger [55], definiendo su sintaxis, semántica y simulador, del cual extiende la programación de sistemas distribuidos y protocolos de comunicación [34].

Hablando en general de lenguajes de programación, para su desarrollo y estudio se encuentra la semántica, encargada de dar una descripción y significado formal de su funcionamiento. La semántica se aborda en dos perspectivas, la operacional y denotacional; para aportar la formalidad de éstas se recurre convencionalmente a varios modelos: Lógica, teoría de categorías, cálculo lambda, etc [21].

Con base al estudio de varios lenguajes de programación cuánticos, en la Tabla 2.1 se resumen propiedades generales de algunos estos, identificando características y a qué paradigma pertenecen.

Lenguaje	Semánt. Oper.	Semánt Denot.	Paradigma	Mediciones	Reversibilidad	Implementación	Historial
QML [4]	✓	✓	Funcional		✓	✓	
nQML [32]	✓	✓	Funcional		✓	✓	
Cálculo λ_q[58]	✓		Funcional	✓	✓		✓
QCL [43]			Imperativo	✓	✓	✓	
LanQ [38]	✓		Imperativo	✓	✓	✓	
qGCL [53]	✓		Imperativo	✓			
pQCL [21]		✓	Otro				
QPAlg [28]		✓	Otro				
CQP [22, 23]		✓	Otro				

Tabla 2.1: Descripción de lenguajes de programación cuánticos.

Considerando el hecho de que el imitar el comportamiento de éstos, es otra área importante de trabajo, dado que permitirá modelar e implementar la simulación adecuada de dichos lenguajes cuánticos. A continuación, se mencionarán brevemente algunos simuladores.

2.0.4 Simuladores cuánticos

Como resultado de conocer, aprender, ejemplificar y evidenciar visualmente el comportamiento del procesamiento cuántico, se están desarrollando los simuladores cuánticos. Existen diversos simuladores, algunos de ellos como resultado de la implementación de un lenguaje cuántico de programación, y otros, como herramienta didáctica de aprendizaje y comprensión de un entorno cuántico, esto último con cierta ausencia de formalidad computacional.

Durante los años 90, se desarrollaron algunos como Q-gol 3 (1996), Mathematica notebook simulations (1998), Quantum Turing Machine Simulator (1999), desarrollado estos en Mathematica, basados principalmente en describir operaciones cuánticas, algoritmos, matrices, etc. Entre otros se encuentran Eqcs 0.0.5 (1999), Parallel Quantum Computer Simulator (1996), LQP2 (1997) y M. Hayward's Shor Algorithm Simulation (1999) [61, 57, 20, 15, 45, 38].

Otro simulador importante es LanQ (2007), basado en su propio lenguaje imperativo. Entre algunos de los simuladores más recientes se encuentran: FJ y FJQuantum, siendo orientado a objetos. Y QuIDE Simulator (2015) basado en Java, modelando compuertas y algoritmos cuánticos, éste ya presenta una interfaz completa respecto a cálculos y circuitos [61, 57, 19, 20, 15, 45, 38].

A continuación, se procede con la información referente al lenguaje de programación cuántico QML.

The true Logic for this world is the Calculus of Probabilities, which takes account of the magnitude of the probability.

— James Clerk Maxwell

3 Lenguaje de programación QML

En esta sección se describirá el lenguaje de programación cuántico *Quantum Meta Language*, abreviado como *QML*, al cual se efectuarán modificaciones correspondientes a los aportes de esta investigación. Se comenzará con características generales del mismo.

3.1 QML

Es un lenguaje basado en el paradigma funcional, desarrollado para cálculos cuánticos sobre tipos finitos y no recursivos, establece datos clásicos y cuánticos, y es el primer lenguaje que propone cálculos cuánticos.

Este lenguaje primero es propuesto por Altenkirch y Grattage, quienes se basan en teoría de categorías, definen programas *estrictos o puros* ($\vdash^{\circ}$, sin mediciones) y *no estrictos* ($\vdash$, con mediciones) y su semántica operacional está dada en términos de circuitos cuánticos, que conlleva operaciones reversibles e irreversibles. Es decir, traslada derivaciones de QML a cómputos cuánticos representados como circuitos. Semánticamente los cómputos cuánticos reversibles se modelan con operadores unitarios. Los cómputos irreversibles envuelven probabilidades y son modelados como funciones lineales sobre matrices de densidad, llamadas *superoperadores* [24].

Posteriormente Altenkirch, Grattage, Vizzotto y Sabry retoman sólo la estructura pura del QML (sintaxis), al cual definen una semántica a partir de funciones y tripletas de kleisli (datos cuánticos); no incluyen explícitamente mediciones cuánticas pero existe la regla *weakening*, que permite descartar un término pero sólo para los que no acarrean información. También proponen una teoría ecuacional para el sublenguaje clásico y para el cuántico, así como pruebas de validez y completez [5].

Para esta investigación se reconsidera la perspectiva del último trabajo propuesto por Altenkirch, Grattage, Vizzotto y Sabry [5], a la cual, se le establecerá una pila de historial de operaciones, que posibilitarán realizar reversibilidad explícita, incluso en una computadora cuántica. Esto garantizará la reversibilidad del lenguaje.

3.2 Sintaxis y reglas de escritura

La sintaxis determina los términos que conforman al lenguaje, este lenguaje considera datos cuánticos como superposiciones $t + u$ y términos que tienen asociado una probabilidad de amplitud κt.

(Variables)	$x, y, \ldots$	$\in Vars$
(Amplitudes prob.)	$\kappa, \iota, \ldots$	$\in \mathbb{C}$
(Patterns)	p, q	$::= x \mid (x, y)$
(Terms)	t, u	$::= x \mid () \mid (t, u) \mid$ **let** $p = t$ **in** $u \mid false$ $\mid true \mid \vec{0} \mid \kappa t \mid t + u \mid$ **if**$^\circ$ t **then** u **else** u'

Tabla 3.1: Sintaxis de QML (*Fuente: [5], p. 26*).

Donde x, y corresponden a variables; κ, ι son números complejos llamados amplitudes de probabilidad (Definición 1.13); los términos podrán ser programas formados por variables, tuplas, **let**, **if**$^\circ$, con constantes $true$, $false$ y $\vec{0}$, este último corresponde al término con probabilidad de amplitud cero (vector cero).

La importancia de incorporar valores complejos κ, ι al lenguaje, es porque el mundo cuántico es probabilístico, entonces a cada evento factible se debe asociar un valor que indique la probabilidad de que suceda. A estos términos se les asocia un tipo, para lo cual se define lo siguiente.

3.2.1 Tipos en QML

Los tipos están dados a partir de la gramática $\sigma = \mathcal{Q}_1 | \mathcal{Q}_2 | \sigma \otimes \tau$. Para asignar un tipo a un término, se introduce la definición de *contexto de tipos*.

Definición 3.1 (Contexto de tipos)**.** Los *contextos* corresponden a funciones que van de un conjunto finito de variables a tipos, se podrán denotar convencionalmente por Γ o Δ; para las definiciones posteriores se utilizará especialmente Γ.

Están dados por $\Gamma = \bullet | \Gamma, x : \sigma$; considerando lo siguiente: $\bullet$, es el contexto vacío, se asume que cada variable definida, aparece al menos una vez en un programa y no define tipos recursivos.

El operador $\otimes$ es el producto tensorial y mapea pares de contextos a contextos:

Si un contexto Γ asiga tipos a un conjunto de variables, y a partir de esas variables ya tipadas se puede deducir o derivar un término t, entonces, esto se denota como $\Gamma \vdash t : \sigma$, donde $\vdash$ hace referencia a la derivación de un término t con tipo σ a partir del contexto Γ.

Si se tiene $\Gamma \vdash t : \sigma$, se interpreta (denotado con $[\![\ ]\!]$) como una función $[\![\Gamma \vdash t : \sigma]\!] \in [\![\Gamma]\!] \to [\![\sigma]\!]$, donde $[\![\Gamma]\!]$ y $[\![\sigma]\!]$ son conjuntos finitos. Continuando, los contextos pueden ser representados a partir de igualdades dadas por ciertas condiciones, éstas son:

$\Gamma, x : \sigma \otimes \Delta, x : \sigma \quad = (\Gamma \otimes \Delta), x : \sigma$
$\Gamma, x : \sigma \otimes \Delta \quad = (\Gamma \otimes \Delta), x : \sigma$ si $x \notin dom\Delta$
$\bullet \otimes \Delta \quad = \Delta$

En este punto, los términos tipados se pueden deducir a partir de contextos, y se podrían generar tantos términos como sean ideados, sin embargo no es adecuado; para esto existen reglas para tipar o formar correctamente los términos del lenguaje, en la Tabla 3.2 se establecen las reglas correspondientes.

$$\frac{}{x : \sigma \vdash x : \sigma}\text{var} \qquad \frac{}{\bullet \vdash () : Q_1}\text{unit} \qquad \frac{\Gamma \vdash t : \sigma \qquad \Delta, x : \sigma \vdash u : \tau}{\Gamma \otimes \Delta \vdash \textbf{let } x = t \textbf{ in } u : \tau}\text{let}$$

$$\frac{\Gamma \vdash c : Q_2 \qquad \Delta \vdash t, u : \sigma}{\Gamma \otimes \Delta \vdash \textbf{if}^\circ\ c\ \textbf{then}\ t\ \textbf{else}\ u : \sigma}\textbf{if}^\circ \qquad \frac{\Gamma \vdash t : \sigma \qquad \Delta \vdash u : \tau}{\Gamma \otimes \Delta \vdash (t, u) : \sigma \otimes \tau} \otimes \text{intro}$$

$$\frac{}{\bullet \vdash false : Q_2}\text{f-intro} \qquad \frac{}{\bullet \vdash true : Q_2}\text{t-intro}$$

$$\frac{\Gamma \vdash t : \sigma \otimes \tau \qquad \Delta, x : \sigma, y : \tau \vdash u : \rho}{\Gamma \otimes \Delta \vdash \textbf{let } (x, y) = t \textbf{ in } u : \rho} \otimes -\text{elim} \qquad \frac{\Gamma, x : Q_1 \vdash t : \sigma}{\Gamma \vdash t : \sigma}\text{wk-unit}$$

Tabla 3.2: Reglas de términos clásicos bien tipados (*Fuente: [5], p. 29*).

Con estas reglas, se pueden crear programas con elementos clásicos como $true$ o $false$, o con cuánticos como superposición de estos. La primera percepción del lenguaje, los autores la dan en el subconjunto de términos únicamente clásicos, definiendo lo siguiente.

3.3 Datos y control clásico

Inicialmente se trabaja el sublenguaje que considerará sólo elementos clásicos, para el cual, los autores definen su modelo semántico [5].

3.3.1 Semántica para datos clásicos

El modelo semántico (Tabla 3.3) describe las reglas de derivación basada en composiciones y funciones auxiliares (3.3.1) que permitirán ir evolucionando un programa hasta llegar a un resultado.

$$
\begin{aligned}
[\![\bullet \vdash () : Q_1]\!] &= const\ 0 \\
[\![\bullet \vdash false : Q_2]\!] &= const\ 0 \\
[\![\bullet \vdash true : Q_2]\!] &= const\ 1 \\
[\![x : \sigma \vdash x : \sigma]\!] &= id_+ \\
[\![\Gamma \otimes \Delta \vdash \textbf{let}\ x = t\ \textbf{in}\ u : \sigma]\!] &= g \circ (f \times id) \circ \delta_{\Gamma,\Delta} \\
&\quad \text{donde: } f = [\![\Gamma \vdash t : \sigma]\!] \\
&\quad\quad\ \ g = [\![\Delta,\, x : \sigma \vdash u : \tau]\!] \\
[\![\Gamma \otimes \Delta \vdash (t,u) : \sigma \otimes \tau]\!] &= (f \times g) \circ \delta_{\Gamma,\Delta} \\
&\quad \text{donde: } f = [\![\Gamma \vdash t : \sigma]\!] \\
&\quad\quad\ \ g = [\![\Delta \vdash u : \tau]\!] \\
[\![\Gamma \otimes \Delta \vdash \textbf{let}\ (x,y) = t\ \textbf{in}\ u : \rho]\!] &= g \circ (f \times id) \circ \delta_{\Gamma,\Delta} \\
&\quad \text{donde: } f = [\![\Gamma \vdash t : \sigma \otimes \sigma]\!] \\
&\quad\quad\ \ g = [\![\Delta,\, x : \sigma,\, y : \tau \vdash u : \rho]\!] \\
[\![\Gamma \otimes \Delta \vdash \textbf{if}^\circ\ c\ \textbf{then}\ t\ \textbf{else}\ u : \sigma]\!] &= (g|h) \circ (f \times id) \circ \delta_{\Gamma,\Delta} \\
&\quad \text{donde: } f = [\![\Gamma \vdash c : Q_2]\!] \\
&\quad\quad\ \ g = [\![\Delta \vdash t : \sigma]\!] \\
&\quad\quad\ \ h = [\![\Delta \vdash u : \sigma]\!] \\
[\![\Gamma \vdash t : \sigma]\!] &= f \times id^+ \\
&\quad \text{donde: } f = [\![\Gamma,\, x : Q_1 \vdash t : \sigma]\!]
\end{aligned}
$$

Tabla 3.3: Representación de derivaciones clásicas - Parte I (*Fuente [5], p. 31*).

Funciones auxiliares

Las funciones siguientes permitien establecer una semántica para los programas formados por este lenguaje; las funciones son invocadas en cada derivación. Antes de plasmar cada una, considerar: sea $a, a', b \in S$, $S = \{0, 1\}$.

- $id : S \to S$, definida por: $id(a) = a$
- $id^+ : S \to [\![Q_1]\!] \times S$, definida por: $id^+(a) = (0, a)$
- $id_+ : [\![Q_1]\!] \times S \to S$, definida por: $id_+(0, a) = a$

- $const\ a : [\![Q_1]\!] \to S$, definida por: $(const\ a)(0) = a$
- $\delta : S \to (S, S)$, definida por: $\delta(a) = (a, a)$
- $swap : S \times T \to T \times S$, definida por: $swap(a, b) = (b, a)$
- Sean las funciones $f : S_1 \to T_1$ y $g : S_2 \to T_2$, entonces
  $f \times g : S_1 \times S_2 \to T_1 \times T_2$, se define por:$f \times g\ (a, b) = (f\ a, g\ b)$.
- $\delta_{\Gamma,\Delta} : [\![\Gamma \otimes \Delta]\!] \to [\![\Gamma]\!] \times [\![\Delta]\!]$ definida por inducción como:
  $$\delta_{\Gamma,\Delta} = \begin{cases} \delta_{\Gamma',\Delta'} \times \delta & \text{if } \Gamma = \Gamma', x : \sigma \text{ y } \Delta = \Delta', x : \sigma \\ \delta_{\Gamma',\Delta} \times id & \text{if } \Gamma = \Gamma', x : \sigma \text{ y } x \notin dom(\Delta) \\ id^{+} & \text{if } \Gamma = \bullet \end{cases}$$
- Para cualesquiera función $f, g \in S \to T$, $(f|g) \in S \to T$, el condicional se define como: $(f|g)\ (1, a) = (f\ a)$ y $(f|g)\ (0, a) = (g\ a)$.

Las reglas y funciones anteriores permiten que un programa devuelva un valor clásico, por ejemplo 0 ó 1, pero para datos cuánticos, estas salidas deberán ser concebidas como estados cuánticos, para lo cual, se define lo siguiente.

3.3.2 Datos y control cuántico

Para abordar las reglas respectivas, se parte de que $\Gamma \vdash t : \sigma$ se interpreta como $[\![\Gamma]\!] \to [\![\sigma]\!]$, pero cuánticamente ahora los tipos σ deben ser asociados a vectores unitarios con una amplitud de probabilidad asociada ($\kappa \in \mathbb{C}$). Para esto, se implementa la definición de Tripletas de Kleisli, que permitirá asociar $[\![\sigma]\!]$ como un vector apropiado.

Cuánticamente $\Gamma \vdash t : \sigma$ se interpreta como funciones $[\![\Gamma]\!] \to [\![\sigma]\!]^{Q}$, donde $[\![\sigma]\!]^{Q}$ denota vectores complejos sobre el conjunto base $[\![\sigma]\!]$. Esto es: $[\![\sigma]\!]^{Q}$ corresponde a $[\![\sigma]\!] \to \mathbb{C}$, denotado también como $V[\![\sigma]\!]$.

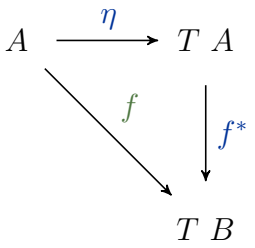
La importancia de $[\![\sigma]\!]^{Q}$ radica en que los estados de un sistema cuántico deben ser vectores unitarios y este caso no es la excepción. Por lo tanto, se puede decir que $[\![\Gamma]\!] \to [\![\sigma]\!]^{Q}$, considera un conjunto de variables bien tipadas y devuelve un vector complejo. Como se mencionó, las Tripletas de Kleisli posibilitarón esto, por lo tanto a continuación se describen.

Tripletas de Kleisli

Esta definición requiere de información referente a Teoría de Categorías, lo cual se encuentra en el Apendice C.

Definición 3.2. Una tripleta de Kleisli sobre una categoría $\mathcal{C}$ es $(T, \eta, \_^*)$, donde:

- $T : \text{Obj}(\mathcal{C}) \to \text{Obj}(\mathcal{C})$
- $\eta : A \to TA$, para $A \in \text{Obj}(\mathcal{C})$
- $f^* : TA \to TB$, para $f : A \to TB$



Satisfaciendo ciertas condiciones (Definición C.1). Los componentes de esta definición deberán ser asociados al lenguaje QML.

- La transformación η es la operación *unidad* definida como $return : S \to VS$, esto es: $return :: S \to S \to \mathbb{C}$. $\forall a, b \in S$, $return\ (a)(b) = \begin{cases} 1.0 & \text{si a=b} \\ 0.0 & \text{otro caso} \end{cases}$
- La operación *lift* o *levantamiento (.*)* está descrita como: Sea una función $f : S \to VT$, entonces el *levantamiento* de f, está dado por: $f^* : VS \to VT$. Esto posibilita trabajar entre vectores.

Este ajuste se realiza para todas las funciones auxiliares. Por ejemplo, la función *const* queda como $const\ v : [\![Q_1]\!] \to V\ S$. La composición de funciones $g \circ f$ ahora está dada por $g^* \circ f$. Las composiciones del modelo operacional también deberán ser modificados. El producto $f \times g$ usará el producto tensorial sobre vectores, esto es: $(f \times g)(a, b)(x, y) = (f\ a\ x)(g\ b\ y)$.

Entonces, todas las funciones auxiliares (Sección 3.3.1) son extrapoladas para que retornen vectores, si se tiene la función $f : S \to S$, entonces la nueva función estará dada por $f : S \to V\ S$. El resultado de todas las funciones con este ajuste se verán detalladamente en el capítulo de resultados.

Observación. Los ajustes previamente mencionados, los autores lo asocian a la definición de Monada (Sección C.0.4), dado que ésta se puede representar a través de los componentes de una Tripleta de Kleisli.

Proposición 3.1. *Hay una correspondencia uno a uno entre tripleta de Kleisli y Monadas (Sección C.0.5).*

Otra propiedad que deben satisfacer los estados cuánticos es la ortogonalidad entre sí y que la norma de cualquier estado sea 1. Por ejemplo considere la expresión $t + u$, en este t y u deben ser ortogonales, otro caso está en la estructura **if**$^{\circ}c$ **then** t **else** u, donde c debe ser una superposición de los términos t y u y estos deben cumplir también la condición de ortogonalidad.

Ortogonalidad $\perp$

Dos términos t, u son *ortogonales* si su producto interno es igual a cero, $\langle t|u\rangle = 0$. Dos términos ortogonales se denotan como $t \perp u$.

Las sentencias de producto interno sobre términos de QML, se definen con base a las reglas que se encuentran en la Tabla 3.4. Considerar que la notación λ^* representa el complejo conjugado de la probabilidad $\lambda \in \mathbb{C}$.

$\langle t\|t\rangle = 1$ si $t \neq \overrightarrow{0}$	$\langle \lambda t + \lambda' t'\|u\rangle = \overline{\lambda}\langle t\|u\rangle + \overline{\lambda'}\langle t'\|u\rangle$
$\langle false\|true\rangle = 0$	$\langle t\|\kappa u + \kappa' u'\rangle = \kappa\langle t\|u\rangle + \kappa'\langle t\|u'\rangle$
$\langle true\|false\rangle = 0$	$\langle \lambda t\|u\rangle = \overline{\lambda}\langle t\|u\rangle$
$\langle \overrightarrow{0}\|true\rangle = 0 = \langle true\|\overrightarrow{0}\rangle$	$\langle t\|\lambda u\rangle = \lambda\langle t\|u\rangle$
$\langle \overrightarrow{0}\|false\rangle = 0 = \langle false\|\overrightarrow{0}\rangle$	$\langle t + t'\|u\rangle = \langle t\|u\rangle + \langle t'\|u\rangle$
$\langle \overrightarrow{0}\|x\rangle = 0 = \langle x\|\overrightarrow{0}\rangle$	$\langle t\|u + u'\rangle = \langle t\|u\rangle + \langle t\|u'\rangle$
$\langle (t,t')\|(u,u')\rangle = \langle t\|u\rangle\langle t'\|u'\rangle$	$\langle t\|u\rangle =?$, otro caso

Tabla 3.4: Producto interno y ortogonalidad [5].

$\langle t|u\rangle =?$, representa al producto interno que no se puede reducir alguna de las formas de la tabla respectiva, entonces se desconoce su valor final al cual asociamos el símbolo ?.

Ejemplo 3.1. Sea la superposición de estados: $-\frac{1}{\sqrt{2}}\ true + \frac{1}{\sqrt{2}}\ false$, donde $\langle true|false\rangle = 0$ y $\left\|-\frac{1}{\sqrt{2}}\right\|^2 + \left\|\frac{1}{\sqrt{2}}\right\|^2 = 1$, por lo tanto es un programa válido.

3.3.3 Semántica para datos cuánticos

Teniendo presente que los elementos cuánticos deben satisfacer las condiciones previas, procedemos con la semántica para datos cuánticos (Tabla 3.5), donde las reglas respectivas para programas cuánticos bien formados están en la Tabla 3.6.

$[\![\bullet \vdash \vec{0} : \sigma]\!]^{\mathrm{Q}} = const\ v$	$\forall a \in [\![\sigma]\!], v\ a = 0.0$
$[\![\Gamma \vdash \kappa t : \sigma]\!]^{\mathrm{Q}} = (\kappa) f$	$\forall r \in [\![\Gamma]\!], a \in [\![\sigma]\!]$, tal que $g\ r\ a = \kappa(f\ r\ a)$ $f = [\![\Gamma \vdash t : \sigma]\!]^{\mathrm{Q}}$
$[\![\Gamma \vdash t + u : \sigma]\!]^{\mathrm{Q}} = \frac{1}{\sqrt{2}}\big(f + g\big)$	$\forall r \in [\![\Gamma]\!], a \in [\![\sigma]\!]$, tal que $h\ r\ a = \frac{1}{\sqrt{2}}(f\ r\ a + g\ r\ a)$ $f = [\![\Gamma \vdash t : \sigma]\!]^{\mathrm{Q}}$ $g = [\![\Gamma \vdash u : \sigma]\!]^{\mathrm{Q}}$

Tabla 3.5: Modelo semántico para datos cuánticos (Parte II).

Considerando que ahora un programa devuelve un vector (por Tripletas de Kleisli), entonces, v corresponde a éste. Por ejemplo, para el caso de $[\![\bullet \vdash \vec{0} : \sigma]\!]^{\mathrm{Q}} = const\ v$, está representando que el término $\vec{0}$ es igual a la función $const\ v$, donde $v \in V\ S$ es cualquier vector y con cualquier argumento a siempre será igual a 0.

$$\frac{\Gamma \vdash^{\circ} c : Q_2 \quad \Delta \vdash^{\circ} t, u : \sigma}{\Gamma \otimes \Delta \vdash^{\circ} \mathbf{if}^{\circ}\ c\ \mathbf{then}\ t\ \mathbf{else}\ u : \sigma}\mathbf{if}^{\circ} \qquad \frac{}{\bullet \vdash \vec{0} : \sigma}\text{z-intro}$$

$$\frac{\Gamma \vdash^{\circ} t, u : \sigma \quad t \perp u \quad |\lambda|^2 + |\kappa|^2 = 1}{\Gamma \vdash^{\circ} \lambda t + \kappa u : \sigma}\sup^{\circ} \qquad \frac{\Gamma \vdash t : \sigma}{\Gamma \vdash \kappa t : \sigma}\text{prob}$$

$$\frac{\Gamma \vdash^{\circ} t : \sigma \quad \Gamma \vdash t \equiv u : \sigma}{\Gamma \vdash^{\circ} u : \sigma}\text{subst} \qquad \frac{\Gamma \vdash t, u : \sigma}{\Gamma \vdash t + u : \sigma}\sup$$

Tabla 3.6: Reglas de términos cuánticos bien tipados.

La información anterior permite definir los componentes y la forma de operar de QML, ahora se finalizará con la teoría ecuacional de términos clásicos y cuánticos.

3.3.4 Teoría ecuacional cuántica

Las ecuaciones clásicas se agrupan en cuatro categorías:

- **let**-equation

  let $p = val$ **in** $u \equiv u[val/p]$

- η-equations
  $() \equiv t$ si $t : Q_1$
  let $x = t$ **in** $x \equiv t$
  let $(x, y) = t$ **in** $(x, y) \equiv t$
  if$^\circ$ t **then** $true$ **else** $false \equiv t$

- β-equations
  let $(x, y) = (t, u)$ **in** $e \equiv$ **let** $x = t$ **in let** $y = u$ **in** e
  if$^\circ$ $false$ **then** t **else** $u \equiv u$
  if$^\circ$ $true$ **then** t **else** $u \equiv t$

- Conversiones conmutativas
  let $p = t$ **in let** $q = u$ **in e** $\equiv$ **let** $q = u$ **in let** $p = t$ **in** e
  let $p =$**if**$^\circ$ t **then** u_0 **else** $u_1 \equiv$ **if**$^\circ$ t **in** e **then let** $p = u_0$ **in** e **else let** $p = u_1$ **in** e

Las ecuaciones cuánticas son:

- **(if**$^\circ$**)**
  if$^\circ$ $(\lambda t_0 + \kappa t_1)$ **then** u_0 **else** $u_1 \equiv$ λ(**if**$^\circ$ t_0 **then** u_0 **else** u_1)
  $+\kappa$(**if**$^\circ$ t_1 **then** u_0 **else** u_1)

- **(Superposiciones)**
  $t + u \equiv u + t$
  $t + \vec{0} \equiv t$
  $t + (u + v) \equiv (t + u) + v$
  $\lambda(t + u) \equiv \lambda t + \lambda u$
  $\lambda t + \kappa t \equiv (\lambda + \kappa)t$
  $0t \equiv \vec{0}$

Teniendo como preámbulo lo anterior, se procede a abordar los resultados de esta investigación. Recordando que la primera etapa del trabajo, es la incorporación de una pila de historial a QML, así como la ejecución y verificación de reversibilidad explícita en el lenguaje.

The most important application of quantum computing in the future is likely to be a computer simulation of quantum systems, because that's an application where we know for sure that quantum systems in general cannot be efficiently simulated on a classical computer.

— David Deutsch

4
Resultados
Historial y reversibilidad en QML

La siguiente información corresponde al resultado de anexar una pila de historial al lenguaje de programación QML, permitiendo reversibilidad computacional. La incorporación de un historial de cálculos se realizó primero en el lenguaje con datos clásicos y posteriormente se extendió a los datos cuánticos.

La reversibilidad computacional en la mecánica cuántica está en su naturaleza y en el desarrollo de lenguajes de programación no debería ser la excepción, sin embargo, la accesibilidad a una computadora cuántica aún no es posible a pesar del gran trabajo en tal área; entonces si existiera una implementación, ésta sería aún en una computadora clásica, propiciando investigar esta propiedad en QML y garantizando que dicho lenguaje es reversible.

Durante la revisión de la literatura se encontró que una computadora puede efectuar reversibilidad si se guarda su historial de cómputos [33, 42]. Esta idea se modela con cálculo lambda cuántico, al cual, Van Tonder [58] incorpora un historial de operaciones y reglas que garantizan que tal lenguaje es reversible.

La idea de Van Tonder, promueve en este trabajo la intención de agregar una pila de historial al lenguaje QML, permitiendo realizar reversibilidad explícita (cómo si se estuviera en un entorno clásico). La idea primordial es ¿Cómo hacerlo en QML? Una generalización del proceso se observa en la Figura 4.1.

La idea es que si el modelo operacional está basado en composición de funciones, entonces, sería adecuado guardar las funciones inversas respectivas (para saber qué se ha ejecutado), esperando que una vez que un programa devuelva un valor, y

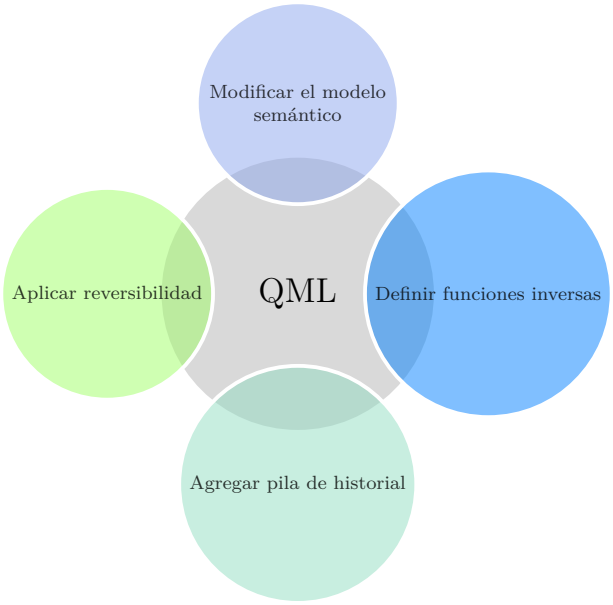


Figura 4.1: Etapas para agregar reversibilidad a QML.

a éste se le apliquen las funciones inversas en orden adecuado, sea inminente la reversibilidad.

Se comenzó modificando el modelo semántico del lenguaje, agregando funciones inversas a éste, posteriormente proponer las funciones inversas; esto implicaba la generación de una pila o pista de historial de operaciones, dada tal pila se definieron reglas de reversibilidad para finalmente ver esta propiedad explícita.

Para visualizar la dificultad de definir funciones inversas, agregarlas y ordenarlas se describe en un ejemplo, para posteriormente dar pie con los resultados que garantizan que el lenguaje QML es reversible.

Problemática

La primera idea es que si el modelo operacional está basado en composición de funciones, entonces, sería adecuado guardar las funciones inversas respectivas (para saber que se ha ejecutado), esperando que una vez que un programa devuelva un valor, y a éste se le apliquen las funciones inversas en orden adecuado, sea inminente la reversibilidad.

Por ejemplo, sea un programa cualquiera con ciertas derivaciones:

Argumento inicial → función 1 → función 2 → ··· → función n → Valor final

La reversibilidad se desarrollaría como:

Valor final → f. inversa n → ··· → f. inversa 2 → f. inversa 1 → Argumento inicial

Partiendo de este primer bosquejo, la primera etapa para anexar una pila de historial es definir las funciones inversas del lenguaje y almacenarlas en la pila. Esto se visualizaba en el siguiente ejemplo, considerando las reglas semánticas ya establecidas del lenguaje (Tabla 3.3):

Ejemplo 4.1. Operación *not*. A partir del modelo operacional de QML se obtiene:

$$not\ c = \mathbf{if}^{\circ}\ \text{false}\ \mathbf{then}\ \text{false}\ \mathbf{else}\ \text{true}$$

$$[\![\bullet \otimes \bullet \vdash \mathbf{if}^{\circ} false\ \mathbf{then}\ false\ \mathbf{else}\ true : Q_2]\!] = (g|h) \circ (f \times id) \circ \delta_{\Gamma,\Delta}$$

donde:

$$\begin{aligned} f &= [\![\bullet \vdash false : Q_2]\!] = const\ 0 \\ g &= [\![\bullet \vdash false : Q_2]\!] = const\ 0 \\ h &= [\![\bullet \vdash true : Q_2]\!] = const\ 1 \\ \delta_{\Gamma,\Delta} &= id^{+}, \quad \text{tal que } \Gamma = \bullet \end{aligned}$$

$$\begin{aligned} [\![\bullet \otimes \bullet \vdash not : Q_2]\!] &= \Big(const\ 0 \mid const\ 1\Big) \circ \Big(const\ 0 \times id\Big) \circ \Big(id^{+}\Big) \\ &= \Big(const\ 0 \mid const\ 1\Big) \circ \Big(const\ 0 \times id\Big) \circ id^{*}(0) \\ &= \Big(const\ 0 \mid const\ 1\Big) \circ \Big(const\ 0 \times id\Big)(0,0) \\ &= \Big(const\ 0 \mid const\ 1\Big) \circ \Big(const\ 0\ (0),\ id(0)\Big) \\ &= \Big(const\ 0 \mid const\ 1\Big)(0,0) \\ &= (const\ 1)\ 0 = 1\ \text{(esto es true)} \end{aligned}$$

Con estas observaciones se determina que:

1. La primera etapa es alcanzar un programa con la forma siguiente:

$$\underbrace{\boldsymbol{h_{t_n}-;\cdots;h_{t_2}-;h_{t_1}-}}_{\text{Operaciones inversas}};\quad \underbrace{t_1 \circ t_2 \circ \cdots \circ t_n}_{\text{Operaciones del programa}}$$

   En la parte izquierda se esperan almacenar las operaciones inversas (de cualquier programa) y la parte derecha las instrucciones del programa.

2. Las funciones inversas que aparecen en las reglas operacionales respectivas; en este caso son $f = const\ 0,\ g = const\ 0,\ h = const\ 1,\ id$ y id^{+}.

   - Sea $const\ a : [\![\mathcal{Q}_1]\!] \to S$, donde: $const\ a(0) = a$, su inversa se deduce como:
     $const\ a_{-}^{-1} : S \to [\![\mathcal{Q}_1]\!]$, definida como: $const\ a^{-1}(a) = 0$.

- $id : S \to S$, entonces $id_{\_}^{-1} : S \to S$, definida como: $id^{-1}(a) = a$
- $id^{+} : S \to [\![\mathcal{Q}_1]\!] \times S$, entonces $id_{\_}^{+-1} : [\![\mathcal{Q}_1]\!] \times S \to S$, definida como: $id^{+-1}(0, a) = a$.
- Para cualesquiera funciones $f, g \in S \to T$, $(f|g) \in S \to T$, el condicional se define como:

$$(f|g)\ (1, a) = (f\ a)$$

$$(f|g)\ (0, a) = (g\ a)$$

Y la inversa como: $(f|g)_{\_}^{-1} \in T \to S$

$$(f|g)^{-1}\ (f\ a) = (1, a)$$

$$(f|g)^{-1}\ (g\ a) = (0, a)$$

Observación. A todas las funciones inversas se les concatena el símbolo guión bajo _, esto con el propósito de que éste sea remplazado por el argumento que espera.

3. Se agregan las funciones inversas a la ejecución del programa separadas por el símbolo ;

$$[\![\bullet \otimes \bullet \vdash not : Q_2]\!] = \Big(const\ 0_{\_}^{-1}; const\ 0 \mid const\ 1_{\_}^{-1}; const\ 1\Big) \circ \Big(const\ 0_{\_}^{-1}; const\ 0 \times id_{\_}^{-1}; id\Big) \circ \Big(id_{\_}^{+-1}; id^{+}\Big)$$

obteniendo:

$$= \Big(const\ 0_{\_}^{-1} \mid const\ 1_{\_}^{-1}\Big); \Big(const\ 0_{\_}^{-1} \times id_{\_}^{-1}\Big); (id_{\_}^{+-1}); 1$$

Se ha obtenido un valor 1 y una pila de historial:
$\Big(const\ 0_{\_}^{-1} \mid const\ 1_{\_}^{-1}\Big); \Big(const\ 0_{\_}^{-1} \times id_{\_}^{-1}\Big); (id_{\_}^{+-1})$, de la cual, se espera poder ejecutar reversibilidad y a partir de la salida 1, regresar al valor inicial 0. Se obtiene:

$$= \Big(const\ 0_{\_}^{-1} \mid const\ 1_{\_}^{-1}\Big); \Big(const\ 0_{\_}^{-1} \times id_{\_}^{-1}\Big); (id^{+-1}(1))$$
$$= \Big(const\ 0_{\_}^{-1} \mid const\ 1_{\_}^{-1}\Big); \Big(const\ 0_{\_}^{-1} \times id_{\_}^{-1}\Big)(0, 1)$$
$$= \Big(const\ 0_{\_}^{-1} \mid const\ 1_{\_}^{-1}\Big); \Big(const\ 0^{-1}(0),\ id^{-1}(1)\Big)$$
$$= \Big(const\ 0_{\_}^{-1} \mid const\ 1_{\_}^{-1}\Big)(0, 1)$$

En este punto, el condicional no esperaba una tupla, sino un único valor, el cual dependiendo de la función ejecutada tomaría una decisión, sin embargo, la pila de historial no lleva ningún rastreo de lo que sucedio antes, aunado a que el orden del historial no resulta adecuado y la función inversa es inconsistente.

Lo anterior, permite detectar la complejidad de definir funciones inversas y el detectar cómo operarán para alcanzar la meta planteada.

Se comenzará trabajando el sublenguaje clásico de QML, y a partir de los resultados obtenidos, extenderlo a la totalidad del lenguaje, es decir, al lenguaje que considere datos y control cuántico.

4.1 Sublenguaje clásico

Este lenguaje considera sólo el subconjunto de términos que excluyen elementos y superposiciones cuánticas. Este sublenguaje de forma precisa tiene las siguientes restricciones: Descarta control cuántico, carece de las definiciones explícitas para la medición, la regla estructural de debilitamiento no está permitida en situaciones donde la información puede perderse, excepto para el término unitario () o tipo $\mathcal{Q}_1$, porque estos no acarrean información.

La sintaxis del sublenguaje clásico está en la Tabla 4.1, teniendo como referencia en la parte izquierda la totalidad de la sintaxis de QML y que la columna derecha es la de interés.

		Sintaxis de QML		**Sublenguaje de QML**
(Variables)	$x, y, \ldots$	$\in Vars$	$x, y, \ldots$	$\in Vars$
(Amplitudes de prob.)	$\kappa, \iota, \ldots$	$\in \mathbb{C}$	$\kappa, \iota, \ldots$	$\in \mathbb{C}$
(Patrones)	p, q	$::= x \mid (x, y)$	p, q	$::= x \mid (x, y)$
(Términos)	t, u	$::= x \mid x^{\vec{y}}$	t, u	$::= x$
		$\mid () \mid (t, u)$		$\mid () \mid (t, u)$
		$\mid$ **let** $p = t$ **in** u		$\mid$ **let** $p = t$ **in** u
		$\mid$ **if** t **then** u **else** u'		
		$\mid$ **if**$^\circ$ t **then** u **else** u'		$\mid$ **if**$^\circ$ t **then** u **else** u'
		$\mid$ **qfalse**$^{\vec{y}} \mid$ **qtrue**$^{\vec{y}} \mid$ **0**		$\mid$ **false** $\mid$ **true** $\mid \vec{0}$
		$\mid \kappa t \mid t + u$		

Tabla 4.1: Comparativa de sintaxis.

Los tipos son los mismos definidos previamente $[\![\mathcal{Q}_1]\!] = \{0\}$, $[\![\mathcal{Q}_2]\!] = \{0, 1\}$ y $[\![\sigma \otimes \tau]\!] = [\![\sigma]\!] \times [\![\tau]\!]$. Como resultado de la primer propuesta, se tiene que el nuevo modelo operacional permite generar una pila de historial de operaciones, la cual aplicará reversibilidad con reglas que posteriormente se definirán.

4.1.1 Semántica incorporando historial

El modelo para esta primer etapa y el sublenguaje clásico se visualiza en la Tabla 4.2, donde se puede identificar la nueva aparición del símbolo ; el cuál, se encargará de separar las funciones inversas de las funciones preestablecidas.

$[\![\bullet \vdash () : Q_1]\!]$	$= (const\ 0)^{-1}_{\_}; const\ 0$
$[\![\bullet \vdash false : Q_2]\!]$	$= (const\ 0)^{-1}_{\_}; const\ 0$
$[\![\bullet \vdash true : Q_2]\!]$	$= (const\ 1)^{-1}_{\_}; const\ 1$
$[\![x : \sigma \vdash x : \sigma]\!]$	$= id^{+}\_; id_{+}$
$[\![\Gamma \otimes \Delta \vdash \mathbf{let}\ x = t\ \mathbf{in}\ u : \sigma]\!]$	$= g \circ (f \times id^{-1}_{\_}; id) \circ ((\delta_{\Gamma,\Delta})^{-1}_{\_}; \delta_{\Gamma,\Delta})$
	donde: $f = [\![\Gamma \vdash t : \sigma]\!]$
	$g = [\![\Delta,\ x : \sigma \vdash u : \tau]\!]$
$[\![\Gamma \otimes \Delta \vdash (t,u) : \sigma \otimes \tau]\!]$	$= (f \times g) \circ ((\delta_{\Gamma,\Delta})^{-1}_{\_}; \delta_{\Gamma,\Delta})$
	donde: $f = [\![\Gamma \vdash t : \sigma]\!]$
	$g = [\![\Delta \vdash u : \tau]\!]$
$[\![\Gamma \otimes \Delta \vdash \mathbf{let}\ (x,y) = t\ \mathbf{in}\ u : \rho]\!]$	$= g \circ (f \times id^{-1}_{\_}; id) \circ ((\delta_{\Gamma,\Delta})^{-1}_{\_}; \delta_{\Gamma,\Delta})$
	donde: $f = [\![\Gamma \vdash t : \sigma \otimes \sigma]\!]$
	$g = [\![\Delta,\ x : \sigma,\ y : \tau \vdash u : \rho]\!]$
$[\![\Gamma \otimes \Delta \vdash \mathbf{if}^{\circ}\ c\ \mathbf{then}\ t\ \mathbf{else}\ u : \sigma]\!]$	$= (g \mid h) \circ (f \times id^{-1}_{\_}; id) \circ ((\delta_{\Gamma,\Delta})^{-1}_{\_}; \delta_{\Gamma,\Delta})$
	donde: $f = [\![\Gamma \vdash c : Q_2]\!]$
	$g = [\![\Delta \vdash t : \sigma]\!]$
	$h = [\![\Delta \vdash u : \sigma]\!]$
$[\![\Gamma \vdash t : \sigma]\!]$	$= f \times id_{*}\_; id^{*})$
	donde: $f = [\![\Gamma,\ x : Q_1 \vdash t : \sigma]\!]$

Tabla 4.2: Modelo operacional con historial para el sublenguaje clásico.

4.1.2 Funciones inversas auxiliares

Como parte de este ajuste, se incorporan las funciones inversas correspondientes a las descritas en la Sección 3.3.1. Esto con el propósito de aplicar reversibilidad. En este caso como fue únicamente para datos clásicos, la dificultad de definir las inversas fue menor respecto a las cuánticas. Entre las funciones con más labor se encuentra $(id^{*})^{-1}$ y el condicional $(f^{-1}|g^{-1})$.

Sea $S = \{0, 1\}$, donde $a, a', b \in S$

- $id^{-1} : S \to S$, por lo tanto, $id^{-1} = id$
- $(id^{+})^{-1} : S' \times S \to S$, donde $(id^{+})^{-1} = id_{*}$
  $(id^{+})^{-1}(a', a) = \begin{cases} (id^{+})^{-1}(a', a) = a, \text{ si a'=0} \\ (id^{+})^{-1}(a', a) = a', \text{ si a'=1} \end{cases}$
- $(id_{+})^{-1} : S \to Q_1 \times S$, donde $(id_{+})^{-1} = id^{+}$
- $(const\ 0)^{-1} : S \to Q_1$, tal que
  $(const\ 1)^{-1}(a) = \begin{cases} 1, \text{ si a=0} \\ 0, \text{ si a=1} \end{cases}$
- $\delta^{-1} : (S, S) \to S$, donde $\delta^{-1}(a, a) = a$.

- $swap^{-1} : T \times S \to S \times T$, donde $swap^{-1}(b, a) = (a.b)$.
- $(f \times g)^{-1} : (T_1 \times T_2) \to (S_1 \times S_2)$,
  donde $(f \times g)^{-1} = (f^{-1} \times g^{-1})$, por lo tanto: $(f \times g)^{-1}(a, b) = (f^{-1}\ a, g^{-1}\ b)$
- $(\delta_{\Gamma,\Delta})^{-1} : [\![\Gamma]\!] \times [\![\Delta]\!] \to [\![\Gamma \otimes \Delta]\!]$
  $$(\delta_{\Gamma,\Delta})^{-1} = \begin{cases} (\delta_{\Gamma',\Delta'})^{-1} \times \delta^{-1} & \text{si } \Gamma = \Gamma', x : \sigma \text{ y } \Delta = \Delta', x : \sigma \\ (\delta_{\Gamma',\Delta})^{-1} \times id & \text{si } \Gamma = \Gamma', x : \sigma \text{ y } x \notin dom(\Delta) \\ id_* & \text{si } \Gamma = \bullet \end{cases}$$
- Condicional:
  $(f^{-1}|g^{-1}) : T \to ([\![Q_2]\!] \times S)$, donde:
  $$(f^{-1}|g^{-1})(a) = \begin{cases} (a, f^{-1}\ a) & \text{si a=1} \\ (a, g^{-1}\ a) & \text{si a=0} \end{cases}$$

Hasta este punto, se tienen incorporados los ajusten indispensables para generar el historial del sublenguaje clásico de QML, los cuales fueron resultados preliminares. A partir de este modelo operacional, se podrá ejecutar reversibilidad.

4.1.3 Historal y Reversibilidad

Se reconsidera la idea de Van Tonder [58] para ajustar el lenguaje cuántico QML y agregar el historial de operaciones, similar a cómo él lo realiza en cálculo lambda cuántico (Sección A.1.2).

Pila de historial y factorización

Con los ajustes realizados al modelo operacional, éste lleva implicitamente las funciones inversas respectivas a cada programa, las inversas formarán una pila de historial que posibilitará reversibilidad computacional. Este proceso se formaliza a continuación.

Definición 4.1 (Factorización)**.** Sea una expresión generada por el modelo operacional con la forma:

$$\boldsymbol{h_{t_1}}-; t_1 \circ \boldsymbol{h_{t_2}}-; t_2 \circ \cdots \circ \boldsymbol{h_{t_n}}-; t_n$$

donde, el símbolo **;** denota composición de funciones, las expresiones h_{t_i} son historiales correspondientes a cada función t_i, respectivamente.

Los historiales h_{t_i} corresponderán a las funciones inversas t_i; lo que significa que en este punto están mezcladas ambas funciones, pero el interés es poder dividir la expresión en las h_{t_i} y t_i. Para esto se *factorizan* cómo:

$$\boldsymbol{h_{t_1}}-; t_1 \circ \boldsymbol{h_{t_2}}-; t_2 \circ \cdots \circ \boldsymbol{h_{t_n}}-; t_n = \boldsymbol{h_{t_n}}-\boldsymbol{; \cdots ;} \boldsymbol{h_{t_2}}-\boldsymbol{;} \boldsymbol{h_{t_1}}-; t_1 \circ t_2 \circ \cdots \circ t_n$$

Lo anterior produce el denominado estado computacional, que es el resultado de la pila de historial y las reglas de derivación que generaba el modelo operacional (antes de alguna modificación).

Definición 4.2. [Estado computacional] Un *estado computacional* es una secuencia de la forma:

$$\boldsymbol{h_{t_n}-;\cdots;h_{t_2}-;h_{t_1}-};t_1\circ t_2\circ\cdots\circ t_n$$

donde, $h_{t_n}-;\cdots;h_{t_2}-;h_{t_1}-$ es la *pila del historial* y $t_1\circ t_2;\circ\cdots\circ t_n$ es definido como *registro computacional*.

Recordando que de $h_{t_i}-;t_i$, la parte h_{t_i} es un historial que representa la operación inversa que ejecuta t_i.

Observación. La pila de historial (posiblemente vacía) $h_{t_n}-;\cdots;h_{t_2}-;h_{t_1}-$ se sintetiza y denota como $\mathcal{H}$.

Para obtener el estado computacional los términos se factorizan considerando varios casos, estos de acuerdo a los términos que forman el programa.

i) $\mathcal{H};\Big(\boldsymbol{h_t}-;t(a)\Big)$ factorizado como: $\mathcal{H};\boldsymbol{h_t}-;\Big(t(a)\Big)$

ii) $\mathcal{H};\Big(\boldsymbol{h_{t_1}}-;t_1\times\boldsymbol{h_{t_2}}-;t_2\Big)(a,b)$ factorizado como: $\mathcal{H};\boldsymbol{h_{t_1}}-\times\boldsymbol{h_{t_2}}-;\Big(t_1\times,\ t_2\Big)(a,b)$

iii) $\mathcal{H};\Big(\boldsymbol{h_{t_1}}-;t_1+\boldsymbol{h_{t_2}}-;t_2\Big)(a)$ factorizado como: $\mathcal{H};\boldsymbol{h_{t_1}}-+\boldsymbol{h_{t_2}}-;\Big(t_1(a)+t_2(a)\Big)$

iv) $\mathcal{H};\Big(\kappa(\boldsymbol{h_{t_1}}-;t_1)\Big)(a)$ factorizado como: $\mathcal{H};\boldsymbol{h_{t_1}}-;\Big(\kappa t_1(a)\Big)$, donde $\kappa\in\mathbb{C}$

v) Si t,t' son superposiciones o tuplas, entonces
$\Big(\boldsymbol{h_t}-;t|\boldsymbol{h_{t'}}-;t'\Big)(a,b)=\begin{cases}\mathcal{H};\boldsymbol{h_t}-;\Big(t(b)\Big) & \text{si } a=1\\ \mathcal{H};\boldsymbol{h_{t'}}-;\Big(t'(b)\Big) & \text{si } a=0\end{cases}$
sino
$\Big(\boldsymbol{h_t}-;t|\boldsymbol{h_{t'}}-;t'\Big)(a,b)=\mathcal{H};(\boldsymbol{h_t}-|\boldsymbol{h_{t'}}-);\ \Big((t|t')(a,b)\Big)$, y la definición de condicional es aplicada.

La factorización se comienza de derecha a izquierda, esto es:

$$\begin{aligned}\boldsymbol{h_{t_1}}-;t_1\circ\boldsymbol{h_{t_2}}-;t_2\circ\cdots\circ\boldsymbol{h_{t_n}}-;t_n&=\boldsymbol{h_{t_n}}-;\Big(\boldsymbol{h_{t_1}}-;t_1\circ\boldsymbol{h_{t_2}}-;t_2\circ\cdots\circ t_n\Big)\\&=\boldsymbol{h_{t_n}-;\cdots;h_{t_2}}-;\Big(\boldsymbol{h_{t_1}}-;t_1\circ t_2\circ\cdots\circ t_n\Big)\\&=\underbrace{\boldsymbol{h_{t_n}-;\cdots;h_{t_2}-;h_{t_1}}-}_{\text{Pila de historial}(\mathcal{H})};\underbrace{\Big(t_1\circ t_2\circ\cdots\circ t_n\Big)}_{\text{Registro computacional}}\end{aligned}$$

Si existe un condicional $(t|t')$, donde t y t' son superposiciones; su función inversa será factorizada y agregada a la pila del historial hasta que sea evaluado el condicional.

Reversibilidad

La reversibilidad se ejecuta partiendo de una pila de historial y un valor q, esto es: $\boldsymbol{h_{t_n-}; \cdots ; h_{t_2-}; h_{t_1-}}; \Big(t_1 \circ t_2 \circ \cdots \circ t_n\Big)(a) = \cdots = \boldsymbol{h_{t_n-}; \cdots ; h_{t_2-}}; h_{t_1-}; q$.

Si se observa la función inversa h_{t_i-} tiene escrito el símbolo guión bajo _, el cual se encarga de esperar el argumento con el cual trabajará la función respectiva h_{t_i}. El valor q ocupará el espacio del guión bajo del historial inmediato anterior de la pila respectiva, por ejemplo: $h_{t_i-}; q = h_{t_i}(q)$.

Dependiendo del programa desarrollado, el valor q puede tener diferentes formas y por lo tanto la reversibilidad se ejecutará con base a los siguientes casos:

i) $h_{t_i-}; q = h_{t_i}(q)$

ii) $(h_{t_i-} \times h_{t_j-})\ (q_1, q_2) = (h_{t_i}\ q_1, h_{t_j}\ q_2)$

iii) $(h_{t_i-} + h_{t_j-})\ \alpha(q_1 + q_2) = (h_{t_i}\ q_1, h_{t_j}\ q_2)$, donde $\alpha \in \mathbb{C}$ es descartada.

iv) $h_{t_1-}; \kappa q = h_{t_1}(q)$, donde $\kappa \in \mathbb{C}$ es descartada.

Teniendo presente qué es un registro computacional y las reglas paras pasar parámetros a funciones inversas, se procede a extrapolar estas ideas a la totalidad del lenguaje, es decir con datos y control cuántico.

4.2 Lenguaje para datos clásicos y cuánticos

El lenguaje QML permite generar programas con datos cuánticos y clásicos, éste recibe uno o más argumentos, el cual basado en reglas obtiene un valor final; el historial que se propone posibilitará regresar al argumento inicial de un programa, la idea primordial (cómo se mencionó en el marco metodológico) fue definir funciones inversas que ejecutan lo opuesto a lo que realizó un programa esperando regresar al valor inicial.

Este planteamiento resulto erróneo, dado que las funciones inversas devolvían valores no adecuados, para esto se tuvo que hacer un análisis de las funciones, hasta detectar el comportamiento que retornará el valor esperado.

Como se vio en la Sección 4.1, cuándo las funciones ejecutadas (de la semántica) se logran guardar, permitirán generar una pila de historial, para posteriormente aplicar reversibilidad explícita. Antes de abordar este tema para control cuántico, se dará una breve introducción de la importancia de reversibilidad cuántica.

Reversibilidad cuántica

Los cómputos cuanticos hacen uso de operaciones reversibles, dadas en términos de compuertas cuánticas correpondientes a matrices unitarias. Visualizando un algoritmo como un circuito cuántico, la reversibilidad es natural, por que ahí hay rastro de las operaciones ejecutadas, sin embargo, en un lenguaje de programación las operaciones no son almacenadas y la reversibilidad no es trivial. Por esto, varios autores han hecho contribuciones en esta área, sugiriendo almacenar las acciones ejecutadas [33, 42, 3, 10].

Por ejemplo, Van Tonder incorpora una pista de historial a cálculo lambda cuántico, ésta permitirá reversibilidad (sin mediciones intermedias) [58].

4.2.1 Modelo semántico con historial para datos clásicos y cuánticos

Considerando la totalidad de la sintaxis de QML (Sección 3.2), con elementos cuánticos como superposiciones y probabilidades asociadas a términos, dan como resultado el nuevo modelo semántico con historial para datos clásicos (Tabla 4.3) .

4.2.2 Funciones auxiliares

La semántica está definida en término de funciones, a partir de las cuales se incorporaron sus inversas $.^{-1}$ respectivas. Los autores usan Tripletas de Kleisli para pasar de datos clásicos a cuánticos, es decir, tales funciones operan con vectores complejos [5, 39, 8, 27].

Si bien, las funciones inversas parecen definirse naturalmente, esto no es trivial, debe considerarse que se establecen con la estrategia *llamada por nombre* (*call-by-name*), ya que los argumentos no pueden reducirse hasta que sean requeridos, así cómo la dificultad de trabajar tanto con datos clásicos y cuánticos, y controlar el término **if**$^\circ$.

Retomando, en un entorno cuántico los estados son vectores unitarios complejos y pueden estar en superposición (a través de Tripletas de Kleisli). Con esto, la sentencia $\Gamma \vdash t : \sigma$ es una función $[\![\Gamma]\!] \to [\![\sigma]\!]^Q$, donde $[\![\sigma]\!]^Q = [\![\sigma]\!] \to \mathbb{C}$ representa vectores complejos sobre el conjunto base $[\![\sigma]\!]$. La notación $[\![\sigma]\!]^Q$ se denota también como $V[\![\sigma]\!]$.

La implementación de esta definición, implica que cada función de tipo $S \to T$ cambia a la función de tipo $S \to V\ T$ usando *return*. Por ejemplo, considerar la función $const\ a : [\![\mathcal{Q}_1]\!] \to S$, entonces, extrapolando se obtiene: $return \circ const\ a : [\![\mathcal{Q}_1]\!] \to V\ S$. Esto se implementa para las funciones previamente definidas por el

$$
\begin{aligned}
[\![\bullet \vdash () : \mathcal{Q}_1]\!]^{\mathcal{Q}} &= (^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{0})^{-1}_{-};{}^{\mathcal{Q}}\, const\ 0 \\
[\![\bullet \vdash false : \mathcal{Q}_2]\!]^{\mathcal{Q}} &= (^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{0})^{-1}_{-};{}^{\mathcal{Q}}\, const\ 0 \\
[\![\bullet \vdash true : \mathcal{Q}_2]\!]^{\mathcal{Q}} &= (^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{1})^{-1}_{-};{}^{\mathcal{Q}}\, const\ 1 \\
[\![x : \sigma \vdash x : \sigma]\!]^{\mathcal{Q}} &= {}^{\mathcal{Q}}\boldsymbol{id}^{-1}_{+-};\ {}^{\mathcal{Q}}id_{+} \\
[\![\Gamma \otimes \Delta \vdash \mathbf{let}\ x = t\ \mathbf{in}\ u : \sigma]\!]^{\mathcal{Q}} &= g^{*} \circ (f \times {}^{\mathcal{Q}}\boldsymbol{id}^{-1}_{-};\ {}^{\mathcal{Q}}id)^{*} \circ ({}^{\mathcal{Q}}\delta^{-1}_{\Gamma,\Delta-};{}^{\mathcal{Q}}\,\delta_{\Gamma,\Delta}) \\
&\quad \text{dónde} \quad f = [\![\Gamma \vdash t : \sigma]\!]^{\mathcal{Q}} \\
&\qquad\quad\ g = [\![\Delta, x : \sigma \vdash u : \tau]\!]^{\mathcal{Q}} \\
[\![\Gamma \otimes \Delta \vdash (t, u) : \sigma \otimes \tau]\!]^{\mathcal{Q}} &= (f \times g)^{*} \circ ((^{\mathcal{Q}}\boldsymbol{\delta}_{\boldsymbol{\Gamma},\boldsymbol{\Delta}})^{-1}_{-};\ {}^{\mathcal{Q}}\delta_{\Gamma,\Delta}) \\
&\quad \text{dónde} \quad f = [\![\Gamma \vdash t : \sigma]\!]^{\mathcal{Q}} \\
&\qquad\quad\ g = [\![\Delta \vdash u : \tau]\!]^{\mathcal{Q}} \\
[\![\Gamma \otimes \Delta \vdash \mathbf{let}\ (x, y) = t\ \mathbf{in}\ u : \rho]\!]^{\mathcal{Q}} &= g^{*} \circ (f \times {}^{\mathcal{Q}}\boldsymbol{id}^{-1}_{-};\ {}^{\mathcal{Q}}id)^{*} \circ ({}^{\mathcal{Q}}\boldsymbol{\delta}^{-1}_{\boldsymbol{\Gamma},\boldsymbol{\Delta}-};\ {}^{\mathcal{Q}}\delta_{\Gamma,\Delta}) \\
&\quad \text{dónde} \quad f = [\![\Gamma \vdash t : \sigma \otimes \sigma]\!]^{\mathcal{Q}} \\
&\qquad\quad\ g = [\![\Delta, x : \sigma, y : \tau \vdash u : \rho]\!]^{\mathcal{Q}} \\
[\![\Gamma \otimes \Delta \vdash \mathbf{if}^{\circ}\ c\ \mathbf{then}\ t\ \mathbf{else}\ u : \sigma]\!]^{\mathcal{Q}} &= (g|h)^{*} \circ (f \times {}^{\mathcal{Q}}\boldsymbol{id}^{-1}_{-};\ {}^{\mathcal{Q}}id)^{*} \\
&\qquad\qquad\qquad \circ ((^{\mathcal{Q}}\boldsymbol{\delta}_{\boldsymbol{\Gamma},\boldsymbol{\Delta}})^{-1}_{-};\ {}^{\mathcal{Q}}\delta_{\Gamma,\Delta}) \\
&\quad \text{dónde} \quad f = [\![\Gamma \vdash c : \mathcal{Q}_2]\!]^{\mathcal{Q}} \\
&\qquad\quad\ g = [\![\Delta \vdash t : \sigma]\!]^{\mathcal{Q}} \\
&\qquad\quad\ h = [\![\Delta \vdash u : \sigma]\!]^{\mathcal{Q}} \\
[\![\Gamma \vdash t : \sigma]\!]^{\mathcal{Q}} &= (f \times {}^{\mathcal{Q}}\boldsymbol{id}^{+-1}_{-};{}^{\mathcal{Q}}id^{+}) \\
&\quad \text{dónde} \quad f = [\![\Gamma, x : \mathcal{Q}_1 \vdash t : \sigma]\!]^{\mathcal{Q}}
\end{aligned}
$$

Tabla 4.3: Semántica de QML.

autor (Sección 3.3.1). Por simplificación se usará la notación ${}^{\mathcal{Q}}const\ a = return \circ const\ a$.

Ahora, sea una función $f : S \to V\ T$, tal que $f(a) = v_b$, donde $v_b \in V\ T$ es un vector y $b \in T$; esto permitirá llevar los valores de salida a cualquier otra función auxiliar.

Para comprender este proceso, considere cualquier función auxiliar f donde clásicamente $f : [\![\sigma]\!] \to [\![\tau]\!]$, tal que $f(a) = b$. Entonces, la función ${}^{\mathcal{Q}}f$ tiene el tipo $[\![\sigma]\!] \to [\![\tau]\!]^{\mathcal{Q}}$, equivalente a ${}^{\mathcal{Q}}f : [\![\sigma]\!] \to [\![\tau]\!] \to \mathbb{C}$ (por interpretación de $[\![\tau]\!]^{\mathcal{Q}}$), se define cómo: $f(a)(b) = \alpha \in \mathbb{C}$, donde $\alpha = 1.0$ si $a = b$ y 0.0 en otro caso. Reconsiderar que toda función ${}^{\mathcal{Q}}f$ tiene el mecanismo de evaluación llamada por nombre, por lo que su argumento no se evaluará hasta que las funciones no se puedan derivar más.

Las siguientes funciones son parte de nuestra contribución. Esto es, para cada función auxiliar f, se define su inversa f^{-1}, el levantamiento f^* (asociada con Tripletas de Kleisli) y su inversa f^{*-1}.

- ${}^{\mathcal{Q}}id : S \to V\ S$, donde ${}^{\mathcal{Q}}id(a) = v_a$, $v_a \in S \to \mathbb{C}$, $v_a\ b = 1.0$ si $a = b$ ó 0.0, en otro caso.
  - ${}^{\mathcal{Q}}id^* : V\ S \to V\ S$, ${}^{\mathcal{Q}}id^* : V\ S \to S \to \mathbb{C}$. Definida como: ${}^{\mathcal{Q}}id^*(v_x)(b) = \sum_a (v_x\ a)({}^{\mathcal{Q}}id\ a\ b)$, donde $v_x \in V\ S$, $b \in S$ ó ${}^{\mathcal{Q}}id^*(v_x) = \sum_a (v_x\ a)({}^{\mathcal{Q}}id\ a)$.
  - $id^{-1} : V\ S \to S$, $id^{-1}(v_a) = a$
  - $id^{*-1} : (V\ S \to V\ S)$, $id^{*-1} = id^*$
- ${}^{\mathcal{Q}}id^+ : S \to V\ (\llbracket Q_1 \rrbracket \times S)$, ${}^{\mathcal{Q}}id^+(a) = v_{(0,a)}$. Sea $v_{(0,a)} \in V\ (\llbracket Q_1 \rrbracket \times S)$ entonces $v_{(0,a)}\ (b,c) = 1.0$, si $(0,a) = (b,c)$ ó 0.0, en otro caso. Extendido: ${}^{\mathcal{Q}}id^+ : S \to \llbracket Q_1 \rrbracket \times S \to \mathbb{C}$, ${}^{\mathcal{Q}}id^+(a)(0,c) = 1.0$, si $a = c$, ó 0.0 en otro caso.
  - ${}^{\mathcal{Q}}id^{+*} : V\ S \to V\ (\llbracket Q_1 \rrbracket \times S)$, esto es: ${}^{\mathcal{Q}}id^{+*} : V\ S \to (\llbracket Q_1 \rrbracket \times S) \to \mathbb{C}$, tal que ${}^{\mathcal{Q}}id^{+*}(v_x)(0,c) = \sum_a (v_x\ a)({}^{\mathcal{Q}}id^+(a)\ (0,c))$
  - $id^{+-1}\Big(v_{(b,a)}\Big) = \begin{cases} a, \text{ si } b{=}0 \\ b, \text{ si } b{=}1 \end{cases}$
  - $id^{+*-1}\Big(v_{(0,a)}\Big) = v_a$
- ${}^{\mathcal{Q}}id_+ : \llbracket Q_2 \rrbracket \times S \to V\ S$, definida como: ${}^{\mathcal{Q}}id_+(0,a) = v_a$ y ${}^{\mathcal{Q}}id_+(1,a) = v_1$, donde $v_a \in S \to \mathbb{C}$, $v_a\ b = 1.0$, si $a = b$, ó 0.0 en otro caso. Explícitamente: ${}^{\mathcal{Q}}id_+ : (\llbracket Q_1 \rrbracket \times S) \to S \to \mathbb{C}$, ${}^{\mathcal{Q}}id_+(0,a)(b) = 1.0$, si $a = b$, ó 0.0, otro caso.
  - ${}^{\mathcal{Q}}id_+^* : V\ (\llbracket Q_1 \rrbracket \times S) \to V\ S$, definido como: ${}^{\mathcal{Q}}id_+^*(v_{(x,y)})(b) = \sum_a \Big(v_{(x,y)}\ (0,a)\Big)\Big({}^{\mathcal{Q}}id_+\ (0,a)\ b\Big)$.
  - ${}^{\mathcal{Q}}id_+^{-1}(v_a) = (0,a)$
  - ${}^{\mathcal{Q}}id_+^{*-1}(v_a) = v_{(0,a)}$
- ${}^{\mathcal{Q}}const\ a : \llbracket Q_1 \rrbracket \to V\ S$, ${}^{\mathcal{Q}}const\ a(0) = v_a$. Si $v_a \in S \to \mathbb{C}$, $b \in S$, entonces $v_a\ (b) = \begin{cases} 1.0, \text{ si } b = a \\ 0.0, \text{ otro caso} \end{cases}$
  - ${}^{\mathcal{Q}}const\ a^* : V\ \llbracket Q_1 \rrbracket \to S \to \mathbb{C}$. Tal que: ${}^{\mathcal{Q}}const\ a^*(v_x)(b) = \sum_{a' \in \llbracket Q_1 \rrbracket} (v_x\ a')({}^{\mathcal{Q}}const\ a(a')(b))$.
  - ${}^{\mathcal{Q}}const\ 0^{-1}(v_a) = 0$ y ${}^{\mathcal{Q}}const\ 1^{-1}(v_a) = 1$ si $a = 0$ ó 0, si $a = 1$.

  - ${}^{\mathcal{Q}}const\ 0^{*-1}(v_a) = v_0$ y ${}^{\mathcal{Q}}const\ 1^{*-1}(v_a) = v_1$ si $a = 0$, ó v_0 si $a = 1$.

- ${}^{\mathcal{Q}}\delta : S \to V\ (S \times S)$, ${}^{\mathcal{Q}}\delta(a) = v_{(a,a)}$. Si $v_{(a,a)} \in (S \times S) \to \mathbb{C}$, entonces $v_{(a,a)}(b,c) = 1.0$, si $a = b = c$, 0.0 en otro caso. Explícitamente: ${}^{\mathcal{Q}}\delta : S \to (S \times S) \to \mathbb{C}$, ${}^{\mathcal{Q}}\delta(a)(b,c) = \begin{cases} 1.0, \text{ si } a = b = c \\ 0.0, \text{ otro caso.} \end{cases}$

  - ${}^{\mathcal{Q}}\delta^* : V\ S \to (S \times S) \to \mathbb{C}$, ${}^{\mathcal{Q}}\delta^*(v_x)(b,c) = \sum_a\ (v_x\ a)({}^{\mathcal{Q}}\delta\ a(b,c))$.
  - $\delta^{-1}\Big(v_{(a',a)}\Big) = a$.
  - $\delta^{*-1}\Big(v_{(a',a)}\Big) = (v_a)$.

- ${}^{\mathcal{Q}}swap\ :\ S \times T\ \to\ V\ (T \times S)$, ${}^{\mathcal{Q}}swap(a,b) = v_{(b,a)}$, donde, si $v_{(b,a)} \in V\ (T \times S)$, entonces $v_{(b,a)}(b',a') = 1.0$, si $(b,a) = (b',a')$, 0.0 en otro caso. Extendido: ${}^{\mathcal{Q}}swap\ :\ (S \times T)\ \to\ (T \times S)\ \to\ \mathbb{C}$, ${}^{\mathcal{Q}}swap\ (a,b)\ (b',a') = \begin{cases} 1.0, \text{ si } a = a', b = b' \\ 0.0, \text{ otro caso} \end{cases}$

  - ${}^{\mathcal{Q}}swap^*\ :\ V\ (S \times T)\ \to\ (T \times S)\ \to\ \mathbb{C}$, con ${}^{\mathcal{Q}}swap^*(v_{(x,y)})(b,a) = \sum_{a,b}(v_{(x,y)}\ (a,b))({}^{\mathcal{Q}}swap\ (a,b)\ (b,a))$.
  - $swap^{-1}\Big(v_{(b,a)}\Big) = (a,b)$.
  - $swap^{*-1}\Big(v_{(b,a)}\Big) = \Big(v_{(a,b)}\Big)$.

- $({}^{\mathcal{Q}}f \times^{\mathcal{Q}} g) : S_1 \times S_2 \to V\ (T_1 \times T_2)$, definida como $({}^{\mathcal{Q}}f \times^{\mathcal{Q}} g)(a,b) = v_{(fa,gb)}$. Explícitamente: $({}^{\mathcal{Q}}f \times {}^{\mathcal{Q}}g) : (S_1 \times S_2) \to (T_1 \times T_2) \to \mathbb{C}$, $({}^{\mathcal{Q}}f \times {}^{\mathcal{Q}}g)(a,b)(x,y) = ({}^{\mathcal{Q}}f\ a\ x)({}^{\mathcal{Q}}g\ b\ y)$.

  - $({}^{\mathcal{Q}}f \times {}^{\mathcal{Q}}g)^* : V\ (S_1 \times S_2) \to (T_1 \times T_2) \to \mathbb{C}$, donde $({}^{\mathcal{Q}}f \times {}^{\mathcal{Q}}g)^*(v_{(a',b')})\ (x,y) = \sum_{a,b}\Big(v_{(a',b')}\ (a,b)\Big)\Big(({}^{\mathcal{Q}}f\ \times^{\mathcal{Q}}\ g)\ (a,b)\ (x,y)\Big)$, ó $({}^{\mathcal{Q}}f\ \times^{\mathcal{Q}}\ g)^*\ (v_{(a,b)}) = \sum_{a,b}\Big(v_{(a',b')}\ (a,b)\Big)\Big(({}^{\mathcal{Q}}f \times^{\mathcal{Q}} g)(a,b)\Big)$.
  - $({}^{\mathcal{Q}}f^{-1} \times^{\mathcal{Q}} g^{-1})\ \Big(v_{(a,b)}\Big) = (f^{-1}\ a, g^{-1}\ b)$.
  - $({}^{\mathcal{Q}}f^{-1} \times^{\mathcal{Q}} g^{-1})^*\ \Big(v_{(a,b)}\Big) = v_{(f^{-1}a, g^{-1}b)}$.

- $({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g) : ([\![Q_2]\!] \times S) \to V\ T$, es asociada con el condicional **if**°, definida como: $({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g)(1,a) = ({}^{\mathcal{Q}}fa)$ and $({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g)(0,a) = ({}^{\mathcal{Q}}ga)$. Expandida: $({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g) : ([\![Q_2]\!] \times S) \to T \to \mathbb{C}$, $({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g)\ (1,a)\ (b) = \begin{cases} 1.0, \text{ si } ({}^{\mathcal{Q}}f\ a) = b \\ 0.0, \text{ otro caso} \end{cases}$ y $({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g)\ (0,a)\ (b) = \begin{cases} 1.0, \text{ si } ({}^{\mathcal{Q}}g\ a) = b \\ 0.0, \text{ otro caso} \end{cases}$

- $({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g)^* : V\ (\llbracket Q_2 \rrbracket \times S) \to T \to \mathbb{C}$, con $({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g)^* \left(v_{(x',a')}(y)\right) = \sum_{x,a} \left(v_{(x',a')}(x,a)\right)\left(({}^{\mathcal{Q}}f|{}^{\mathcal{Q}}g)(x,a)\right)$.
- $({}^{\mathcal{Q}}f^{-1}|{}^{\mathcal{Q}}g^{-1}) : V\ T \to (\llbracket Q_2 \rrbracket \times S)$, definida como: $(f^{-1}|g^{-1})(v_a) = (a, f^{-1}\ a)$ si $a = 1$, $(a, g^{-1}\ a)$ si $a = 0$.
- $({}^{\mathcal{Q}}f^{-1}|{}^{\mathcal{Q}}g^{-1})^* : V\ T \to V\ (\llbracket Q_2 \rrbracket \times S)$, donde, $({}^{\mathcal{Q}}f^{*-1}|{}^{\mathcal{Q}}g^{*-1})\ (v_a) = (v_{(a,f^{-1}a)}$ si $a = 0$, ó $v_{(a,g^{-1}a)}$ si $a = 1$.

- ${}^{\mathcal{Q}}\delta_{\Gamma,\Delta}$, separa los contextos de tipo, basado en las siguientes condiciones. ${}^{\mathcal{Q}}\delta_{\Gamma,\Delta} : \llbracket \Gamma \otimes \Delta \rrbracket \to V\ (\llbracket \Gamma \rrbracket \times \llbracket \Delta \rrbracket)$ definido como: ${}^{\mathcal{Q}}\delta_{\Gamma,\Delta}(a \otimes a') = v_{(a,a')}$. Si uno de los contextos es vacío $\bullet$, entonces se aplica: ${}^{\mathcal{Q}}\delta_{\Gamma,\Delta}(a) = v_a$. ${}^{\mathcal{Q}}\delta^*_{\Gamma,\Delta}(v_{(a\otimes a')} = v_{(a',a')}$ ó ${}^{\mathcal{Q}}\delta^*_{\Gamma,\Delta}(v_a) = v_a$, las funciones inversas son obtenidas similarmente a los casos previos. ${}^{\mathcal{Q}}\delta_{\Gamma,\Delta} = \begin{cases} {}^{\mathcal{Q}}\delta_{\Gamma',\Delta'} \times^{\mathcal{Q}} \delta & \text{si } \Gamma = \Gamma', x : \sigma \\ & \text{y } \Delta = \Delta', x : \sigma \\ {}^{\mathcal{Q}}\delta_{\Gamma',\Delta} \times^{\mathcal{Q}} id & \text{si } \Gamma = \Gamma', x : \sigma \\ & \text{y } x \notin dom(\Delta). \\ {}^{\mathcal{Q}}id^+ & \text{si } \Gamma = \bullet \end{cases}$

Con las reglas y funciones anteriores, se pueden derivar ciertos programas, sin embargo, aún se requiere anexar las reglas modificadas para términos con amplitudes de probabilidad y superposiciones. Las reglas para formar términos bien tipados y para derivar datos (cuánticos y clásicos) se encuentran en la Tabla 4.4 y la Tabla 4.5, respectivamente. En base a estas definiciones, se procede a describir la estrategia para generar una pila de historial.

$$\frac{}{\bullet \vdash \vec{0} : \sigma}\text{z-intro} \qquad \frac{\Gamma \vdash^\circ c : Q_2 \quad \Delta \vdash^\circ t, u : \sigma}{\Gamma \otimes \Delta \vdash^\circ \textbf{if}^\circ\ c\ \textbf{then}\ t\ \textbf{else}\ u : \sigma}\textbf{if}^\circ$$

$$\frac{\Gamma \vdash t : \sigma}{\Gamma \vdash \kappa t : \sigma}\text{prob} \qquad \frac{\Gamma \vdash^\circ t, u : \sigma \quad t \perp u \quad |\lambda|^2 + |\kappa|^2 = 1}{\Gamma \vdash^\circ \lambda t + \kappa u : \sigma}\text{sup}^\circ$$

$$\frac{\Gamma \vdash t, u : \sigma}{\Gamma \vdash t + u : \sigma}\text{sup} \qquad \frac{\Gamma \vdash^\circ t : \sigma \quad \Gamma \vdash t \equiv u : \sigma}{\Gamma \vdash^\circ u : \sigma}\text{subst}$$

Tabla 4.4: Reglas para términos cuánticos bien formados (II) [5].

$[\![\bullet \vdash \vec{0} : \sigma]\!]^{\mathcal{Q}} = \boldsymbol{^{\mathcal{Q}}const\ a_{-}^{-1}}; {}^{\mathcal{Q}}const\ a$	donde $\forall a \in [\![\sigma]\!].v_a = 0.0$
$[\![\Gamma \vdash \kappa t : \sigma]\!]^{\mathcal{Q}} = (\kappa)f$	donde $\forall b \in [\![\Gamma]\!]$ $f = [\![\Gamma \vdash t : \sigma]\!]^{\mathcal{Q}}$
$[\![\Gamma \vdash t + u : \sigma]\!]^{\mathcal{Q}} = \frac{1}{\sqrt{2}}\big(f + g\big)$	donde $\forall a \in [\![\Gamma]\!]$ $f = [\![\Gamma \vdash t : \sigma]\!]^{\mathcal{Q}}$ $g = [\![\Gamma \vdash u : \sigma]\!]^{\mathcal{Q}}$

Tabla 4.5: Reglas para datos cuánticos.

4.2.3 Semántica para control cuántico e historial

Se incorporan tres reglas que consideran: el vector cero, un término con cierta amplitud de probabilidad y superposición.

Si tenemos datos y control cuánticos, considere los programas que contienen **if**$^{\circ}$ ó $t + u$, sus términos que los conforman deben ser ortogonales entre sí. Esto se denota en las derivaciones con el símbolo $\vdash^{\circ}$, la ortogonalidad con $\perp$ y el producto interno con $\langle \mid \rangle$. Para satisfacer la ortogonalidad, se puede ver la Tabla 3.4, y para ver los términos cuánticos bien formados, ver la Tabla 4.4.

Teniendo lo anterior, ahora se puede aplicar el modelo operativo con un programa, por ejemplo: $[\![\bullet \otimes \bullet \vdash \textbf{if}^{\circ}\ true\ \textbf{then}\ (-1)true + false\ \textbf{else}\ true + false : Q_2]\!]^{\mathcal{Q}} = (g|h)^* \circ (f \times \boldsymbol{^{\mathcal{Q}}id_{-}^{-1}}; {}^{\mathcal{Q}}\, id)^* \circ (\boldsymbol{^{\mathcal{Q}}\delta_{\Gamma,\Delta}{}_{-}^{-1}}; {}^{\mathcal{Q}}\, \delta_{\Gamma,\Delta})$. En este punto, las funciones f se mezclan con sus inversas f^{-1}, lo cual, para nuestros propósitos estas deben estar separadas. El cómo lograr esto, se encuentra en la sección 4.3.

4.3 Historial y Reversibilidad

Para aplicar la reversibilidad, el proceso es análogo con datos clásicos. Teniendo en cuenta que cada derivación se da a partir de funciones, y éstas se almacenan en una pila, entonces, las funciones inversas darán como resultado la reversibilidad, permitiendo retornar al argumento inicial del programa.

Retomando la definición de estado computacional (Definición 4.2), en $\boldsymbol{h_{t_i}}_{-}; t_i$, la parte $\boldsymbol{h_{t_i}}$ representa y almacena la operación inversa de la función t_i. Y que $\mathcal{H}$ (Observación 4.1.3), denota la pista de historial (posiblemente vacía).

A partir de la semántica propuesta, se obtiene una expresión, y para que ésta tenga la forma de un estado computacional, los términos deben ser *factorizados*, esto es, separar y organizar las funciones inversas de las directas.

4.3.1 Factorización

Sea la expresión $\boldsymbol{h_{t_1-}}; t_1 \circ \boldsymbol{h_{t_2-}}; t_2 \circ \cdots \circ \boldsymbol{h_{t_n-}}; t_n$, donde el punto y coma (**;**) denota la concatenación de cadenas. Las expresiones h_{t_i} son historiales correspondientes a cada función inversa t_i, respectivamente. Los términos t_i están asociados con funciones del modelo operativo, por lo tanto, recibirán argumentos. Una función inversa se denota cómo ${}^{\mathcal{Q}}h_{t_i-}$ y la función cuántica cómo ${}^{\mathcal{Q}}t_i$.

Las expresiones $(h_{t_i-}; t_i)$ pueden ser para los casos clásico y cuántico, y tener diferentes formas dependiendo del programa, para lo cual, considere los siguientes casos:

1) $\mathcal{H}; \Big(\boldsymbol{h_{t-}}; t\Big)(a)$ factorizado como: $\mathcal{H}; \boldsymbol{h_{t-}}; \Big(t(a)\Big)$

2) $\mathcal{H}; \Big(\boldsymbol{h_{t_1-}}; t_1 \times \boldsymbol{h_{t_2-}}; t_2\Big)(a,b)$ factorizado como: $\mathcal{H}; \boldsymbol{h_{t_1-}} \times \boldsymbol{h_{t_2-}}; \Big(t_1 \times\ t_2\Big)(a,b)$

3) $\mathcal{H}; \Big(\boldsymbol{h_{t_1-}}; t_1 + \boldsymbol{h_{t_2-}}; t_2\Big)(a)$ factorizado como: $\mathcal{H}; \boldsymbol{h_{t_1-}} + \boldsymbol{h_{t_2-}}; \Big(t_1(a) + t_2(a)\Big)$

4) $\mathcal{H}; \Big(\kappa(\boldsymbol{h_{t_1-}}; t_1)\Big)(a)$ factorizado como: $\mathcal{H}; \boldsymbol{h_{t_1-}}; (\kappa t_1(a))$, donde $\kappa \in \mathbb{C}$

5) Si t, t' son superposiciones o túplas, entonces

$$\mathcal{H}; \Big(\boldsymbol{h_{t-}}; t|\boldsymbol{h_{t'-}}; t'\Big)(a,b) = \begin{cases} \mathcal{H}; \boldsymbol{h_{t-}}; t(b), \text{si } a = 1 \\ \mathcal{H}; \boldsymbol{h_{t'-}}; t'(b), \text{si } a = 0 \end{cases}$$

o también

$\mathcal{H}; \Big(\boldsymbol{h_{t-}}; t|\boldsymbol{h_{t'-}}; t'\Big)(a,b) = \mathcal{H}; (\boldsymbol{h_{t-}}|\boldsymbol{h_{t'-}}); \Big((t|t')(a,b)\Big)$, y la definición del condicional es aplicado.

Cuando la expresión tiene la forma $(h_{t_i-}; t_i)^*$, tener en cuenta los siguientes casos:

a) $\mathcal{H}; \Big(\boldsymbol{h_{t-}}; t\Big)^*(v_a)$ factorizado como: $\mathcal{H}; \boldsymbol{h_{t-}}^*; \Big(t(v_a)\Big)^*$

b) $\mathcal{H}; \Big(\boldsymbol{h_{t_1-}}; t_1 \times \boldsymbol{h_{t_2-}}; t_2\Big)^*(v_{(a,b)})$ factorizado como: $\mathcal{H}; (\boldsymbol{h_{t_1-}} \times \boldsymbol{h_{t_2-}})^*; \Big(t_1 \times t_2\Big)^*(v_{(a,b)})$

c) $\mathcal{H}; \Big(\boldsymbol{h_{t_1-}}; t_1 + \boldsymbol{h_{t_2-}}; t_2\Big)^*(v_a)$ factorizado como: $\mathcal{H}; \boldsymbol{h^*_{t_1-}} + \boldsymbol{h^*_{t_2-}}; \Big(t_1 + t_2\Big)^*(v_a)$

d) $\mathcal{H}; \alpha(\boldsymbol{h_{t_1-}}; t_1)^*(v_a)$ factorizado como: $\mathcal{H}; \boldsymbol{h_{t_1-}}^*; \alpha t_1^*(v_a)$, donde $\alpha \in \mathbb{C}$

e) Para el condicional $(f|g)^*$: Si t, t' son superposiciones o túplas, implica:

$$(\boldsymbol{h_{t-}}; t|\boldsymbol{h_{t'-}}; t')^*(v_{(a,b)}) = \begin{cases} \mathcal{H}; \boldsymbol{h_{t-}}^*; t(b), \text{ si } a = 1 \\ \mathcal{H}; \boldsymbol{h_{t'-}}^*; t'(b), \text{si } a = 0 \end{cases}$$

Para cualquier término con o sin levantamiento ($.^*$), la factorización también se inicia de derecha a izquierda:

$$\begin{aligned}\boldsymbol{h_{t_1}-}; t_1 \circ \boldsymbol{h_{t_2}-}; t_2 \circ \cdots \circ \boldsymbol{h_{t_n}-}; t_n =& \boldsymbol{h_{t_n}-}; \left(\boldsymbol{h_{t_1}-}; t_1 \circ \boldsymbol{h_{t_2}-}; t_2 \circ \cdots \circ t_n\right) \\ =& \boldsymbol{h_{t_n}-; \cdots ; h_{t_2}-}; \left(\boldsymbol{h_{t_1}-}; t_1 \circ t_2 \circ \cdots \circ t_n\right) \\ =& \underbrace{\boldsymbol{h_{t_n}-; \cdots ; h_{t_2}-; h_{t_1}-}}_{\text{Pila de historial}(\mathcal{H})}; \underbrace{\left(t_1 \circ t_2 \circ \cdots \circ t_n\right)}_{\text{Registro computacional}}\end{aligned}$$

Si existe un condicional $(t|t')$, donde t y t', son superposiciones; sus funciones inversas correspondientes incorporarán a la pista una vez que se haya evaluado el condicional. Con lo anterior tenemos el estado computacional, y continuamos con las reglas propuestas para la reversibilidad.

4.3.2 Reversibilidad

La reversibilidad como en la parte clásica, funciona a partir de un estado computacional (resultado de derivaciones) con la forma: $\boldsymbol{h_{t_1}-; \cdots ; h_{t_2}-; h_{t_1}-}; v_x$, donde v_x es un término irreducible o vector (para el modelo).

Dado un historial h_{t_i-}, se aplica como argumento v al historial inmediato anterior, reemplazandolo por el símbolo $\_$, y aplicando la función inversa respectiva, esto es, $h_{t_i-}; v = h_{t_i}(v)$. La reversibilidad se puede aplicar a partir de un valor v, desde la pila de historial $(h_{t_1-}; h_{t_2-}; \cdots ; h_{t_n-})$, considerando los siguientes casos:

1. $h_{t_i-}; v = h_{t_i}(v)$.
2. $(h_{t_i-} \times h_{t_j-})\ (v_1, v_2) = (h_{t_i}\ v_1, h_{t_j}\ v_2)$
3. $({}^{\varrho}h_{t_i-} \times^{\varrho} h_{t_j-})^*\ v_{(x,y)} = v_{(h_{t_i(x)}, h_{t_j(y)})}$
4. $(h_{t_i-} + h_{t_j-})\ \alpha(v_1 + v_2) = (h_{t_i}\ v_1, h_{t_j}\ v_2)$, donde $\alpha \in \mathbb{C}$.
5. $({}^{\varrho}h_{i-} +^{\varrho} h_{j-})^*\ \alpha(v_x + v_y) = v_{({}^{\varrho}h_{i(v_x)},\ {}^{\varrho}h_{j(v_y)})}$, donde $\alpha \in \mathbb{C}$.
6. $h_{t_1-}; (\alpha v) = h_{t_1}(v)$

donde $\alpha \in \mathbb{C}$ se descartan.

Para englobar y aterrizar toda la información previa, se mencionará un ejemplo de un programa cuántico, este es la compuerta de Hadamard.

Ejemplo: Compuerta de Hadamard

$had\ true = \mathbf{if}^{\circ}\ true\ \mathbf{then}\ (-1)true + false\ \mathbf{else}\ true + false.$

$$
\begin{aligned}
[\![\bullet \otimes \bullet \vdash had\ true : Q_2]\!]^{Q} =& (g|h)^* \circ (f \times \boldsymbol{\mathcal{Q}id_{-}^{-1}};^{\mathcal{Q}} id)^* \circ ((\boldsymbol{\mathcal{Q}\delta_{\Gamma,\Delta}})_{-}^{-1};^{\mathcal{Q}} \delta_{\Gamma,\Delta}) \\
=& \Big(\overbrace{\frac{1}{\sqrt{2}}\Big(-1\boldsymbol{\mathcal{Q}const\ 1_{-}^{-1}};^{\mathcal{Q}} const\ 1 + \boldsymbol{\mathcal{Q}const\ 0_{-}^{-1}};^{\mathcal{Q}} const\ 0\Big)}^{g} \Big)^* \\
& | \overbrace{\frac{1}{\sqrt{2}}\Big(\boldsymbol{\mathcal{Q}const\ 1_{-}^{-1}};^{\mathcal{Q}} const\ 1 + \boldsymbol{\mathcal{Q}const\ 0_{-}^{-1}};^{\mathcal{Q}} const\ 0\Big)^*}^{h} \Big) \\
& \circ \Big(\boldsymbol{\mathcal{Q}const\ 1_{-}^{-1}};^{\mathcal{Q}} const\ 1 \times \boldsymbol{id_{-}^{-1}}; id \Big)^* \circ \Big(\boldsymbol{\mathcal{Q}id_{-}^{+-1}};^{\mathcal{Q}} id^{+} \Big) \\
=& \overbrace{\boldsymbol{(\mathcal{Q}id_{-}^{+-1}); (\mathcal{Q}const\ 1_{-}^{-1} \times^{\mathcal{Q}} id_{-}^{-1})^*}}^{\mathcal{H}}; \\
& \Big(g|h\Big)^* \circ (^{\mathcal{Q}}const\ 1 \times^{\mathcal{Q}} id)^* \circ {}^{\mathcal{Q}}id^{+}(0) \\
=& \mathcal{H}; \Big(g|h\Big)^* \circ (^{\mathcal{Q}}const\ 1 \times^{\mathcal{Q}} id)^*\ (v_{(0,0)}) \\
=& \mathcal{H}; \Big(g|h\Big)^* \circ \Big(v_{(0,0)}\ (0,0)(^{\mathcal{Q}}const\ 1 \times^{\mathcal{Q}} id)\ (0,0)\Big) \\
=& \mathcal{H}; \Big(g|h\Big)^*\ (v_{(1,0)}) = \mathcal{H}; \Big(v_{(1,0)}\ (1,0)(g|h)\ (1,0)\Big) = \mathcal{H}; g(0) \\
=& \mathcal{H}; \Big(\frac{1}{\sqrt{2}}\Big((-1)\boldsymbol{\mathcal{Q}const\ 1_{-}^{*-1}};^{\mathcal{Q}} const\ 1 + \\
& \boldsymbol{\mathcal{Q}const\ 0_{-}^{*-1}};^{\mathcal{Q}} const\ 0\Big)\Big)(0) \\
=& \mathcal{H}; (\boldsymbol{\mathcal{Q}const\ 1_{-}^{*-1}} + \boldsymbol{\mathcal{Q}const\ 0_{-}^{*-1}}); \\
& \Big(\frac{1}{\sqrt{2}}\Big((-1)^{\mathcal{Q}}const\ 1 + {}^{\mathcal{Q}}const\ 0\Big)\Big)(0) \\
=& \mathcal{H}; \frac{1}{\sqrt{2}}\Big((-1)^{\mathcal{Q}}const\ 1\ (0) + {}^{\mathcal{Q}}const\ 0\ (0)\Big) \\
=& H; \frac{1}{\sqrt{2}}\Big((-1)v_1 + v_0\Big)
\end{aligned}
$$

Si evaluamos las funciones obtenemos: $\frac{1}{\sqrt{2}}\Big((-1)(0.0) + (1.0)\Big)$. Explícitamente $\mathcal{H}$ es: $\boldsymbol{(\mathcal{Q}id_{-}^{+-1}); (\mathcal{Q}const\ 1_{-}^{-1} \times^{\mathcal{Q}} id_{-}^{-1})^*; (\mathcal{Q}const\ 1_{-}^{*-1} + ^{\mathcal{Q}}const\ 0_{-}^{*-1})}$;.

Aplicando reversibilidad se tiene:

$$
\begin{aligned}
&(^{\mathcal{Q}}\boldsymbol{id}_{-}^{+-1}); (^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{1}_{-}^{-1} \times {}^{\mathcal{Q}}\boldsymbol{id}_{-}^{-1})^{*};\\
&\quad (^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{1}_{-}^{*-1} + {}^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{0}_{-}^{*-1}); \frac{1}{\sqrt{2}}\Big((-1)v_1 + v_0\Big)\\
&= (^{\mathcal{Q}}\boldsymbol{id}_{-}^{+-1}); (^{\mathcal{Q}}\boldsymbol{const}\mathbf{1}_{-}^{-1} \times {}^{\mathcal{Q}}\boldsymbol{id}_{-}^{-1})^{*}; (^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{1}_{-}^{*-1}\\
&\quad +^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{0}_{-}^{*-1})\ \ (v_1 + v_0)\\
&= (^{\mathcal{Q}}\boldsymbol{id}_{-}^{+-1}); (^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{1}_{-}^{-1} \times {}^{\mathcal{Q}}\boldsymbol{id}_{-}^{-1})^{*}\\
&\quad v_{(^{\mathcal{Q}}const\ 1^{-1}(v_1), ^{\mathcal{Q}}const\ 0^{-1}(v_0))}\\
&= (^{\mathcal{Q}}\boldsymbol{id}_{-}^{+-1}); (^{\mathcal{Q}}\boldsymbol{const}\ \mathbf{1}_{-}^{-1} \times {}^{\mathcal{Q}}\boldsymbol{id}_{-}^{-1})^{*}\ v_{(0,0)}\\
&= (^{\mathcal{Q}}\boldsymbol{id}_{-}^{+-1}); v_{(const\ 1^{-1}\ 0,\ id_{-}^{-1}\ 0)}\\
&= {}^{\mathcal{Q}}id^{+-1}\ v_{(1,0)} = 1
\end{aligned}
$$

A través de este ejemplo, se observa cómo se ejecuta la reversibilidad y la aplicación de las reglas definidas. A continuación, se procede con la segunda parte de esta investigación.

Most of the fundamental ideas of science are essentially simple, and may, as a rule, be expressed in a language comprehensible to everyone.

— Albert Einstein

5

Lenguaje QML & Lambda cuántico-$\mathcal{S}$

Mediciones cuánticas

En esta sección se encontrará la integración del Lenguaje de Programación Cuántico QML y Cálculo Lambda Cuántico (Lambda-$\mathcal{S}$) [18], obteniendo un lenguaje que retoma la sintaxis de QML e implementa las reglas estándar de Lambda-$\mathcal{S}$, funciona con datos y control cuántico, tiene un modelo operacional e incorpora mediciones cuánticas.

5.1 Sintaxis

La sintaxis incorpora términos lineales $!t$, superposición de estados $\sum_i t_i$ y la regla de medición $\pi_j t$. Esto se observa en la Tabla 5.1.

(Variables)	$x, y, \ldots$	$\in Vars$
(Amplitudes Pro.)	$\kappa, \iota, \ldots$	$\in \mathbb{C}$
(Patrones)	p, q	$::= x \mid !x \mid (x, y)$
(Términos)	t, u	$::= x \mid (t, u) \mid \lvert \mathbf{false} \mid \mathbf{true} \mid \kappa t \mid \vec{0} \mid \sum_i t_i \mid$ $\mathbf{let}\ p = t\ \mathbf{in}\ u \mid \mathbf{if}^\circ\ t\ \mathbf{then}\ u\ \mathbf{else}\ u' \mid \pi_j t$

Tabla 5.1: Sintaxis QML y Cálculo Lambda-$\mathcal{S}$.

El tipado de los términos de esta sintaxis está dado por la siguiente gramática: $\sigma = \mathcal{Q} \mid \sigma \otimes \sigma \mid !\sigma$. Y los contextos de tipo (Γ, Δ) se denotan como $\Gamma = \bullet \lvert \Gamma, x : \sigma$.

Los términos ya tipados, se deberán distinguir si son lineales sobre la superposición o no, o si son tipos base. Los tipos base, serán interpretados con vectores unitarios mientras los tipos superpuestos se interpretan como espacios vectoriales; resaltando que los únicos tipos que se pueden clonar son los tipo base.

- El tipo base se distingue como $!\sigma$ y se puede copiar .
- Un término denotado por $!t$ es lineal en superposición, está asociado con estados cuánticos (no duplicables), más adelante se encuentra el tipado de éstas.
- $\Gamma \vdash^{\circ} t : \sigma$, garantiza $||t|| = 1$, al cual llamaremos *estricto*.

Definición 5.1. Un término se denota si es estricto o no de la siguiente manera: $\Gamma \vdash^{a} t : \sigma$, donde $a \in \{-, \circ\}$; $\circ$ denota un término estricto.

Si $a, b \in \{-, \circ\}$, entonces $a \sqcap b = \begin{cases} \circ, \text{ si } a = \circ = b \\ -, \text{ otro caso} \end{cases}$

Respecto a tipos se define lo siguiente:

Notación 5.1. Considerar lo siguiente: $\mathcal{Q}^n = \mathcal{Q} \otimes \mathcal{Q} \otimes \cdots \otimes \mathcal{Q}$ (n-times).

Notación 5.2. Sea $[!]\mathcal{Q}$, que es interpretado como: $\mathcal{Q}$ o $!\mathcal{Q}$.

Notación 5.3. $!(\mathcal{Q} \otimes \mathcal{Q}) = !\mathcal{Q} \otimes !\mathcal{Q}$

Definición 5.2. En tipos, la expresión $|\ |$ se define inductivamente cómo: $|\mathcal{Q}| = \mathcal{Q}$, $|\sigma \otimes \sigma| = |\sigma| \otimes |\sigma|$ ó $|!\sigma| = |\sigma|$.

El propósito de esta definición es que para un término con un tipo de la forma $!\sigma$, se puede descartar $!$ y ver como un tipo no duplicable σ. Esto permitirá garantizar en las reglas de términos bien formados la no clonación cuántica.

Definición 5.3. Considerar $|\tau|_{\sigma} = \begin{cases} \tau, \text{ si } \sigma = !\sigma \\ |\tau|, \text{ otro caso} \end{cases}$

Definición 5.4. Considerar $u \perp_{a \sqcap b} u' = \begin{cases} u \perp u', \text{ si } a \sqcap b = \circ \\ \textbf{false}, \text{ otro caso} \end{cases}$

Para conocer si dos términos son ortogonales entre sí, se tiene la Tabla 5.2. Recordando que dos términos t y u son ortogonales $t \perp u$ si su producto interno es 0, es decir, $\langle t\,|u\rangle = 0$; y que $\overline{\lambda}$ es el transpuesto conjugado de λ.

Con las consideraciones anteriores, se procede a establecer las reglas propuestas para términos bien tipados y el modelo operacional.

$\langle t\|t\rangle = 1$ si $\Gamma \vdash t : !\sigma$	$\langle \lambda t + \lambda' t'\|u\rangle = \overline{\lambda}\langle t\|u\rangle + \overline{\lambda'}\langle t'\|u\rangle$
$\langle !\mathbf{false}\|!\mathbf{true}\rangle = 0$	$\langle t\|\kappa u + \kappa' u'\rangle = \kappa\langle t\|u\rangle + \kappa'\langle t\|u'\rangle$
$\langle !\mathbf{true}\|!\mathbf{false}\rangle = 0$	$\langle \lambda t\|u\rangle = \overline{\lambda}\langle t\|u\rangle$
	$\langle t\|\lambda u\rangle = \lambda\langle t\|u\rangle$
$\langle t + t'\|u\rangle = \langle t\|u\rangle + \langle t'\|u\rangle$	$\langle t\|u + u'\rangle = \langle t\|u\rangle + \langle t\|u'\rangle$
$\langle (t,t')\|(u,u')\rangle = \langle t\|u\rangle\langle t'\|u'\rangle$	$\langle t\|u\rangle =?$ otherwise

Tabla 5.2: Producto interno y ortogonalidad.

5.2 Reglas para términos bien formados

Las reglas de la Tabla 5.3, determinan cómo tipar o formar programas de manera correcta.

5.3 Semántica operacional

Las reglas de la semántica operacional se encuentran en la Tabla 5.4. Estableciendo que opera con la estrategia *llamada por base* (*call-by-base*), por ejemplo, la regla β_b opera sólo cuando el término es de la base computacional (**false** o **true**), en caso contrario, se va reduciendo tal término hasta que sea de la base y aplicar la sustitución. Para lograr reducir términos, se tiene la tabla de equivalencias (Tabla 5.5), la cual se explicará adelante.

Equivalencias

Las equivalencias entre términos están para que un término pueda representarse de otra manera o con otro término, permitiendo reducirse. Tales reglas en superposiciones aplican linealidad y se verifica que en ambos elementos de la equivalencia el tipo se preserve, excepto para $t + 0r \not\equiv t$, donde se hizo una exclusión de la equivalencia.

Observación. Considere la siguiente expresión $t + 0r \not\equiv t$. En este caso, $t + 0r$ no se reescribe y no es equivalente a t, porque si asumimos que t y r son funciones, entonces el dominio de $t + 0r$ es la intersección de los dominios de t y $0r$, mientras que t solo tiene su propio dominio; por lo tanto, los dominios de ambas partes pueden ser diferentes y en consecuencia, esta equivalencia no está garantizada.

$$\frac{\Gamma \vdash^{\circ} t : \sigma}{\Gamma \vdash t : \sigma}\ \boldsymbol{D_{\circ}} \qquad \frac{\Gamma \vdash^{a} t : !\sigma}{\Gamma \vdash^{a} t : \sigma}\ \boldsymbol{D_{!}} \qquad \frac{}{!x : !\mathcal{Q} \vdash^{\circ} x : !\mathcal{Q}}\ \boldsymbol{!var}$$

$$\frac{}{x : |\sigma| \vdash^{a} x : |\sigma|}\ \boldsymbol{var} \qquad \frac{\Gamma \vdash^{a} t : \sigma \quad \Delta \vdash^{b} u : \tau}{\Gamma \otimes \Delta \vdash^{a \sqcap b} (t, u) : \sigma \otimes \tau}\ \otimes\boldsymbol{intro} \qquad \frac{}{\bullet \vdash \overrightarrow{0} : |\sigma|}\ \boldsymbol{z\text{-}intro}$$

$$\frac{}{\bullet \vdash^{\circ} \mathbf{false} : !\mathcal{Q}}\ \boldsymbol{f\text{-}intro} \qquad \frac{}{\bullet \vdash^{\circ} \mathbf{true} : !\mathcal{Q}}\ \boldsymbol{t\text{-}intro} \qquad \frac{\Gamma \vdash^{a} t : \sigma \quad |\alpha| = 1}{\Gamma \vdash^{a} \alpha t : \sigma}\ \boldsymbol{fase}$$

$$\frac{\Gamma \vdash^{a} t : \sigma \quad |\alpha| \neq 1}{\Gamma \vdash \alpha t : |\sigma|}\ \boldsymbol{prob} \qquad \frac{\Gamma \vdash \beta t : \sigma \quad \Gamma \vdash^{a} t : \sigma' \quad |\alpha\beta| = 1}{\Gamma \vdash^{a} \alpha(\beta t) : |\sigma|}\ \boldsymbol{prob_1}$$

$$\frac{\Gamma \vdash^{a} t_i : \sigma \quad \forall i}{\Gamma \vdash \sum_{i=1}^{n} t_i : \sigma}\ \boldsymbol{sup} \qquad \frac{\Gamma \vdash^{a} t_i : !\sigma \quad \forall i, j, t_i \perp_a t_j \quad \sum_{i=1}^{n} |\alpha| = 1}{\Gamma \vdash^{a} \alpha\left(\sum_{i=1}^{n} t_i\right) : |\sigma|}\ \boldsymbol{sup_{\alpha}}$$

$$\frac{\Gamma \vdash^{\circ} t_i : !\sigma \quad \forall i, j, t_i \perp_a t_j \quad \sum_{i=1}^{n} |\alpha_i| = 1, n \geq 1}{\Gamma \vdash^{a} \sum_{i=1}^{n} \alpha_i t_i : |\sigma|}\ \boldsymbol{sup^{\circ}}$$

$$\frac{\Gamma \vdash^{a} t : \sigma \quad \Delta, x : \sigma \vdash^{b} u : \tau}{\Gamma \otimes \Delta \vdash^{a \sqcap b} \mathbf{let}\ x = t\ \mathbf{in}\ u : \tau}\ \boldsymbol{let} \qquad \frac{\Gamma \vdash^{a} t : \sigma \quad \Delta, !x : !\sigma \vdash^{b} u : \tau}{\Gamma \otimes \Delta \vdash^{a \sqcap b} \mathbf{let}\ !x = t\ \mathbf{in}\ u : |\tau|_{\sigma}}\ \boldsymbol{!let}$$

$$\frac{\Gamma \vdash [\alpha_i]\mathbf{let}\ !x = [\beta_i]t_i\ \mathbf{in}\ u : |\tau| \quad \Gamma' \vdash^{a} \mathbf{let}\ !x = t_i\ \mathbf{in}\ u : \tau \quad \forall i \quad \sum_{i=1}^{n} |[\alpha_i][\beta_i]| = 1}{\Gamma \vdash^{a} \sum_{i=1}^{n} [\alpha_i]\mathbf{let}\ !x = [\beta_i]t_i\ \mathbf{in}\ u : |\tau|}\ \boldsymbol{let_{\alpha}}$$

$$\frac{\Gamma \vdash^{a} t : \sigma \otimes \tau \quad \Delta, !x : !\sigma, !y : !\tau \vdash^{b} e : \rho}{\Gamma \otimes \Delta \vdash^{a \sqcap b} \mathbf{let}\ (!x, !y) = t\ \mathbf{in}\ e : ||\rho|_{\tau}|_{\sigma}}\ \otimes\boldsymbol{\text{-}!elim}$$

$$\frac{\Gamma \vdash^{a} t : [!]\mathcal{Q} \quad \Delta \vdash^{b} u, u' : \sigma \quad u \perp_{a \sqcap b} u'}{\Gamma \otimes \Delta \vdash^{a \sqcap b} \mathbf{if}^{\circ}\ t\ \mathbf{then}\ u\ \mathbf{else}\ u' : |\sigma|_{[!]\mathcal{Q}}}\ \boldsymbol{if^{\circ}}$$

$$\frac{\Gamma \vdash [\alpha_i]\mathbf{if}^{\circ}\ [\beta_i]t_i\ \mathbf{then}\ u\ \mathbf{else}\ u' : |\sigma| \quad \Gamma \vdash^{a} \mathbf{if}^{\circ}\ t_i\ \mathbf{then}\ u\ \mathbf{else}\ u' : \sigma \quad \forall i \quad \sum_{i=1}^{n} |[\alpha_i][\beta_i]| = 1}{\Gamma \vdash^{a} \sum_{i=1}^{n} [\alpha_i]\mathbf{if}^{\circ}\ [\beta_i]t_i\ \mathbf{then}\ u\ \mathbf{else}\ u' : |\sigma|}\ \boldsymbol{if^{\circ}_{\alpha}}$$

$$\frac{\Gamma \vdash^{a} t : \mathcal{Q}^n \quad n \geq j}{\Gamma \vdash^{a} \pi_j t : (!\mathcal{Q})^j \otimes \mathcal{Q}^{n-j}}\ \boldsymbol{M^{\circ}}$$

Tabla 5.3: Reglas para términos bien tipados.

Por ejemplo, la siguiente equivalencia garantiza que ambos representan el mismo programa, asegurando que sean términos bien formados, estrictos y con el mismo tipo. Esto se logro con ayuda de las reglas de la Tabla 5.3, que permite definir que las equivalencias están bien formadas.

$$\vdash^{\circ} \mathbf{let}\ !x = \left(\frac{1}{\sqrt{2}}\mathbf{true} + \frac{1}{\sqrt{2}}\mathbf{false}\right)\ \mathbf{in}\ x : Q$$

$$\begin{array}{rcll}
\textbf{let } x = t \textbf{ in } u & \longrightarrow & u[t/x] & \beta_n \\
\textbf{let } !x = t \textbf{ in } u & \longrightarrow & u\ [t/!x] \text{ if } t = \begin{cases} \textbf{false} \\ \textbf{true} \end{cases} & \beta_b \\
\textbf{let } (!x, !y) = (t, u) \ \textbf{ in } e & \longrightarrow & \textbf{let } !x = t \textbf{ in let } !y = u \textbf{ in } e & \text{let}_{\otimes}
\end{array}$$

$$\begin{array}{rcll}
\textbf{if}^{\circ} \textbf{ false then } u \textbf{ else } u' & \longrightarrow & u' & \text{if}^{\circ}_{\text{f}} \\
\textbf{if}^{\circ} \textbf{ true then } u \textbf{ else } u' & \longrightarrow & u & \text{if}^{\circ}_{\text{t}}
\end{array}$$

$$\begin{array}{rcll}
\pi_j \left(\sum_{i=1}^{n} \alpha_i \prod_{h=1}^{m} b_{hi} \right) & \underset{p_b}{\longrightarrow} & b \otimes \phi_b & \text{M}^{\circ} \\
& & \phi_b = \sum_{i \in T_b} \left(\frac{\alpha_i}{\sqrt{\sum_{r \in T_b} |\alpha_r|^2}} \right) \prod_{h=j+1}^{m} b_{hi} & \\
& & p_b = \sum_{i \in T_b} \left(\frac{|\alpha_i|^2}{\sum_{r=1}^{n} |\alpha_r|^2} \right) & \\
& & T_b = \{ i \leq n | b_{1i} \otimes \cdots \otimes b_{ji} = b \} & \\
& & \forall i,\ b_{hi} = true \ \text{ ó } \ b_{hi} = false &
\end{array}$$

Reglas contextuales

$$\frac{\alpha t \quad t \longrightarrow t'}{\alpha t'} \qquad \frac{tu \quad t \longrightarrow t'}{t'u} \qquad \frac{t + u \quad t \longrightarrow t'}{t' + u}$$

$$\frac{\pi_j\ t \quad t \longrightarrow t'}{\pi_j\ t'} \qquad \frac{(t, u) \quad t \longrightarrow t'}{(t', u)} \qquad \frac{(t, u) \quad t \not\longrightarrow t' \quad u \longrightarrow u'}{(t, u')}$$

Tabla 5.4: Modelo operacional.

Siendo equivalente a:

$$2\textbf{let } !x = \left(\frac{1}{2\sqrt{2}} \textbf{true} \right) \textbf{ in } x + 2\textbf{let } !x = \left(\frac{1}{2\sqrt{2}} \textbf{false} \right) \textbf{ in } x$$

En equivalencias que involucran superposiciones, estas son lineales dependiendo del término formado, por ejemplo considerar $(\sum_i \alpha_i t_i, r) \equiv \sum_i \alpha_i (t_i, r)$, formando:

$$\left(\frac{1}{\sqrt{2}} \textbf{false} + \frac{1}{\sqrt{2}} \textbf{true},\ \textbf{true} \right) \equiv \frac{1}{\sqrt{2}} \Big(\textbf{false}, \textbf{true} \Big) + \frac{1}{\sqrt{2}} \Big(\textbf{true}, \textbf{true} \Big)$$

5.4 Regla de medición cuántica

Considerando que el axioma de medición calcula con qué probabilidad un estado puede colapsar a otro, con la regla de medición $\pi_j\ t$, se modela el cómo medir al

$1t$	$\equiv$	t
$\alpha t + \alpha u$	$\equiv$	$\alpha(t+u)$
$\alpha t + \beta t$	$\equiv$	$(\alpha+\beta)t$
$\alpha(\beta t)$	$\equiv$	$(\alpha\beta)t$
$t+u$	$\equiv$	$u+t$
If $FV(t)=\emptyset,\ 0t$	$\equiv$	$\vec{0}$
$(\sum_i \alpha_i t_i, r)$	$\equiv$	$\sum_i \alpha_i (t_i, r)$
$(r, \sum_i \alpha_i t_i)$	$\equiv$	$\sum_i \alpha_i (r, t_i)$
$\textbf{let } !x = \sum_{i=1}^{n} \alpha_i t_i \textbf{ in } u$	$\equiv$	$\sum_{i=1}^{n} \left(\alpha_i \textbf{ let } !x = t_i \textbf{ in } u \right)$
$\textbf{let } (!x, !y) = \sum_{i=1}^{n} \alpha_i t_i \textbf{ in } u$	$\equiv$	$\sum_{i=1}^{n} \left(\alpha_i \textbf{ let } (!x, !y) = t_i \textbf{ in } u \right)$
$\textbf{if}^\circ \sum_{i=1}^{n} \alpha_i t_i \textbf{ then } u \textbf{ else } u'$	$\equiv$	$\sum_{i=1}^{n} \alpha_i \textbf{ if}^\circ\ t_i \textbf{ then } u \textbf{ else } u'$

Tabla 5.5: Equivalencias

estado t hasta la j-ésima posición de éste; basta con determinar hasta que índice se desea (j) medir y se consideran todas las posibles salidas, con qué probabilidad puede suceder cada evento y se normaliza el estado resultante.

Para comprender esto, se describe el funcionamiento de la regla con ayuda de un ejemplo.

Suponer el término (estado) $q =$

$$\frac{1}{\sqrt{2}}(false, true, false) + \frac{1}{\sqrt{4}}(false, true, true) + \frac{1}{\sqrt{4}}(true, false, false),$$

y se desea medir el primer estado, es decir, cuándo $j = 1$. Entonces se considera las posibilidades de colapso para la posición $j = 1$.

Para identificar los índices $j = 1$ dentro de cada subtérmino, se recurre a la notación $\sum_{i=1}^{n} \alpha_i \prod_{h=1}^{m} b_{hi}$, donde para todo i, α_i son las amplitudes de cada término, mientras $\prod_{h=1}^{m} b_{hi}$, hace referencia al producto tensorial (asociativo) de los tipos base (**true**,**false**), de ahí que b_{hi} sea **false** o **true**.

Para asimilar mejor la expresión $\sum_{i=1}^{n} \alpha_i \prod_{h=1}^{m} b_{hi}$, la cual formará un estado cuántico, suponer $\sum_{i=1}^{3} \alpha_i \prod_{h=1}^{3} b_{hi} = \frac{1}{\sqrt{2}}(\textbf{false}, \textbf{true}, \textbf{false}) + \frac{1}{\sqrt{4}}(\textbf{false}, \textbf{true}, \textbf{true}) + \frac{1}{\sqrt{4}}(\textbf{true}, \textbf{false}, \textbf{false})$. Ahora, para fines de establecer la regla de medición se requiere de esta notación para identificar los elementos a los cuales se puede colapsar

y con qué probabilidad. Para eso se denotan los subíndices i y h. Al formar el término no se escriben los subíndices, sin embargo, para fines de esta explicación, se abusará la notación explícitamente como:

$$\sum_{i=1}^{3}\alpha_i\prod_{h=1}^{3}b_{hi} = \frac{1}{\sqrt{2}}(\mathbf{false}_{11},\mathbf{true}_{21},\mathbf{false}_{31}) + \frac{1}{\sqrt{4}}(\mathbf{false}_{12},\mathbf{true}_{22},\mathbf{true}_{32}) + \frac{1}{\sqrt{4}}(\mathbf{true}_{13},\mathbf{false}_{23},\mathbf{false}_{33}).$$

Entonces, al aplicar π_1 q, se observa que las diferentes salidas se reducen a **true**, **false** o el tensor de estos, tal que los posibles eventos los tendrá el conjunto T_b.

Con cualquiera de las posibles salidas, se concatena a ésta (con $\otimes$) el resto del o los estados que coinciden con alguno de los eventos posibles; visualmente se observa como:

$$\underbrace{\tfrac{1}{\sqrt{2}}(\mathbf{false}_{11},\mathbf{true}_{21},\mathbf{false}_{31})}_{p_{b1}} + \tfrac{1}{\sqrt{4}}(\mathbf{false}_{12},\mathbf{true}_{22},\mathbf{true}_{32}) + \underbrace{\tfrac{1}{\sqrt{4}}(\mathbf{true}_{13},\mathbf{false}_{23},\mathbf{false}_{33})}_{p_{b2}}$$

$$\Big(\underbrace{\mathbf{false}}_{b},\underbrace{\big(\beta_{h'i}(\mathbf{true},\mathbf{false})+\beta_{h''i}(\mathbf{true},\mathbf{true})\big)}_{\phi_b}\Big) \qquad \Big(\underbrace{\mathbf{true}}_{b},\underbrace{1(\mathbf{false},\mathbf{false})}_{\phi_b}\Big)$$

Considerando lo anterior, se retoma la regla y define que:

$$\pi_j\left(\sum_{i=1}^{n}\alpha_i\prod_{h=1}^{m}b_{hi}\right) \underset{p_b}{\longrightarrow} b\otimes\phi_b$$

$$\phi_b = \sum_{i\in T_b}\left(\frac{\alpha_i}{\sqrt{\sum_{r\in T_b}|\alpha_r|^2}}\right)\prod_{h=j+1}^{m}b_{hi}$$

$$p_b = \sum_{i\in T_b}\left(\frac{|\alpha_i|^2}{\sum_{r=1}^{n}|\alpha_r|^2}\right)$$

$$T_b = \{i\le n | b_{1i}\otimes\cdots\otimes b_{ji} = b\}$$

$$\forall i,\ b_{hi} = true \ \text{ ó } \ b_{hi} = false$$

- b, corresponde a alguno de los posibles eventos de colapso.
- ϕ_b, es el resto normalizado del o los estados que coinciden con b.
- T_b, va contener las posibles salidas al medir, hasta la j-ésima posición.

Enseguida, se define un ejemplo completo, asumiendo un valor específico de colapso para exhibir mejor el funcionamiento de esta regla.

5.4.1 Ejemplo de medición cuántica

Sea el término:

$$q = \frac{1}{\sqrt{8}}(\mathbf{false},\mathbf{false},\mathbf{false}) + \frac{1}{\sqrt{8}}(\mathbf{false},\mathbf{false},\mathbf{true}) + \frac{1}{\sqrt{8}}(\mathbf{false},\mathbf{true},\mathbf{false}) + \frac{1}{\sqrt{8}}(\mathbf{false},\mathbf{true},\mathbf{true}) + \frac{1}{\sqrt{8}}(\mathbf{true},\mathbf{false},\mathbf{false}) + \frac{1}{\sqrt{8}}(\mathbf{true},\mathbf{false},\mathbf{true}) + \frac{1}{\sqrt{8}}(\mathbf{true},\mathbf{true},\mathbf{false}) + \frac{1}{\sqrt{8}}(\mathbf{true},\mathbf{true},\mathbf{true})$$

Para describir cómo sucedería en un caso particular, suponer que b=(**false**,**true**).

Observación. Tener presente que las mediciones son probabilísticas y no se puede conocer a que subestado va colapsar un término al medirlo.

Aplicando la regla $\pi_2\ q$, se obtiene que:

- $T_b = \{b_{1i} \otimes b_{2i} | i \leq 8\}$. Si $b = (\mathbf{false},\mathbf{true})$, entonces se reduce a considerar los términos $T_b = \{(\mathbf{false}^{13},\mathbf{true}^{23}),(\mathbf{false}^{14},\mathbf{true}^{24})\}$, con $i = 3$ e $i = 4$.
- $p_b = \frac{|\frac{1}{\sqrt{8}}|^2}{1} + \frac{|\frac{1}{\sqrt{8}}|^2}{1} = \frac{1}{4}$.
- $\sqrt{\sum_{r\in T_b} |\alpha_r|^2} = \sqrt{|\frac{1}{\sqrt{8}}|^2 + |\frac{1}{\sqrt{8}}|^2} = \frac{1}{2}$
- ϕ_b se obtiene a partir de:

$$\phi_b = \sum_{i\in T_b} \left(\frac{\alpha_i}{\sqrt{\sum_{r\in T_b} |\alpha_r|^2}} \right) \prod_{h=2+1}^{3} b_{hi} = \frac{\frac{1}{\sqrt{8}}}{\frac{1}{2}} b_{33} + \frac{\frac{1}{\sqrt{8}}}{\frac{1}{2}} b_{34}$$
$$= \frac{2}{\sqrt{8}}\mathbf{false} + \frac{2}{\sqrt{8}}\mathbf{true}$$

En conclusión, si b fuera (**false**,**true**) la medición reduciría a:

$$\pi_2\ q \underset{\frac{1}{4}}{\longrightarrow} \left((\mathbf{false},\mathbf{true}), (\frac{2}{\sqrt{8}}\mathbf{false} + \frac{2}{\sqrt{8}}\mathbf{true}) \right)$$

Con este ejemplo, se concluye las contribuciones y modificaciones del lenguaje QML para agregar un comportamiento más intuitivo en conjunto con cálculo Lambda-$\mathcal{S}$.

A continuación se anexan las demostraciones que garantizan el funcionamiento adecuado de la propuesta.

5.5 Subject reduction

La *reducción de un término*, también llamada preservación de tipo o que convencionalmente se conoce como *subject reduction*, es una propiedad que establece que el tipo de un término se preserva bajo la reducción, es decir, si se deriva un término $\Gamma \vdash t : \sigma$ y t reduce a otro término t', entonces $\Gamma \vdash t' : \sigma$. En esta sección se encontrarán las demostraciones correspondientes.

Previo a definir las demostraciones de subjet reduction, se requieren los lemas de generación, los cuales establecen que las sentencias pueden derivarse únicamente con el uso de las reglas que este lema establece; éste aborda la forma de los términos. Posterior a los lemas de generación, se realizan las pruebas para el *lema de substitución* y que por convención se llaman *substitution lemma*, de términos no lineales y lineales.

Lema 5.1 (Lemas de generación). *1. Si* $\Gamma \vdash t : \sigma$ *entonces* $\Gamma \vdash^{\circ} t : \sigma$.

2. Si $\Gamma \vdash^{a} t : \sigma$ *entonces* $\Gamma \vdash^{a} t : !\sigma$, *donde* $a = \circ$ *ó* $a = -$.

3. Si $\Gamma \vdash^{a} x : \sigma$, *entonces:*

(a) $\Gamma = !x : !Q$, $\sigma = !\mathcal{Q}$ *y* $a = \circ$.

(b) $\Gamma = x : |\sigma|$ *y* $\sigma = |\sigma|$.

4. Si $\Gamma \vdash^{a} (t_1, t_2) : \sigma$, *entonces para* $i = 1, 2$, $\Delta_i \vdash^{b_i} t_i : \tau_i$, *donde* $\Delta_1 \otimes \Delta_2 = \Gamma$, $\tau_1 \otimes \tau_2 = \sigma$ *y también* $b_1 \sqcap b_2 = a$, *or* $a = -$.

5. If $\Gamma \vdash^{a} \vec{0} : \sigma$, *then* $\sigma = |\sigma|$, $a = -$ *and* $\Gamma = \bullet$.

6. Si $\Gamma \vdash^{a} \mathbf{false} : \sigma$, *entonces* $\sigma : [!]\mathcal{Q}$, *y* $\Gamma = \bullet$.

7. Si $\Gamma \vdash^{a} \mathbf{true} : \sigma$, *entonces* $\sigma : [!]\mathcal{Q}$ *y* $\Gamma = \bullet$.

8. Si $\Gamma \vdash^{a} \alpha t : \sigma$, *entonces:*

(a) Para $|\alpha| = 1$, $\Gamma \vdash^{a} t : \sigma$.

(b) Para $|\alpha| \neq 1$, $\sigma = |\sigma|$, $\Gamma \vdash^{a'} t : \sigma$ *y* $a = -$.

(c) Para $t = (\beta t')$, $\sigma = |\sigma|$, *donde* $|\alpha\beta| = 1$, $\Gamma \vdash^{a} t' : \sigma'$, $\Gamma \vdash \beta t' : \sigma$.

(d) Para todo i, $t = \sum_{i=1}^{n} t_i$, $\sigma = |\sigma|$, *donde* $\sum_{i=1}^{n} |\alpha| = 1$, $\Gamma \vdash^{a} t_i : !\sigma$, *y para todo* i, j, $t_i \perp_a t_j$.

9. Si $\Gamma \vdash^{a} \mathbf{let}\ x = t\ \mathbf{in}\ u : \sigma$, *entonces* $\Delta_1 \vdash^{b} t : \tau$ *y* $\Delta_2, x : \tau \vdash^{c} u : \sigma$, *donde* $\Delta_1 \otimes \Delta_2 = \Gamma$ *y también* $b \sqcap c = a$, *o* $a = -$.

10. *Si* $\Gamma \vdash^a \mathbf{let}\ !x = t\ \mathbf{in}\ u : \sigma$, *entonces* $\sigma = |\sigma'|_\tau$, $\Delta_1 \vdash^b t : \tau$, *y* $\Delta_2, !x : !\tau \vdash^c u : \sigma'$, *donde* $\Delta_1 \otimes \Delta_2 = \Gamma$, *y también* $b \sqcap c = a$, *o* $a = -$.

11. *Si* $\Gamma \vdash^a \mathbf{let}\ (!x, !y) = t\ \mathbf{in}\ u : \sigma$, *entonces* $\sigma = ||\rho|_\tau|_\varepsilon$, $\Delta_1 \vdash^b t : \varepsilon \otimes \tau$, *and* $\Delta_2,\ !x : !\varepsilon,\ !y : !\tau\ \vdash^c u : \rho$, $\Gamma = \Delta_1 \otimes \Delta_2$, *y también* $b \sqcap c = a$, *o* $a = -$.

12. *Si* $\Gamma \vdash^a \mathbf{if}^\circ\ t\ \mathbf{then}\ u\ \mathbf{else}\ u' : \sigma$, *entonces* $\Delta_1 \vdash^b t : [!]\mathcal{Q}$, $\Delta_2 \vdash^c u, u' : \sigma'$, $\sigma = |\sigma'|_{[!]\mathcal{Q}}$, *y* $u \perp_{b \sqcap c} u'$, *donde* $\Delta_1 \otimes \Delta_2 = \Gamma$, *y también* $b \sqcap c = a$, *o* $a = -$.

13. *Si* $\Gamma \vdash^a \sum_{i=1}^{n} t_i : \sigma$, *entonces:*

    (a) *Para todo* i, $\Gamma \vdash^a t_i : \sigma$, $a = -$.

    (b) *Para todo* i, $t_i \equiv \alpha_i u_i$, *donde* $\sigma = |\sigma|$, $\Gamma \vdash^\circ u_i : !\sigma$, $\sum_{i=1}^{n} |\alpha_i|^2 = 1$, *y para todo* i, j, $u_i \perp u_j$.

    (c) *Para todo* i, $t_i = \sum_{i=1}^{n} [\alpha_i]\mathbf{let}\ !x = [\beta_i]t_i'\ \mathbf{in}\ u$, $\sigma = |\tau|$, $\Gamma \vdash [\alpha_i]\mathbf{let}\ !x = [\beta_i]t_i'\ \mathbf{in}\ u : |\tau|$, $\Gamma \vdash^a \mathbf{let}\ !x = t_i'\ \mathbf{in}\ u : \tau$, $\forall i, \sum_{i=1}^{n} |[\alpha_i][\beta_i]| = 1$.

    (d) *Para todo* i, $t_i = [\alpha_i]\mathbf{if}^\circ\ [\beta_i]t_i'\ \mathbf{then}\ u\ \mathbf{else}\ u'$, $\sigma = |\sigma|$,
    $\Gamma \vdash [\alpha_i]\mathbf{if}^\circ\ [\beta_i]t_i'\ \mathbf{then}\ u\ \mathbf{else}\ u' : |\sigma|$, $\Gamma \vdash^a \mathbf{if}^\circ\ t_i'\ \mathbf{then}\ u\ \mathbf{else}\ u' : \sigma$, $\forall i, \sum_{i=1}^{n} |[\alpha_i][\beta_i]| = 1$.

    Notación 5.4. La expresión $[\alpha_i]expr1[\beta]expr2$, donde para todo i, $\alpha_i, \beta_i \in \mathbb{C}$, corresponde a que puede o no estar α_i y/o β_i multiplicando a las expresiones subsecuentes, esto es:

    $$[\alpha_i]expr1[\beta]expr2 = \begin{cases} [\alpha_i]expr1expr2 \\ expr1[\beta]expr2 \\ [\alpha_i]expr1[\beta]expr2 \end{cases}$$

    Considerando que si aparece algún valor α_i y/o β_i, entonces, siempre deberá aparecer donde se haga referencia a éste. Las dos regla previas (d) y (c), hacen referencia a que debe aparecer al menos una de $[\alpha]$ o $[\beta]$ o ambas.

14. *Si* $\Gamma \vdash^a \pi_j t : \sigma$, *entonces* $\Gamma \vdash^a t : \mathcal{Q}^n$, $n \geq j$, $\sigma = (!\mathcal{Q})^j \otimes \mathcal{Q}^{n-j}$, *donde si* $n = j$ *entonces* $\sigma = \mathcal{Q}^n$.

Demostración. Se puede observar que las reglas ⊗***intro***, ***z-intro***, ***f-intro***, ***t-intro***, ***let***, ***!let***, ⊗***-!elim***, ***if***$^\circ$y $\boldsymbol{M}^\circ$ están controladas directamente por la sintaxis. Del resto de la reglas, $\boldsymbol{D}_\circ$ y $\boldsymbol{D}_!$ permite eliminar lo estricto ($\vdash^\circ$) del secuente y el ! del tipo, respectivamente. Con respecto a las reglas ***fase***, ***prob*** y $\boldsymbol{prob}_1$, encauzan a términos estrictos o no, de acuerdo a ciertas probabilidades de amplitud. Con

respeto a las reglas ***sup***, ***sup***$_\alpha$, ***sup***$^\circ$, son superposiciones de elementos con o sin amplitudes asociadas. Para las reglas ***let***$_\alpha$ y ***if***$^\circ{}_\alpha$, establecen términos dónde la linealidad es aplicada, existiendo una única regla para cada una.

Lema 5.2 (Substitución de términos no lineales)**.** *Si $u \neq \pi_j r$, se tiene $\Gamma, !x : !\sigma \vdash^a u : \tau$ y $\Delta \vdash^b t : !\sigma$, entonces $\Gamma \otimes \Delta \vdash^{a \sqcap b} u[t/x] : \tau$. También, $\Gamma, !x : !\sigma \vdash^a \pi_j r : \tau$ y $\Delta \vdash^\circ t : !\sigma$, entonces $\Gamma \otimes \Delta \vdash^a \pi_j r[t/x] : \tau$.*

Demostración. Por inducción sobre u.

- Sea $u = x$. Entonces, se tiene $\Gamma, !x : !\sigma \vdash^a x : \tau$. Por Lema 5.1 (3), $\Gamma = !x : !\mathcal{Q}$, $\tau = [!]\mathcal{Q}$, $\sigma = \mathcal{Q}$. Por hipótesis de inducción (H.I) $\Gamma \otimes \Delta \vdash^{a \sqcap b} x[t/x] : [!]\mathcal{Q}$. If $\tau = !\mathcal{Q}$, se conluye. En otro caso, se aplica regla $\boldsymbol{D}_!$ llegando a $\tau = \mathcal{Q}$, $\Gamma \otimes \Delta \vdash^{a \sqcap b} x[t/x] : \tau$.

- Sea $u = y$. Se tiene $\Gamma, !x : !\sigma \vdash^a y : \tau$. Por Lema 5.1 (3), este caso es imposible.

- Sea $u = (u_1, u_2)$. Se tiene $\Gamma, !x : !\sigma \vdash^a (u_1, u_2) : \tau$. Por Lema 5.1 (4), $\tau = \tau_1 \otimes \tau_2$, $a = a_1 \sqcap a_2$ o $a = -$, $\Gamma, !x : !\sigma = \Gamma_1 \otimes \Gamma_2$, $\Gamma_1 \vdash^{a_1} u_1 : \tau_1$, y $\Gamma_2 \vdash^{a_2} u_2 : \tau_2$.

  Casos:

  - Cuando $\Gamma_1 = \Gamma_1, !x : !\sigma$ y $\Gamma_2 = \Gamma_2, !x : !\sigma$, por H.I $\Gamma_1 \otimes \Delta \vdash^{a_1 \sqcap b} u_1[t/x] : \tau_1$ y $\Gamma_2 \otimes \Delta \vdash^{a_2 \sqcap b} u_2[t/x] : \tau_2$, por lo tanto, por regla $\otimes$***intro***, $\Gamma \otimes \Delta \vdash^{(a_1 \sqcap b) \sqcap a_2} (u_1[t/x], u_2[t/x]) : \tau$, correspondiente a $\Gamma \otimes \Delta \vdash^{a \sqcap b} (u_1[t/x], u_2[t/x]) : \tau$. Note que $(a_1 \sqcap b) \sqcap a_2 = a \sqcap b$.
  - Si $\Gamma_1 = \Gamma_1', !x : !\sigma$ y $!x : !\sigma \notin \Gamma_2$ ó $\Gamma_2 = \Gamma_2', !x : !\sigma$ y $!x : !\sigma \notin \Gamma_1$, es analogo al caso previo.

- Sea $u =$ **false**. Entonces, se tiene $\Gamma, !x : !\sigma \vdash^a$ **false** $: \tau$. Por Lema 5.1 (6), este caso es imposible

- Sea $u =$ **true**. Este caso es similar al anterior.

- Sea $u = \alpha u'$. Se tiene $\Gamma, !x : !\sigma \vdash^a \alpha u' : \tau$. Por lema 5.1 (8), se tienen los casos:

  - Para $|\alpha| = 1$, $\Gamma, !x : !\sigma \vdash^a u' : \tau$. Por H.I $\Gamma \otimes \Delta \vdash^{a \sqcap b} u'[t/x] : \tau$. Aplicando regla ***fase***, $\Gamma \otimes \Delta \vdash^{a \sqcap b} \alpha(u'[t/x]) : \tau$.

  - Para $|\alpha| \neq 1$, $\tau = |\sigma'|$, $\Gamma, !x : !\sigma \vdash^a u' : \sigma'$, y $a = -$. Por H.I $\Gamma \otimes \Delta \vdash^{a \sqcap b} u'[t/x] : \sigma'$. Aplicando regla ***prob***, $\Gamma \otimes \Delta \vdash^{a \sqcap b} \alpha(u'[t/x]) : \tau$. Si $a \sqcap b = \circ$, se aplica regla $\boldsymbol{D}_\circ$, y se concluye.
  - Para $u = \beta u'$, $\Gamma, !x :!\sigma \vdash^a \alpha(\beta u') : \tau$. Teniendo $\tau = |\sigma'|$, $|\alpha\beta| = 1$, $\Gamma, !x :!\sigma \vdash^a u' : \sigma''$, $\Gamma, !x :!\sigma \vdash \beta u' : \sigma'$. Por H.I. $\Gamma \otimes \Delta \vdash \beta(u'[t/x]) : \sigma'$ y $\Gamma \otimes \Delta \vdash^{a \sqcap b} u'[t/x] : \sigma''$. Por regla $\boldsymbol{prob}_1$, se llega a $\Gamma \otimes \Delta \vdash^{a \sqcap b} \alpha(\beta(u'[t/x])) : \tau$.
  - Para todo i, $u = \sum_{i=1}^n u_i$, entonces, $\tau = |\sigma'|$, $\sum_{i=1}^n |\alpha| = 1$, $\Gamma, !x :!\sigma \vdash^a u_i :!\sigma'$, para todo i, j, $u_i \perp_a u_j$. Por H.I, para todo i: $\Gamma \otimes \Delta \vdash^{a \sqcap b} u_i[t/x] : !\sigma'$, aplicando regla $\boldsymbol{sup}_\alpha$, $\Gamma \otimes \Delta \vdash^{a \sqcap b} \alpha(\sum_{i=1}^n u_i[t/x]) : \tau$.

- Sea $u = \vec{0}$. Entonces $\Gamma, !x : !\sigma \vdash^a \vec{0} : \tau$. Por Lema 5.1 (5), este caso es imposible.

- Sea $u = \textbf{let } y = u_1 \textbf{ in } u_2$. Se tiene $\Gamma, !x : !\sigma \vdash^a \textbf{let } y = u_1 \textbf{ in } u_2 : \tau$. Por Lema 5.1 (9): $\Delta_1 \vdash^c u_1 : \sigma'$, $\Delta_2, y : \sigma' \vdash^d u_2 : \tau$, $\Gamma, !x : !\sigma = \Delta_1 \otimes \Delta_2$ y $a = c \sqcap d$ o $a = -$.

  Casos:

  - Si para $i = 1, 2$, $\Delta_i = \Delta_i, !x : !\sigma$, entonces, por H.I $\Delta_1 \otimes \Delta \vdash^{c \sqcap b} u_1[t/x] : \sigma'$ and $(\Delta_2, y : \sigma') \otimes \Delta \vdash^{d \sqcap b} u_2[t/x] : \tau$. Por lo tanto, por a regla ***let***, $\Gamma \otimes \Delta \vdash^{a \sqcap b} \textbf{let } y = u_1[t/x] \textbf{ in } u_2[t/x] : \tau$. Note que $(c \sqcap b) \sqcap (d \sqcap b) = (c \sqcap d) \sqcap b = a \sqcap b$.
  - Si $\Delta_1 = \Delta_1', !x : !\sigma$ y $!x : !\sigma \notin \Delta_2$ o $\Delta_2 = \Delta_2', !x : !\sigma$ y $!x : !\sigma \notin \Delta_1$, es analogo al caso previo.

- Sea $u = \textbf{let } !y = u_1 \textbf{ in } u_2$. Analogo al caso previo.

- Sea $u = \textbf{let } (!y, !z) = u_1 \textbf{ in } u_2$. Entonces $\Gamma, !x : !\sigma \vdash^a \textbf{let } (!y, !z) = u_1 \textbf{ in } u_2 : \tau$. Por Lema 5.1 (11), se tiene: $\tau = ||\tau'|_\varepsilon|_\rho$, $\Delta_1 \vdash^c u_1 : \varepsilon \otimes \rho$, $\Delta_2, !y : !\varepsilon, !z : !\rho \vdash^d u_2 : \tau'$, $\Gamma, !x : !\sigma = \Delta_1 \otimes \Delta_2$, y $a = c \sqcap d$ ó $a = -$.

  Casos:

  - Si para $i = 1, 2$, $\Delta_i = \Delta_i', !x : !\sigma$, entonces por H.I, $\Delta_1' \otimes \Delta \vdash^{c \sqcap b} u_1[t/x] : \varepsilon \otimes \rho$, $(\Delta_2', !y : !\varepsilon, !z : !\rho) \otimes \Delta \vdash^{d \sqcap b} u_2[t/x] : \tau'$. Por lo tanto, por regla ***⊗-!elim***, $\Gamma \otimes \Delta \vdash^{(c \sqcap b) \sqcap (d \sqcap b)} \textbf{let } (!y, !z) = u_1[t/x] \textbf{ in } u_2[t/x] : \tau$. Note que $(c \sqcap b) \sqcap (d \sqcap b) = (c \sqcap d) \sqcap b = a \sqcap b$.
  - Si $\Delta_1 = \Delta_1', !x : !\sigma$ y $!x : !\sigma \notin \Delta_2$ o $\Delta_2 = \Delta_2', !x : !\sigma$ y $!x : !\sigma \notin \Delta_1$, es analogo al caso anterior.

- Sea $u = \mathbf{if}^{\circ}\ r\ \mathbf{then}\ u_1\ \mathbf{else}\ u_2$. Se tiene $\Gamma, !x : !\sigma \vdash^{a} \mathbf{if}^{\circ}\ r\ \mathbf{then}\ u_1\ \mathbf{else}\ u_2 : \tau$. Por Lema 5.1 (12), $\Delta_1 \vdash^{c} r : [!]Q$, $\Delta_2 \vdash^{d} u_1, u_2 : \tau'$, $\tau = |\tau'|_{[!]Q}$, $u_1 \perp_{c \sqcap d} u_2$, $\Gamma, !x : !\sigma = \Delta_1 \otimes \Delta_2$, y $a = c \sqcap d$ o $a = -$.

  Casos:

  - Si para $i = 1, 2$, $\Delta_i = \Delta_i', !x : !\sigma$, entonces por H. I, $\Delta_1' \otimes \Delta \vdash^{c \sqcap b} r[t/x] : [!]Q$, $\Delta_2' \otimes \Delta \vdash^{d \sqcap b} u_1[t/x], u_2[t/x] : \tau'$. Además, por regla ***if°***, $\Gamma \otimes \Delta \vdash^{(c \sqcap b) \sqcap (d \sqcap b)} \mathbf{if}^{\circ}\ r[t/x]\ \mathbf{then}\ u_1[t/x]\ \mathbf{else}\ u_2[t/x] : \tau$. Note que $(c \sqcap b) \sqcap (d \sqcap b) = (c \sqcap d) \sqcap b = a \sqcap b$.
  - Si $\Delta_1 = \Delta_1', !x : !\sigma$ y $!x : !\sigma \notin \Delta_2$ o $\Delta_2 = \Delta_2', !x : !\sigma$ y $!x : !\sigma \notin \Delta_1$, es analogo al caso anterior.

- Sea $u = \sum_{i=1}^{n} u_i$. Se tiene $\Gamma, !x : !\sigma \vdash^{a} \sum_{i=1}^{n} u_i : \tau$. Por Lema 5.1 (13), tenemos los siguientes casos:

  - Para todo i, $\Gamma, !x : !\sigma \vdash^{a} u_i : \tau$, $a = -$. Por H.I, se tiene $\Gamma \otimes \Delta \vdash^{a \sqcap b} u_i[t/x] : \tau$. Aplicando regla ***sup***, se obtiene $\Gamma \otimes \Delta \vdash^{a \sqcap b} \sum_{i=1}^{n} u_i[t/x] : \tau$. Si $a \sqcap b = \circ$, se aplica regla $\boldsymbol{D}_{\circ}$.
  - Para todo i, $u_i = \alpha_i u_i'$, $\tau = |\sigma'|$, $\Gamma, !x : !\sigma \vdash^{a} u_i' : !\sigma'$, y $\sum_{i=1}^{n} |\alpha_i|^2 = 1$, y para todo i, j, $u_i' \perp u_j'$. Por H.I se tiene: $\Gamma \otimes \Delta \vdash^{a \sqcap b} u_i'[t/x] : !\sigma'$. Aplicando regla ***sup°***: $\Gamma \otimes \Delta \vdash^{a \sqcap b} \sum_{i=1}^{n} \alpha_i (u_i[t/x]) : \tau$, donde $\tau = |!\sigma'|$.
  - Para todo i, $u_i = \sum_{i=1}^{n} [\alpha_i] \mathbf{let}\ !y = [\beta_i] u_i'\ \mathbf{in}\ e$, $\tau = |\tau'|$, $\Gamma, !x : !\sigma \vdash [\alpha] \mathbf{let}\ !y = [\beta_i] u_i'\ \mathbf{in}\ e : \tau'$ y

    $$\Gamma, !x : !\sigma \vdash^{a} \mathbf{let}\ !y = u_i'\ \mathbf{in}\ e : \tau' \tag{5.1}$$

    para todo i, $\sum_{i=1}^{n} |[\alpha_i][\beta_i]| = 1$.

    Por prueba previa, se sabe que para todo i,

    $$\Gamma \otimes \Delta \vdash^{a \sqcap b} \mathbf{let}\ !y = u_i'[t/x]\ \mathbf{in}\ e[t/x] : \tau' \tag{5.2}$$

    Considerando esto, se tienen los siguientes casos:

    · Presencia de α_i, resultando: $\alpha_i(\mathbf{let}\ !y = u_i'\ \mathbf{in}\ e)$, satisfaciendo $\sum_{i=1}^{n} |\alpha_i| = 1$. Aplicando para todo i, la regla ***prob***,

    $$\Gamma \otimes \Delta \vdash \mathbf{let}\ !y = u_i'[t/x]\ \mathbf{in}\ e[t/x] : |\tau'| \tag{5.3}$$

    Aplicando regla $\boldsymbol{let}_{\alpha}$ con ec. 5.2 y ec. 5.3: $\Gamma \otimes \Delta \vdash^{a \sqcap b} \sum_{i=1}^{n} \alpha_i (\mathbf{let}\ !y = u_i'\ \mathbf{in}\ e) : \tau$.

· Presencia de β_i, satisfaciendo $\sum_{i=1}^{n} |\beta_i| = 1$. Por demostraciones anteriores se sabe que para todo i: $\Gamma_1 \otimes \Delta \vdash^{a_1 \sqcap b} u_i[t/x] : \sigma'$ y $\Gamma_2 \otimes \Delta, !y : !\sigma'^{a_2 \sqcap b} e[t/x] : \tau'$, aplicando regla ***prob***: $\Gamma_1 \otimes \Delta \vdash \beta_i(u_i'[t/x]) : |\sigma'|$. Aplicando regla ***let***, $\Gamma \otimes \Delta \vdash \mathbf{let}\ !y = \beta_i(u_i'[t/x])\ \mathbf{in}\ e[t/x] : |\tau'|$; a esta última, con ec. 5.2 y aplicando regla ***let***$_\alpha$, $\Gamma \otimes \Delta \vdash^{a \sqcap b} \sum_{i=1}^{n} \mathbf{let}\ !y = \beta_i(u_i'[t/x])\ \mathbf{in}\ e[t/x] : \tau$.

· Presencia de α_i y β_i, resultando: $\alpha_i(\mathbf{let}\ !y = \beta_i u_i'\ \mathbf{in}\ e)$, satisfaciendo $\sum_{i=1}^{n} |\alpha_i \beta_i| = 1$. Considerando los puntos anteriores, para todo i y aplicando regla ***prob***: $\Gamma \otimes \Delta \vdash \alpha_i(\mathbf{let}\ !y = \beta_i(u_i'[t/x])\ \mathbf{in}\ e[t/x]) : ||\tau'||$. Aplicando regla ***let***$_\alpha$, se concluye: $\Gamma \otimes \Delta \vdash^{a \sqcap b} \sum_{i=1}^{n} \alpha_i(\mathbf{let}\ !y = \beta_i(u_i'[t/x])\ \mathbf{in}\ e[t/x]) : \tau$.

– Para todo i, $u_i = [\alpha_i](\mathbf{if}^\circ\ [\beta_i]u_i'\ \mathbf{then}\ u_1\ \mathbf{else}\ u_2)$. La demostración es analoga a la anterior, con los ajustes respectivos para un término $\mathbf{if}^\circ$.

- Sea $u = \pi_j\ u'$. Se tiene $\Gamma, !x : !\sigma \vdash^a \pi_j\ u' : \tau$. Por Lema 5.1 (14): $\Gamma, !x : !\sigma \vdash^\circ u' : \mathcal{Q}^n$, $n \geq j$, $\tau = (!\mathcal{Q})^j \otimes \mathcal{Q}^{n-j}$.

  Por hipótesis de inducción, $\Gamma \otimes \Delta \vdash^a u'[t/x] : \mathcal{Q}^n$, si $a = -$, se aplica la regla $\boldsymbol{D}_\circ$. Y entonces por regla $\boldsymbol{M}^\circ$, se concluye con $\Gamma \otimes \Delta \vdash^a \pi_j u'[t/x] : \tau$.

Lema 5.3 (Lema de Substitución)**.** *Para $u \neq \pi_j r$, si $\Gamma, x : \sigma \vdash^a u : \tau$ y $\Delta \vdash^b t : \sigma$, entonces $\Gamma \otimes \Delta \vdash^{a \sqcap b} u[t/x] : \tau$. Además, si $\Gamma, x : \sigma \vdash^a \pi_j r : \tau$ y $\Delta \vdash t : \sigma$, entonces $\Gamma \otimes \Delta \vdash^a \pi_j r[t/x] : \tau$.*

Demostración. Caso similar al anterior.

Teorema 5.1 (Subject reduction)**.** *Si $\Gamma \vdash^a t : \sigma$ y $t \to r$, entonces $\Gamma \vdash^a r : \sigma$.*

Demostración. Por inducción sobre la relación de reducción $\to$. Casos:

- $\mathbf{let}\ x = t\ \mathbf{in}\ u \to u[t/x]$. Por Lema 5.1 (9): $\Gamma = \Delta_1 \otimes \Delta_2$, $a = b \sqcap c$ ó $a = -$,

  1. $\Delta_1 \vdash^b t : \tau$
  2. $\Delta_2, x : \tau \vdash^c u : \sigma$

  Por ecuaciones (1),(2) y Lema 5.3: $\Gamma \vdash^{b \sqcap c} u[t/x] : \sigma$. Si $a = -$, se aplica la regla $\boldsymbol{D}_\circ$.

- $\mathbf{let}\ !x = t\ \mathbf{in}\ u \to u[t/x]$. Por Lema 5.1 (10) se tiene: $\Gamma = \Delta_1 \otimes \Delta_2, a = b \sqcap c$ ó $a = -$, $\sigma = |\sigma'|_\tau$ y

1. $\Delta_1 \vdash^b t : \tau$, donde $t = \mathbf{false}$ ó $t = \mathbf{true}$.
2. $\Delta_2,\ !x : !\tau \vdash^c u : \sigma'$

Acerca de t, se tienen los casos:

- Si $t = \mathbf{false}$, entonces por Lema 5.1 (6), $\tau = [!]\mathcal{Q}$ y $\Delta_1 = \bullet$. Por regla ***f-intro***, y eventualmente, $\boldsymbol{D}_\circ$, $\bullet \vdash^b \mathbf{false} : !\mathcal{Q}$. De ahí, por Lema 5.2, $\Gamma \vdash^{b \sqcap c} u[\mathbf{false}/x] : \sigma'$. Si $a = -$, se aplica regla $\boldsymbol{D}_\circ$. Si $\sigma \neq |\sigma'|_\tau$, aplicar regla $\boldsymbol{D}_!$.
- Si $t = \mathbf{true}$, similar al caso anterior.

- **let** $(!x, !y) = (t, u)$ **in** $e \to$ **let** $!x = t$ **in let** $!y = u$ **in** e. Por Lema 5.1 (11) se tiene: $\Gamma = \Delta_1 \otimes \Delta_2$ y $a = b \sqcap c$ ó $a = -$, $\sigma = ||\rho|_\tau|_\varepsilon$ y

  1. $\Delta_1 \vdash^b (t, u) : \varepsilon \otimes \tau$
  2. $\Delta_2,\ !x : !\varepsilon,\ !y : !\tau \vdash^c e : \rho$

  Por ecuación (1) y Lema 5.1 (4): $\Delta_1 = \Delta_{11} \otimes \Delta_{12}$, $b = b_1 \sqcap b_2$ ó $b = -$,

  3. $\Delta_{11} \vdash^{b_1} t : \varepsilon$
  4. $\Delta_{12} \vdash^{b_2} u : \tau$

  Por lo tanto

$$\dfrac{\Delta_{11} \vdash^{b_1} t : \varepsilon \quad \dfrac{\Delta_{12} \vdash^{b_2} u : \tau \quad \Delta_2, !x : !\varepsilon, !y : !\tau \vdash^c e : \rho}{\Delta_{12}, !x : !\varepsilon \vdash^{b_2 \sqcap c} \mathbf{let}\ !y = u\ \mathbf{in}\ e : |\rho|_\tau}\ !\text{let}}{\Gamma \vdash^a \mathbf{let}\ !x = t\ \mathbf{in let}\ !y = u\ \mathbf{in}\ e : \sigma}\ !\text{let}$$

- **if**$^\circ$ **true then** u **else** $u' \to u$. Por Lema 5.1 (12), $\Delta_1 \vdash^b \mathbf{true} : [!]\mathcal{Q}$, $\Delta_2 \vdash^c u : \sigma'$, con $\Delta_1 \otimes \Delta_2 = \Gamma$, $\sigma = |\sigma'|_{[!]\mathcal{Q}}$ y $b \sqcap c = a$ ó $a = -$.

  Por Lema 5.1 (7), $\Delta_1 = \bullet$, así que $\Gamma = \Delta_2$. Si $\sigma = |\sigma|$, entonces por regla $\boldsymbol{D}_!$, $\Gamma \vdash^c u : \sigma$. Otro caso, $\sigma' = \sigma$, y así, $\Gamma \vdash^c u : \sigma$. Además, si $b = \circ$, entonces $c = a$, y se concluye. Otro caso, $a = -$, se usa la regla $\boldsymbol{D}_\circ$ para obtener $\Gamma \vdash^a u : \sigma$.

- **if**$^\circ$ **false then** u **else** $u' \to u'$. Similar al caso previo.

- $\pi_j \left(\underbrace{\sum_{i=1}^{n} \alpha_i \prod_{h=1}^{m} b_{hi}}_{t} \right) \underset{(p_b)}{\longrightarrow} b \otimes \phi_b$.

  Por Lema 5.1 (14) en t: $\sigma = (!\mathcal{Q})^j \otimes \mathcal{Q}^{n-j}$, $\Gamma \vdash^c \sum_{i=1}^{n} \alpha_i \prod_{h=1}^{m} b_{hi} : \mathcal{Q}^n$. De la anterior sentencia y por Lema 5.1 (13): $\Gamma \vdash^\circ \prod_{h=1}^{m} b_{hi} : !\mathcal{Q}^n$, $\sum_{i=1}^{n} |\alpha_i|^2 = 1$, $\forall i,k \ \prod_{h=1}^{m} b_{hi} \perp \prod_{h=1}^{m} b_{hk}, m \geq 1$. De lo anterior, se obtiene: $\forall h$, si $\Delta_h \vdash^{c_h} b_{hi} : \sigma_h$, $\underset{h}{\otimes}\sigma_h = !\mathcal{Q}^m$, $\circ = \underset{h}{\sqcap} c_h$, $\underset{h}{\otimes}\Delta_h = \Gamma$, entonces: $\forall i, \Delta_h \vdash^\circ b_{hi} : !\mathcal{Q}$. Por lo tanto $\Delta_j \vdash^\circ \prod_{h=1}^{j} b_{hi} : !\mathcal{Q}^j$ y $\overset{m}{\underset{j+1}{\otimes}} \Delta_{j+1} \vdash^\circ \sum_{i \in T_b} \left(\frac{\alpha_i}{\sqrt{\sum_{r \in T_b} |\alpha_r|^2}} \right) \prod_{h=j+1}^{m} b_{hi} : \mathcal{Q}^{m-j}$.

  Simplificando $b = \prod_{h=1}^{j} b_{hi}$, $i \leq n$, y $\phi_b = \sum_{i \in T_b} \left(\frac{\alpha_i}{\sqrt{\sum_{r \in T_b} |\alpha_r|^2}} \right) \prod_{h=j+1}^{m} b_{hi}$, donde $T_b = \{i \leq n | b_{1i} \otimes \cdots \otimes b_{ji} = b\}$. Concluyendo que $\Gamma \vdash^a b \otimes \phi_b$. Si $a = \circ$, se ha terminado; sino se aplica la regla $\boldsymbol{D}_\circ$.

El resultado de unir el lenguaje QML y Lambda-$\mathcal{S}$, propició un modelo más íntegro, considerando datos clásicos y cuánticos, estableciendo reglas para tipar correctamente programas, abordando la linealidad en superposiones y anexando la regla de medición, la cual en la semántica denotacional de QML no se encontraba.

De forma inminente, el estudio de las propiedades acerca de términos cuánticos (estrictos) y la definición de su semántica denotacional resulta interesante seguir, sin embargo, por ahora con esto se dan por concluidos los aportes de esta investigación, agradeciendo a Alejandro Diáz-Caro el tiempo y los extraordinarios aportes a este trabajo.

The present is the only things that has no end.

— Erwin Schrödinger

6

Conclusiones y trabajo a futuro

El cómputo cuántico ofrece expectativas optimistas para solucionar problemas de forma más eficiente que las soluciones hasta ahora planteadas por el cómputo clásico, y partiendo de esto, nuestra investigación se centró en contribuir con el cómputo cuántico y en lenguajes de programación cuánticos.

El motivo de trabajar lenguajes, es que son un medio para interactuar y establecer instrucciones del humano a una computadora, en particular, el lenguaje de programación cuántico QML, el cual manipula datos y control clásicos y cuánticos.

En este trabajo se abordó la inherente propiedad de reversibilidad que tiene el cómputo cuántico, siendo interesante debido a que si aún está en desarrollo una computadora cuántica, entonces, esta propiedad podría ser modelada de alguna manera que garantizará que determinado lenguaje es reversible.

De esta idea y otras investigaciones, se propuso definir semánticamente la reversibilidad en el lenguaje de programación cuántico QML. La construcción teórica de esto, se trabajo inicialmente con sólo datos clásicos y posteriormente se anexaron datos cuánticos.

La reversibilidad se logró a través de incorporar a la semántica, funciones inversas que en conjunto con las funciones ya definidas, generan un historial de operaciones con la forma $\boldsymbol{h_{t_1-}}; t_1 \circ \boldsymbol{h_{t_2-}}; t_2 \circ \cdots \circ \boldsymbol{h_{t_n-}}; t_n$, donde los $\boldsymbol{h_{t_i-}}$ son las funciones inversa de cada t_i, respectivamente.

Dada una pila $\boldsymbol{h_{t_1-}}; t_1 \circ \boldsymbol{h_{t_2-}}; t_2 \circ \cdots \circ \boldsymbol{h_{t_n-}}; t_n$, se separarón las funciones convencionales de sus inversas, llegando a funciones de la forma $\boldsymbol{h_{t_n-}; \cdots ; h_{t_2-}; h_{t_1-}}; t_1 \circ t_2 \circ \cdots \circ t_n$. Las funciones inversas que se proponen en este trabajo, garantizan semánticamente la reversibilidad.

Teniendo la pila de historial, se incluyo el conjunto de reglas que permiten ejecutar reversibilidad, esto es, cuándo se tenga $\boldsymbol{h_{t_n-}; \cdots ; h_{t_2-}; h_{t_1-}}; v$, donde v es un valor, a partir de éste se estableció cómo regresar en cada paso hasta el valor inicial de un programa y en consecuencia, lograr reversibilidad.

Con respecto a la asociación del lenguaje QML y lambda-$\mathcal{S}$ se logra progresar en el nivel de abstracción, recordando se tienen los circuitos cuánticos, cálculo lambda y finalmente lenguajes de programación cuánticos. Entonces, QML operacionalmente se definió en términos de circuitos cuánticos, posteriormente, en esta investigación se vinculó con cálculo lambda y dió pauta a un lenguaje de programación cuántico más expresivo.

La cohesión entre ambos lenguajes, implicaron modificar la sintaxis original de QML, agregando la identificación de términos estrictos, no estrictos y la regla de medición cuántica; los términos estrictos son aquellos que están normalizados y son ortogonales. A partir de la nueva sintaxis, se modificaron y anexaron nuevas reglas para construir programas bien tipados.

También se describe el modelo operacional considerando términos en superposición y mediciones cuánticas. Durante la conexión de ambos lenguajes, se tuvo cuidado con respecto a la no clonación de datos cuánticos (weakening) y la perdida de información (contraction). Y se concluye con la realización de pruebas formales respecto a esta propuesta.

Finalmente, se reportan tres artículos como resultado de esta investigación y un artículo sometido a revisión [50, 48, 49].

Los aportes siguientes e inmediatos de este trabajo serían: Agregar la semántica operacional del lenguaje propuesto (de QML y lambda-$\mathcal{S}$), posterior verificar la propiedad de *Strictness*, la cual garantizará que todo vector del lenguaje, semánticamente tiene norma 1, es decir, satisface que $\forall v \in [\![t]\!]$, $||v|| = 1$; también considerar la definición de una teoría ecuacional, así como una posible implementación en una computadora clásica.

Bibliografía

[1] S. Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

[2] S. Abramsky. A cook's tour of a simple quantum programming language. *Theory Day of the Dutch Association for Theoretical Computer Science*, 2004.

[3] S. Abramsky. A structural approach to reversible computation. *Theoretical Computer Science*, 347(3):441 – 464, 2005.

[4] T. Altenkirch and J. Grattage. A functional quantum programming language. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 249 – 258, June 2005.

[5] T. Altenkirch, J. Grattage, J. K Vizzotto, and A. Sabry. An algebra of pure quantum programming. *Electronic Notes in Theoretical Computer Science*, 170:23 – 47, 2007. Proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005).

[6] P. Arrighi, A. Díaz-Caro, and B. Valiron. The vectorial lambda-calculus. *Information and Computation*, 254(1):105–139, Jun 2017.

[7] P. Arrighi and G. Dowek. Lineal: A linear-algebraic lambda-calculus. *Logical Methods in Computer Science*, (13):1–8, 02 2017.

[8] S. Awodey. *Category Theory*. Oxford Logic Guides. Ebsco Publishing, 2006.

[9] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics. Elsevier Science, 2014.

[10] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17(6):525–532, 11 1973.

[11] S. Bettelli, T. Calarco, and L. Serafini. Toward an architecture for quantum programming. *The European Physical Journal D - Atomic, Molecular, Optical and Plasma Physics*, 25(2):181–200, 08 2003.

[12] V. Danos and E. Kashefi. Determinism in the one-way model. *Physical Review A*, 74(5), 11 2006.

[13] V. Danos and E. Kashefi. Pauli measurements are universal. volume 170, pages 95 – 100. Proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005), 2007.

[14] V. Danos, E. Kashefi, and P. Panangaden. The measurement calculus. *J. ACM*, 54(2), 04 2007.

[15] H. De Raedt and K. Michielsen. Computational methods for simulating quantum computers. In *Handbook of Theoretical and Computational Nanotechnology*, volume 3, page 248. American Scientific Publisher, 2006.

[16] A. Díaz-Caro. Agregando medición al cálculo de van tonder. Master's thesis, Universidad Nacional de Rosario, Argentina, 12 2007.

[17] A. Díaz-Caro, P. Arrighi, M. Gadella, and J. Grattage. Measurements and confluence in quantum lambda calculi with explicit qubits. *Electronic Notes in Theoretical Computer Science*, 270(1):59 – 74, 2011. Proceedings of the Joint 5th International Workshop on Quantum Physics and Logic and 4th Workshop on Developments in Computational Models (QPL/DCM 2008).

[18] A. Díaz-Caro, G. Dowek, and J.P. Rinaldi. Two linearities for quantum computing in the lambda calculus. *Biosystems*, pages 1–69, 2019.

[19] S. S. Feitosa, J. K. Vizzotto, E. K. Piveta, and A. R. Du Bois. Fjquantum - a quantum object oriented language. *Electron. Notes Theor. Comput. Sci.*, 324(C):67–77, 09 2016.

[20] S.S. Feitosa, J.K. Vizzotto, E.K. Piveta, and A.R. Du Bois. Fjquantum - a quantum object oriented language. *Electronic Notes in Theoretical Computer Science*, 324:67 – 77, 2016.

[21] S. J. Gay. Quantum programming languages: Survey and bibliography. *Mathematical. Structures in Comp. Sci.*, 16(4):581–600, 08 2006.

[22] S. J. Gay and R. Nagarajan. Communicating quantum processes. In *Proceedings of the 32Nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '05, pages 145–157, New York, NY, USA, 01 2005. ACM.

[23] S. J. Gay and R. Nagarajan. Communicating quantum processes. *SIGPLAN Not.*, 40(1):145–157, January 2005.

[24] J. J. Grattage. A functional quantum programming language.

[25] C. Hankin. *An Introduction to Lambda Calculi for Computer Scientists*. Texts in computing. Kings College, 2004.

[26] N. B. Hasser, J. P. La Salle, and J. A. Sullivan. *Análisis matemático-Curso intermedio*. trillas, 1990.

[27] C. Heunen and M. Karvonen. Reversible monadic computing. *Electronic Notes in Theoretical Computer Science*, 319:217 – 237, 2015. The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI).

[28] P. Jorrand and M. Lalire. *From Quantum Physics to Programming Languages: A Process Algebraic Approach*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[29] J. Karczmarczuk. Structure and interpretation of quantum mechanics: A functional framework. In *Proceedings of the 2003 ACM SIGPLAN Workshop on Haskell*, Haskell '03, pages 50–61, New York, NY, USA, 2003. ACM.

[30] E. Knill. *Conventions for Quantum Pseudocode*. Office of Scientific and Technical Information, U.S. Dept. of Energy, 09 1996.

[31] U. Lago, A. Masini, and M. Zorzi. On a measurement-free quantum lambda calculus with classical control. *Mathematical. Structures in Comp. Sci.*, 19(2):297–335, 04 2009.

[32] M. Lampis, K. G. Ginis, M. A. Papakyriakou, and N. S. Papaspyrou. Quantum data and control made easier. *Electron. Notes Theor. Comput. Sci.*, 210:85–105, 07 2008.

[33] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 07 1961.

[34] W. Mauerer. Semantics and simulation of communication in quantum programming. Master's thesis, University Erlangen-Nuremberg, 2005.

[35] P. Maymin. Extending the lambda calculus to express randomized and quantumized algorithms, 01 1996.

[36] D. McMahon. *Quantum Computing Explained*. John Wiley & Sons, 2007.

[37] J. A. Miszczak. High-level structures for quantum computing. *Synthesis Lectures on Quantum Computing*, 4:1–129, 05 2012.

[38] H. Mlnar'k. *Quantum Programming Language LanQ*. phdthesis, Faculty of Informatics, September 2007.

[39] E. Moggi. Computational lambda-calculus and monads. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, pages 14–23. IEEE Press, 1989.

[40] S.-C. Mu and R. Bird. Functional quantum programming. In *In Proceedings of the 2nd Asian Workshop on Programming Languages and Systems*, pages 1–14, 2001.

[41] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 2000.

[42] N. Nishida, A. Palacios, and G. Vidal. Reversible computation in term rewriting. *Journal of Logical and Algebraic Methods in Programming*, 94:128 – 149, 2018.

[43] B. Ömer. A procedural formalism for quantum computing. *AIP Conference Proceedings*, 627(1):276, 2002.

[44] N. K. Papanikolaou. Techniques for design and validation of quantum protocols, 2004.

[45] J. Patrzyk, B. Patrzyk, K. Rycerz, and M. Bubak. Towards a novel environment for simulation of quantum computing. *Computer Science*, 16(1):1–103, 2015.

[46] S. Perdrix. Quantum patterns and types for entanglement and separability. *Electronic Notes in Theoretical Computer Science*, 170(Supplement C):125 – 138, 2007. Proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005).

[47] S. Perdrix. A hierarchy of quantum semantics. *Electronic Notes in Theoretical Computer Science*, 192(3):71 – 83, 2008. Proceedings of the Third International Workshop on Developments in Computational Models (DCM 2007).

[48] N. Plata-Cesar and J. R. Marcial-Romero. Historial y reversibilidad en el sublenguaje clásico qml. *Research in Computing Science*, 147(12):1–12, 2018.

[49] N. Plata-Cesar, J. R. Marcial-Romero, and J.A. Hernández-Servín. A history and reversibility for quantum programming language QML. In *Proceedings of the Eleventh Latin American Workshop on Logic/Languages, Algorithms and New Methods of Reasoning*, pages 25–39, Puebla, Mexico, 11 2018.

[50] N. Plata-Cesar, J. R. Marcial-Romero, and J.A. Hernández-Servín. Lenguajes de programación cuánticos. *Komputer Sapiens*, 1(12), 2020.

[51] S. Roman. *Advanced Linear Algebra*. Springer, softcover of or edition, 2008.

[52] A. Sabry. Modeling quantum computing in haskell. In *Proceedings of the 2003 ACM SIGPLAN Workshop on Haskell*, Haskell '03, pages 39–49, New York, NY, USA, 2003. ACM.

[53] J. W. Sanders and P. Zuliani. Quantum programming. In *In Mathematics of Program Construction*, pages 80–99. Springer-Verlag, 1999.

[54] P. Selinger. A brief survey of quantum programming languages. In *In Proceedings of the 7th International Symposium on Functional and Logic Programming*, pages 1–6. Springer, 2004.

[55] P. Selinger. Towards a semantics for higher-order quantum computation. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages*, pages 127–143, Tehran, 2004. Workshop on Quantum Computing.

[56] J. Skibinski. Haskell simulator of quantum computer, 04 2001.

[57] D. A. Sofge. A survey of quantum programming languages: History, methods, and tools. In *Second International Conference on Quantum, Nano and Micro Technologies (ICQNM 2008)*, pages 66–71, 02 2008.

[58] A. V. Tonder. A lambda calculus for quantum computation. *SIAM Journal Comput.*, 33(5):1109–1135, 05 2004.

[59] A Van Tonder. Quantum computation, categorical semantics and linear logic. pages 1–21, 10 2003.

[60] J. Kaizer Vizzotto and A.Carlos da Rocha Costa. Concurrent quantum programming in haskell. In *VII Congresso Brasileiro de Redes Neurais*, pages 1–6. Sessão de Computação Quântica, Brazilian Association of Computational Intelligence, 2005.

[61] J. Wallace. Quantum computer simulators - a review version 2.1, 1999.

[62] Mingsheng Ying. Foundations of quantum programming. In *APLAS 2010*, 2016.

[63] M. Zorzi. On quantum lambda calculi: a foundational perspective. *Mathematical Structures in Computer Science*, 26(7):1107–1195, 10 2016.

A

Apéndices
Cálculo lambda

Previo a cálculo lambda cuántico λ_q, se mencionará el denominado cálculo lambda intermedio (λ_i), propuesto por Van Tonder, este lenguaje permite la transición del contexto clásico a cuántico[25, 9, 58].

Cálculo lambda clásico

El cálculo lambda clásico es un lenguaje de programación motivado por la lógica matemática y las funciones computacionales. Éste representa funciones que no tienen ningún tipo establecido, lo que se define como lenguaje no tipado. El cálculo lambda (λ), es encargado de representar funciones computacionales a partir de un conjunto de elementos dados por el mismo lenguaje [25, 9]. Se dejará al lector el estudio de este lenguaje.

A.1 Cálculo lambda intermedio λ_i

Este cálculo λ_i, se encuentra en la transición de cálculo λ clásico a cuántico λ_q, aquí se encuentran las primeras modificaciones que se hicieron al cálculo clásico que permitieron corregir, anexar o ajustar lo necesario para definir λ_q [58]. Aquí se mencionarán las definiciones más significantes para esta investigación.

A.1.1 Sintaxis

Las expresiones son llamadas *términos*, y son construidas recursivamente de variables, abstracciones y aplicaciones. El primer ajuste respecto al entorno clásico, es

agregar los símbolos primitivos 0 y 1, así como operaciones cuánticas (compuertas), denotadas por letras mayúsculas (Lista de compuertas cuánticas D).

t::=	terms:
x	variable
(λx.t)	abstraction
(t t')	application
c	constant
c::=	constants:
0 | 1 | H | S | R_3 | cnot | X | Y | Z | ...	

Sintaxis del lenguaje intermedio λ_i.

Un término de la forma $(\lambda x.t)$ es llamado *abstracción funcional* o sólo abstracción y representa una función $x \to t$, por ejemplo, la abstracción $(\lambda x.x)$, es la función identidad $x \to x$. Respecto a $(t\ t)$, es llamada *aplicación*, significa que es la operación de aplicar una función t a su argumento t'. También se tienen las constantes c, es el caso de los símbolos primitivos 0 y 1, así como las compuertas cuánticas H, S, R_3, cnot, ... Las constantes c, contendrán los elementos cuánticos que permitirán definir funciones con éstos.

Observación 1. Cuando se trata de términos que deriven en expresiones cuánticas convencionalmente se utilizará la notación de ket $|\ \rangle$. Por ejemplo en la aplicación $|(H\ 1)\rangle$, se implementa ya que la compuerta H genera un estado cuántico.

Observación 2. El símbolo $\to$ se interpreta como derivación de un término a otro.

Es relevante mencionar que las expresiones generadas con los términos de tal sintaxis, tienen como propósito emitir un resultado o valor, para llegar a esto, se recurre a un estrategia nombrada *llamada por valor o call-by-value* (Definición B.3), la cual permitirá definir qué elementos serán considerados como valores.

Dada esta sintaxis, se ejemplifica una expresión cuántica.

A.1.2 Historial de cálculos

Para comprender la idea de reversibilidad, se retoma tal definición, y así posteriormente conocer qué es un *historial de cálculos*.

Definición A.1 (Reversibilidad)**.** La reversibilidad consiste en que cualquier proceso de cómputo partiendo de un punto inicial y llegando a una conclusión final, éste puede regresar una vez en el final del mismo, a la operación previa. Esto través de algún método o reglas aplicadas a términos λ [10, 58].

Entonces, para llevar un control de las operaciones realizadas, Tonder retoma lo descrito por Bennett ([10]), quien determina que si se tiene cierto cómputo clásico, se puede transformar de tal manera que sea un cómputo reversible, es decir, que se tenga el rastro de las ejecuciones previas y poder regresar al cálculo anterior.

Este proceso de pasar de un término a otro o que también llamaremos *reducción*, es interpretado de la siguiente manera:

$$x \to (x, \beta(x)) \to (x, \beta(x), \beta^2(x)) \to (x, \beta(x), \beta^2(x), \beta^3(x)) \to ...$$

Esto significa que x es el término a ser calculado y $\beta(x)$ es el cómputo resultante de su reducción, y así sucesivamente.

De tal procedimiento, se puede ya definir formalmente el historial, que se encargará de preservar e ir concatenando en la parte izquierda los cálculos realizados.

Definición A.2 (Registro cuántico)**.** Un estado computacional se toma como una superposición cuántica de secuencias de la forma: $h_1; \ldots; h_n; t$, donde a $h_1; \ldots; h_n$ se le define como *historial de seguimiento* y a t el *registro computacional*.

Para el caso de cálculo lambda intermedio y cuántico, el historial se separa por el símbolo ;.

Observación 3 ($\mathcal{H}$)**.** Dado un registro cuántico, un historial $h_1; \ldots; h_n$ se puede resumir como $\mathcal{H}$. Donde éste contendrá todo el historial de las operaciones ejecutadas.

Implementado el historial en el cálculo λ_i, se tiene:

Ejemplo A.1. $|(H\ 0)\rangle \to \frac{1}{\sqrt{2}}(|(H\ 0);\ 0\rangle + |(H\ 0);\ 1\rangle) = |(H\ 0)\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

Dado el ejemplo A.1, el historial y el cálculo se está repitiendo en la superposición, por lo tanto, intuitivamente éste se podría separar y seguir con las operaciones correspondientes. De ahí se tiene que:

Definición A.3 (Factorización de historial)**.** Dada una expresión de la forma $|h'; t_1\rangle + |h'; t_2\rangle + \cdots + |h'; t_n\rangle$, se puede expresar como $|h'\rangle \otimes \Big(|t_1\rangle + |t_2\rangle + \cdots + |t_n\rangle\Big)$, lo cual se define como *factorizar* el historial. Esto se logra moviendo el historial a la izquierda del registro computacional, separado con el operador $\otimes$.

En la reversibilidad sólo se requiere del operador aplicado y conservar el término actual. Por lo tanto, se procede a eliminar la información innecesaria.

Observación 4. Dado un historial $\mathcal{H}$ de un registro cuántico, en cada paso se remplazarán los subtérminos que no son necesarios para la reversibilidad por el guión bajo "_".

Ejemplo A.2. Si se tiene:
$|(H\ (H\ 0))\rangle \rightarrow \frac{1}{\sqrt{2}}\Big(|(H\ (H\ 0))\rangle\Big)\otimes\Big(|(H\ 0);0\rangle+|(H\ 0);1\rangle+|(H\ 1);0\rangle-|(H\ 1);1\rangle\Big)$
aplicando el procedimiento de la Observación 4, se obtiene:
$|(H\ (H\ 0))\rangle \rightarrow \frac{1}{\sqrt{2}}\,|(\_\ (H\ \_))\rangle\otimes\Big(|(H\ \_);0\rangle+|(H\ \_);1\rangle+|(H\ \_);0\rangle-|(H\ \_);1\rangle\Big)$
Aquí se aplica la factorización, concluyendo que el resultado final es $|0\rangle$:
$|(H\ (H\ 0))\rangle = |(\_(H\ \_));(H\ \_)\rangle\otimes|0\rangle = \underbrace{|(\_(H\ \_));(H\ \_)\rangle}_{\textbf{Historial}}\otimes\underbrace{|0\rangle}_{\textbf{Reg. computacional}}$

Descrito lo relacionado a reversibilidad basada en un historial de cálculos, se continua definiendo cálculo λ_i.

Reducción β en λ_i

Las expresiones formadas a partir de sintaxis respectiva, éstas pueden ser adecuadamente reducidas a partir de una regla denominada *reducción* β, la cual consiste en: $((\lambda x.t)\ v) = t[v/x]$.

Dada esta regla se observa un término nuevo, definido como v, el cual se nombra como *valor*, definidos a partir de la siguiente tabla:

v::=	
x	variable
c	constant
$(\lambda x.\mathrm{t})$	abstraction value

Valores del lenguaje intermedio λ_i.

La utilidad de definir valores radica en que permitirán establecer reglas para ir reduciendo expresiones con términos λ. El procedimiento para definir tales reglas, se basa en la estrategía *llamada por valor (call-by-value)* definida previamente en la sección B.3. Se proceden a precisar las reglas de reducción en cálculo λ_i.

A.1.3 Modelo operacional λ_i

Se definen las reglas de evolución o reducción en λ_i, denominadas en conjunto como *modelo operacional*. Estas reglas permiten pasar de un término a otro, teniendo un orden y con el propósito de llegar un valor final irreducible.

$\dfrac{t_1 \ \rightarrow \ h_1; t_1'}{\mathcal{H}; (t_1\ t_2) \ \rightarrow \ \mathcal{H};\ (h_1\ \_);\ (t_1'\ t_2)}$	(app_1)
$\dfrac{t_2 \ \rightarrow \ h_2; t_2'}{\mathcal{H}; (v_1\ t_2) \ \rightarrow \ \mathcal{H};\ (\_\ h_2);\ (v_1\ t_2')}$	(app_2)
$\mathcal{H};\ ((\lambda x.t)\ v) \ \rightarrow \ \mathcal{H};\ ((\lambda x.\overline{t}_x)\_);\ t[v/x]$	(β_1) Si $x \in Fv(t)$
$\mathcal{H};\ ((\lambda x.t)\ v) \ \rightarrow \ \mathcal{H};\ ((\lambda x.\_)\ v);\ t$	(β_1) Si $x \notin Fv(t)$
$\lvert\mathcal{H};\ (c_U\ \phi)\rangle \ \rightarrow \ \lvert\mathcal{H};\ (c_U\ \_)\rangle \otimes U\,\lvert\phi\rangle$	(U)
$\mathcal{H};\ t \rightarrow \mathcal{H};\ \_;\ t$	(Id) en otro caso

Modelo operacional de λ_i.

Definición A.4. $\overline{t_x}$, consiste en que dado un término t, se remplazarán todos los elementos distintos de x por el símbolo _, y las apariciones de la variable x se preservarán. Por ejemplo: si $t \equiv (\lambda y.\lambda x)$, entonces, $\overline{t_x} \equiv (\_.\lambda x)$.

Uno problema que se presenta en tales reglas, es que no siempre termina un programa, ya que la regla aplicada en las últimas derivaciones será siempre (Id), la forma de terminarlo se resolverá posteriormente.

Otro problema se identifica cuando el historial de operaciones quedo enredado con el estado, por ejemplo, si se tiene $(|((\lambda x.\_)\ 0); 0\rangle + |((\lambda x.\_)\ 1); 1\rangle)$, los historiales son $((\lambda x.\_)\ 0)$ y $((\lambda x.\_)\ 1)$; el problema se presenta por qué la constante 0 y 1, no fueron eliminadas durante la derivación respectiva, sino que el estado se preservo en el lugar inadecuado.

Para evitar este tipo de dificultades, se definen subexpresiones definidas y otros conceptos que solventen los problemas mencionados. Estas deficiencias se corrigen en el cálculo λ cuántico. Esto se abordará en la siguiente sección.

Cálculo lambda cuántico

En esta sección se continúa con λ_i, corrigiendo errores detectados, así como la incorporación final de propiedades cuánticas. En este bloque se encontrará la estructura, reglas de operación y teoría ecuacional del cálculo lambda cuántico (λ_q); éste se mantiene como un lenguaje no tipado. Se comenzará dando algunas definiciones necesarias para su comprensión.

A.2 Cálculo lambda cuántico

Rumbo a definir formalmente cálculo λ_q, se describen las definiciones de *subexpresión definida* y *términos no lineales*.

Definición A.5 (Subtérmino definido)**.** Se dice que un subtérmino es *definido* con respecto a la base computacional, si éste es textualmente el mismo para todas las ramas de una superposición. Por ejemplo: $\frac{1}{\sqrt{2}}(|(\lambda x.0)\ 0\rangle + |(\lambda x.0)\ 1\rangle)$, donde la abstracción $(\lambda x.0)$ es definida.

Las subexpresiones definidas pueden considerarse como un recurso clásico, mientras las *no definidas* representan fuentes cuánticas puras.

Observación 5. La importancia del control de subexpresiones definidas y no definidas, es hacer imposible escribir una función que descarte una fuente no definida. Es decir, evitar eliminar fuentes cuánticas a través de reglas de derivación.

Definición A.6 (Términos no lineales)**.** Los términos de la forma !t son llamados *no lineales*. Se conciben como *términos definidos* (fuentes clásicas) con respecto a la base computacional, estos podrían ser descartados y copiados.

Los términos lineales, pueden ser no definidos (expresiones cuánticas), ya que pueden contener qubits en superposiciones cuánticas con respecto a la base computacional; esto se puede asociar en un entorno algebraico como una combinación lineal.

A.2.1 Sintaxis de λ_q

t::=	términos:
x	variable
(λx.t)	abstracción
(t t)	aplicación
c	constante
!t	término no lineal
(λ!x.t)	abstracción no lineal
c::=	constantes:
0 | 1 | H | S | R_3 | cnot | X | Y | Z | ...	

Sintaxis del lenguaje λ_q.

A.2.2 Términos bien formados en λ_q

Esto corresponde a la restricción de que los argumentos lineales aparezcan en el cuerpo de una función y que todas las variables libres (Definición B.2) de un término !t refieran a variables no lineales.

Esto significa que si los argumentos lineales aparecen en el cuerpo, entonces, son recursos cuánticos (posiblemente una superposición como $\frac{1}{\sqrt{n}}(|(\lambda x.0)\ c_1\rangle + |(\lambda x.0)\ c_2\rangle + \cdots + |(\lambda x.0)\ c_n\rangle$. Y que las variables libres, por ejemplo x refieran a variables no lineales, como $!x$, esto para que no puedan ser desarrolladas ni descartadas.

También las reglas de términos bien formados impiden escribir funciones que descarten argumentos lineales, es decir, que el argumento eliminado sea una superposición, lo cual, puede causar que no haya equivalencia entre reglas de reducción.

Otra consideración es que en una abstracción lineal se pueden usar todos los términos no lineales $!t$ que se quiera (o ninguno), pero debe haber sólo un término lineal t, en el cuerpo de la función. A continuación en la Tabla A.2.2, se visualizan las restricciones para escribir funciones que no conlleven los problemas previos.

$\dfrac{}{\vdash c}$	Const	$\dfrac{\Gamma \vdash t}{\Gamma, !x \vdash t}$	Weakening
$\dfrac{}{x \vdash x}$	Id		
$\dfrac{!x_1, \ldots, !x_n \vdash t}{!x_1, \ldots, !x_n \vdash !t}$	Promotion	$\dfrac{\Gamma, x \vdash t}{\Gamma \vdash (\lambda x.t)}$	$\multimap$-I
$\dfrac{\Gamma, x \vdash t}{\Gamma, !x \vdash t}$	Dereliction	$\dfrac{\Gamma, !x \vdash t}{\Gamma \vdash (\lambda !x.t)}$	$\to$-I
$\dfrac{\Gamma, !x, !y \vdash t}{\Gamma, !z \vdash t[z/x, z/y]}$	Contraction	$\dfrac{\Gamma \vdash t_1 \quad \Delta \vdash t_2}{\Gamma, \Delta \vdash (t_1\ t_2)}$	$\multimap$-E

Reglas para términos bien formados en cálculo λ_q.

Antes de abordar las reglas, se establecerá como interpretar el símbolo $\vdash$, también qué es un contexto y una consideración entre términos t y $!t$.

Definición A.7 ($\vdash$)**.** Las teorías son presentadas como un conjunto de teoremas dados, axiomas y reglas para derivar nuevos teoremas. Usualmente se escribe como: $\Gamma \vdash thm$, donde thm representa el nuevo teorema y $\vdash$ el símbolo de derivación.

$\vdash thm$, significa que el nuevo teorema es demostrable en una teoría T, es decir, es un axioma o derivable de los axiomas usando las reglas de T y otras derivaciones de teoremas.

Definición A.8 (Contextos)**.** Sean Γ y Δ *contextos*, definidos como conjuntos que contienen supuestos de linealidad de la forma x y $!x$, donde cada x es distinta.

Durante la investigación se ha mencionado el problema de evaluar o derivar términos de la forma $!t$, para solucionar esto, se considero el aporte de Abramsky ([1]), quien amplia la definición de los valores para seguir reduciendo con la estratégia call-by-value, permitiendo aplicar reglas que no desarrollen los $!t$:

v::=	values:
x	variable
c	constant
$(\lambda x.\mathrm{t})$	linear abstraction
$(\lambda !x.\mathrm{t})$	nonlinear abstraction
!t	!-suspension

Valores en el lenguaje λ_q.

Dada la sintaxis, las reglas de términos bien formados y los valores, se procede a definir el modelo operacional de λ_q. Este modelo, se basa en el cálculo λ_i (Sección A.1.3), pero permitiendo transitar y corregir errores, para modelar finalmente expresiones cuánticas.

A.2.3 Modelo operacional de λ_q

Las siguientes reglas, permiten ir evolucionando en una función hasta que ésta emita una resultado. Se puede presentar el caso de no terminar y quedar ciclado en una reducción.

Las reglas son similares a las descritas en la Tabla A.1.3, pero incorporando los términos y argumentos no lineales (!). Estas se describen en la Tabla A.2.3.

$\dfrac{t_1 \;\to\; h_1; t_1'}{\mathcal{H}; (t_1\ t_2) \;\to\; \mathcal{H};\ (h_1\ \_);\ (t_1'\ t_2)}$	(app_1)
$\dfrac{t_2 \;\to\; h_2; t_2'}{\mathcal{H};\ (v_1\ t_2) \;\to\; \mathcal{H};\ (\_\ h_2);\ (v_1\ t_2')}$	(app_2)
$\mathcal{H};\ ((\lambda x.t)\ v) \;\to\; \mathcal{H};\ ((\lambda x.\bar{t}_x)\_);\ t[v/x]$	(β)
$\mathcal{H};\ ((\lambda !x.t)\ !t') \;\to\; \mathcal{H};\ ((\lambda !x.\bar{t}_x)\_);\ t[t'/x]$	$(!\beta_1)$ Si $x \in Fv(t)$
$\mathcal{H};\ ((\lambda !x.t)\ !t') \;\to\; \mathcal{H};\ ((\lambda !x.\_)\ !t');\ t$	$(!\beta_2)$ Si $x \notin Fv(t)$
$\lvert\mathcal{H};\ (c_U\ \phi)\rangle \;\to\; \lvert\mathcal{H};\ (c_U\ \_)\rangle \otimes U\,\lvert\phi\rangle$	(U)
$\mathcal{H};\ t \to \mathcal{H};\ \_;\ t$	(Id) en otro caso

Tabla A.1: Modelo operacional de λ_q.

Estas reglas permiten crear superposiciones cuánticas no arbitrarias, donde los términos en la superposición sólo cambian en las posiciones que contienen las constantes 0 y 1. Para esto, se define el concepto *términos congruentes* y algunos lemas [58, 16].

Definición A.9 (Términos congruentes)**.** Dos términos son congruentes si ellos coinciden símbolo por símbolo excepto posiblemente en las posiciones que contienen 0 o 1.

Dada esta definición, se definen ciertos lemas asociados.

Lema 1. Todos los términos en una superposición obtenidos mediante una secuencia de reducción de un término inicial definido son congruentes.

Lema 2. Si t es bien formado y $|\mathcal{H}; t\rangle \rightarrow \Sigma_i c_i \, |\mathcal{H}'_i; t'_i\rangle$, entonces, todos los términos t'_i que aparecen en la superposición resultante son bien formados.

Lema 3. Comenzando con un término inicial definido, cualquier *!-suspensión* que ocurra durante la reducción es definido con respecto a la base computacional.

Respecto al historial, se determina que:

Lema 4. Dado un término inicial definido, el contenido del historial se mantiene definido a través de la reducción.

Con tal información, se procede a definir las reglas de reducción o modelo operacional de λ_q.

A.2.4 Reglas de reducción de λ_q

De acuerdo al modelo operacional y lo descrito anteriormente se definen las reglas de reducción.

Teorema A.1. *En el cálculo cuántico λ_q, la evolución del registro computacional es governado por las reglas de reducción dadas en la Tabla A.1.*

$\dfrac{t_1 \ \rightarrow t'_1}{(t_1 \ t_2) \ \rightarrow \ (t'_1 \ t_2)}$	*(app_1)*
$\dfrac{t_2 \ \rightarrow \ t'_2}{(v_1 \ t_2) \ \rightarrow \ (v_1 \ t'_2)}$	*(app_2)*
$((\lambda x.t) \ v) \ \rightarrow \ t[v/x]$	*(β)*
$((\lambda !x.t) \ !t') \ \rightarrow \ t[t'/x]$	*($!\beta$)*
$(c_U \ \phi) \ \rightarrow \ U \, \lvert\phi\rangle$	*(U)*

Tabla A.2: Reglas de reducción para el cálculo cuántico λ_q.

Este teorema provee un conjunto de reglas de reducción simple y que nos permite razonar sobre los cómputos sin tener que acarrear el historial. Estas reglas se comportan de manera similar al modelo operacional de λ_q (Tabla A.2.3).

Continuando, en λ_q, la teoría ecuacional define una noción de igualdad que es compatible con las reglas de reducción. Ésta se define enseguida.

A.2.5 Teoría ecuacional de λ_q

Previo a definir la teoría, se aborda rápidamente los conceptos de reducción y contexto de término.

Definición A.10 (Reducción)**.** La *reducción* puede considerarse como una operación algebraica de remplazar subtérminos con subtérminos iguales, tomando en cuenta que no puede haber reducción dentro de las !-suspensiones.

Definición A.11. Contexto de término (term context). Son expresiones acotadas con el símbolo [ ], y con la siguiente estructura:

$$\text{C, C}_i ::= [\,] \mid (\text{t c}) \mid (\text{c t}) \mid (\lambda \text{x.c}) \mid (\lambda !\text{x.c}) \mid \textstyle\sum_i c_i C_i$$

Donde, la expresión $\sum_i c_i C_i$, denota una superposición de contextos congruentes (Definición A.9).

Observación 6. Notar que no hay contextos de la forma $!(\cdots [\,] \cdots)$, lo cuales no serán capaces de realizar sustituciones mientras se tenga el símbolo !.

Con estas definiciones, se hace posible establecer la teoría ecuacional.

Esta teoría ecuacional se puede concebir como un conjunto de axiomas y reglas de inferencia dadas por la Tabla A.2.5:

$t = t$	$(refl)$	$\dfrac{t_1 = t_2}{t_2 = t_1}$	(sym)
$\dfrac{t_1 = t_2 \;\; t_2 = t_3}{t_1 = t_3}$	$(trans)$	$\dfrac{t_1 = t_2 \;\; t_3 = t_4}{(t_1\ t_3) = (t_2\ t_4)}$	(app)
$\dfrac{t_1 = t_2}{\lambda x.t_1 = \lambda x.t_2}$	(λ_1)	$\dfrac{t_1 = t_2}{\lambda !x.t_1 = \lambda !x.t_2}$	(λ_2)
$(\lambda x.t)\ v = t[v/x]$	(β)	$(\lambda !x.t)\ !t' = t[t'/x]$	$(!\beta)$
$\lvert(c_u\ \phi)\rangle = U\,\lvert\phi\rangle$	(U)		

Teoría ecuacional para λ_q.

De lo cual se concluye que:

Teorema A.2. *En el Cálculo Lambda Cuántico, la evolución del registro computacional se da reemplazando términos por términos iguales de acuerdo a la teoría ecuacional de la Tabla A.2.5.*

Para finalizar la sección, se concluye que el cálculo lámbda cuántico aporta un historial de operaciones, un gramática, modelo operacional y teoría ecuacional que incluye recursos cuánticos de manera adecuada y formal, permitiendo fundamentar que será una base para la investigación actual.

B
Otras definiciones

Definición B.1 (Variable limitada)**.** El conjunto de variables límite están definidas inductivamente por la siguiente función [25]: $BV : \Lambda \to \delta(Var)^3$. Dada por las reglas:

$$\begin{aligned} BVx &= \oslash \\ BV(\lambda x M) &= (BV\ M) \cup \{x\} \\ BV(M\ N) &= (BV\ M) \cup (BV\ N) \end{aligned}$$

Definición B.2 (Variable libre)**.** El conjunto de variables libres en un término están definidas inductivamente por la siguiente función [25]: $FV : \Lambda \to \delta(Var)$. Dada por las reglas:

$$\begin{aligned} FVx &= \oslash\{x\} \\ FV(\lambda x M) &= (FV\ M) - \{x\} \cup \\ FV(M\ N) &= (FV\ M) \cup (FV\ N) \end{aligned}$$

Definición B.3 (Call by value)**.** Método de reducción en el cual las funciones sólo se pueden aplicar a valores, es decir, los argumentos para una función son evaluados primero, y el resultado se pasa dentro de la función. Por ejemplo, si se tiene $(\lambda x.t)\ t'$, la reducción se realiza siempre y cuando t' sea un valor.

C
Teoría de Categorías

En este apéndice se encontrará información referente a *teoría de categorías*, abordando los conceptos requeridos en esta investigación.

C.0.1 Teoría de categorías

Brevemente, una *categoría* se conforma por objetos y flechas con ciertas características entre sí. Las propiedades de una categoría son:

Objeto: Es un conjunto o colección finita de elementos.

Morfismo (Flecha): Un morfismo es un proceso de ir de un objeto a otro. Éste consiste de tres elementos:

$$\underbrace{A}_{dominio} \longrightarrow \underbrace{B}_{codominio} \tag{C.1}$$

1. Un objeto A, llamado el *dominio* del morfismo.
2. Un objeto B, llamado el *codominio* del morfismo.
3. Una regla que le asigna a cada elemento $a \in A$ un elemento $b \in B$. Esta b se denota formalmente como $f(a)$.

1. Ley de la identidad: Si $A \xrightarrow{f} B$ entonces $1_B \circ f = f$ y $f \circ 1_A = f$.

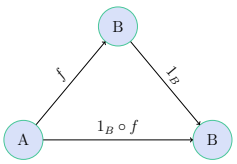


Figura C.1: $1_B \circ f = f$.

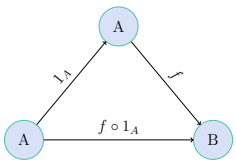


Figura C.2: $f \circ 1_A = f$.

2. Ley asociativa: Si $A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$, entonces $(h \circ g) \circ f = h \circ (g \circ f)$.

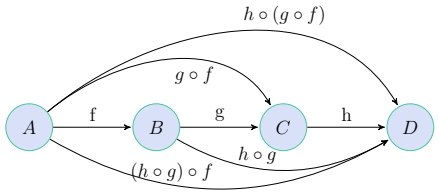


Figura C.3: Composición de una categoría.

A continuación se definen conceptos referentes a Tripletas de Kleisli y Monadas.

C.0.2 Tripleta de Kleisli

Definición C.1. Una tripleta de Kleisli sobre una categoría $\mathcal{C}$ es $(T, \eta, \_^*)$, donde:

- $T : \text{Obj}(\mathcal{C}) \to \text{Obj}(\mathcal{C})$
- $\eta : A \to T\ A$, para $A \in \text{Obj}(\mathcal{C})$
- $f^* : T\ A \to T\ B$, para $f : A \to T\ B$

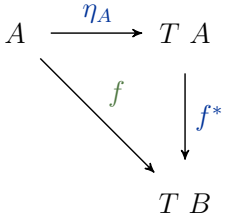
Que satisface lo siguiente:

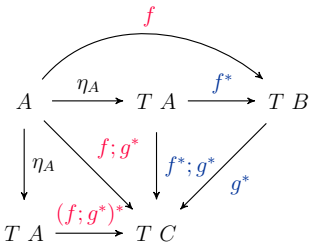
a) $\eta_A^* = id_{TA}$.

$$A \xrightarrow{\eta_A} T\ A \xrightarrow{\eta_A^*} T\ A \xleftarrow[\text{Id}_{T\ A}]{} T\ A$$

b) $\eta_A; f^* = f$, para $f : A \to T\ B$.

c) $f^*; g^* = (f; g^*)^*$, para $f : A \to T\ B$ y $g : B \to T\ C$.



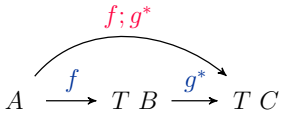


C.0.3 Categoría de Kleisli

Definición C.2. Dada una Tripleta de Kleisli $(T, \eta, \_^*)$ sobre una categoría C, una Categoría de Kleisli C_T está definida como:

- Los objetos de C_T son los de C.

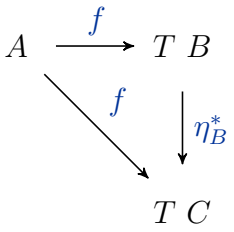
- El conjunto C_T (A,B) de morfismos de A a B, en C son C(A, T B), es decir: $C_T(A, B) = C(A, TB)$.
- La identidad sobre A en C_T es: $\eta_A : A \to T\ A$.
- Sean $f \in C_T(A, B)$ seguido de $g \in C_T(B, C)$, la *composición* en C_T está dada por: $f; g^* : A \to TC$.



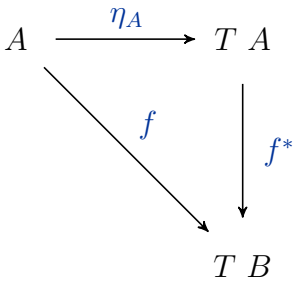
La tripleta de Kleisli tiene tres axiomas, los cuales en la categoría C_T se definen como:

- Ley de la identidad o unitariedad:

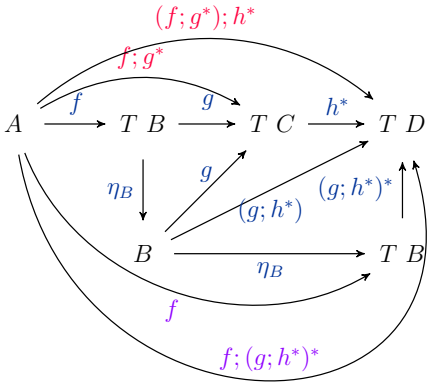
$f; \eta_B^* = f$, para $f : A \to T\ B$



$\eta_A; f^* = f$, para $f : A \to T\ B$



- Ley asociativa: $(f; g^*); h^* = f; (g; h^*)^*$. Para $f : A \to TB, g : B \to TC$ y $h : C \to TD$



Comparativa Tripleta de Kleisli vs Monada

Tripleta de Kleisli	Categoría de Kleisli C_T
$T : Obj(C) \to Obj(C)$ $\eta_A = A \to T\ A$, para $A \in Obj(C)$	Los objetos de C_T son los de C $\eta_A : A \to T\ A$ $C_T(A, B) = C(A, TB)$
$f^* : T\ A \to T\ B$, para $f : A \to T\ B$	$f^* : T\ A \to T\ B$, para $f : A \to B$
Satisfaciendo: $\eta_A^* = Id_{TA}$ $\eta_A; f^* = f$	$f; \eta_B^* = f$ $\eta_A; f^* = f$
$f^*; g^* = (f; g^*)^*$	$(f; g^*); h^* = f; (g; h^*)^*$

C.0.4 Monada

Definición C.3. Una monada sobre una categoría C es un tripleta (T, η, μ), donde

- T se define como: $T : C \longrightarrow C$.

Dos transformaciones:

- Unidad $\eta : Id_C \to T$. Con $Id_A : A \longrightarrow A$, donde $A \in C$, se tiene $\eta_A : A \to T\ A$.
- Multiplicación: $\mu : T^2 \to T$. Esto es: $\mu : T\ T \to T$.

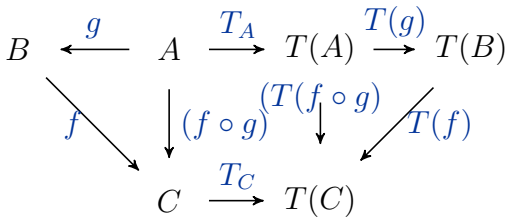
Satisfaciendo:

i) $T(Id_A) = Id_{TA}$, para $A \in Obj(\mathcal{C})$

$$T(A) \underset{Id_{T(A)}}{\overset{T(Id_A)}{\rightrightarrows}} T(A)$$

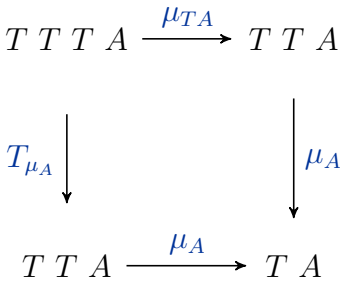
ii) Composición: Sean $f, g \in \mathcal{C}$, donde $f : A \to B$ y $g : B \to C$.

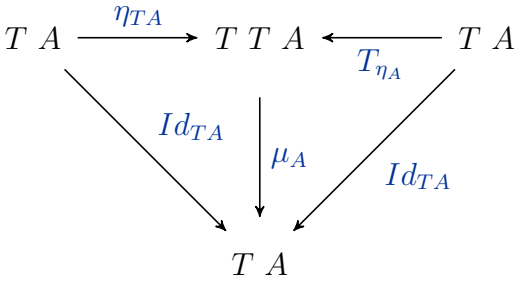
$$T(f \circ g) = T(f) \circ T(g)$$



También con:

- $\mu_{TA}; \mu_A = T_{\mu_A}; \mu_A$
- $\eta_{TA}; \mu_A = Id_{TA} = T_{\eta_A}; \mu_A$



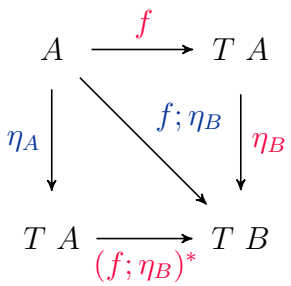


C.0.5 Asociación Tripleta de Kleisli y Monada

Proposición 1. Hay una correspondencia uno a uno entre las tripletas de Kleisli y Monadas.

C.0.6 Generación de un Monada dada una Tripleta de Kleisli

- $T : C \to C$
  Esta dada por: $Tf = (f; \eta_B)^* : TA \to TB$.



- $\mu_A : T^2 A \to T\ A$.
  Se define como: $\underbrace{\mu_A}_{Monada} = \underbrace{(Id_{TA})^*}_{T.Kleisli} = (T\ A \to T\ A)^* : TTA \to TA$.

- $\eta : Id_C \to T$

$$\overbrace{\eta_A}^{Monada} = \overbrace{\eta_A}^{T.Kleisli}$$

$$\eta_A : A \to T\ A$$

Que satisfacen: $\eta_{TA}; \mu_A = Id_{TA} = T_{\eta_A}; \mu_A$ y $\mu_{TA}; \mu_A = T_{\mu_A}; \mu_A$

C.0.7 Generación de una Tripleta de Kleisli dada una Monada

Dada una monada (T, η, μ) sobre una categoría C, la Tripleta de Kleisli $(T, \eta, \_^*)$ sobre C, se define como:

- $T : Obj(C) \to Obj(C)$
- Para $A \in Obj(C)$, $\underbrace{\eta_A}_{T.Kleisli} = \underbrace{\eta_A}_{Monada} = (Id_A \longrightarrow T\ A) : A \to T\ A$.
- Para obtener f^*. Sea f una función: $f : A \to T\ B$, entonces f^*, se define como: $f^* : T\ A \longrightarrow^{Tf} T\ T\ B \longrightarrow^{\mu_B} T\ B$, es decir: $f^* = (Tf; \mu_B) : T\ A \longrightarrow T\ B$

Satisfaciendo: $\eta_A; f^* = f$, para $f : A \to TB$; $\eta_A; f^* = f$, para $f : A \to TB$; y $f^*; g^* = (f; g^*)^*$, para $f : A \to TB$, $g : B \to TC$.

C.0.8 Otras definiciones

Definición C.4 (Functor)**.** Sean $\mathcal{B}$ y $\mathcal{C}$ categorías. Un *functor* $F : \mathcal{B} \to \mathcal{C}$ que relaciona dos categorías, está dado por:

- Una función, denotada también como:

  $$F : Obj(\mathcal{B}) \to Obj(\mathcal{C})$$

  Que asigna a cada objeto B de $\mathcal{B}$ un objeto $F(B)$ en $\mathcal{C}$.

- Para cada par de objetos X, Y de $\mathcal{B}$, y un morfismo $f : X \to Y$, se define:

  $$F : Mor_{\mathcal{B}}(X, Y) \to Mor_{\mathcal{C}}(F(X), F(Y))$$

  Es decir, asigna a cada morfismo $f : X \to Y$ se tiene $F(f) : F(X) \to F(Y)$. Visto como:

  $$f \longmapsto F(f)$$

Que cumple las siguientes condiciones:

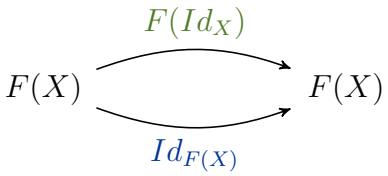
i) **Para todo objeto** $X \in Obj(\mathcal{B})$

$$F(Id_X) = Id_{F(X)}$$

Si $Id_X : X \to X$, entonces por definición se tiene:

$$F(Id_X) : F(X) \to F(X)$$

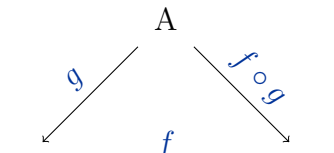
Por lo tanto:



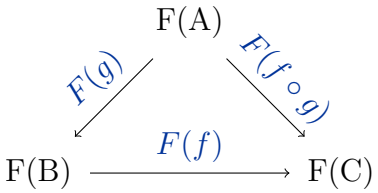
ii) Composición dados los morfismos $f, g \in \mathcal{B}$, donde $g : A \to B$, $f : B \to C$. Se define como:

$$F(f \circ g) = F(f) \circ F(g)$$

Se tiene:



Con functores:



Transformación

Sean $\mathcal{B}$ y $\mathcal{C}$ categorías, y $F, G : \mathcal{B} \to \mathcal{C}$ functores. Una *transformación natural* es:

$$\alpha : F \to G$$

que asigna a cada objeto $X \in \mathcal{B}$ un morfismo:

$$\alpha_X : F(X) \to G(X) \text{ en } \mathcal{C},$$

tal que para cada morfismo $f : X \to Y$ de $\mathcal{B}$, el siguiente diagrama conmuta:

$$\begin{array}{ccc} F(X) & \xrightarrow{\alpha_X} & G(X) \\ \downarrow{\scriptstyle F(f)} & & \downarrow{\scriptstyle G(f)} \\ F(Y) & \xrightarrow{\alpha_Y} & F(Y) \end{array}$$

D

Compuertas cuánticas

H Matriz de Walsh-Hadamard $W = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$.

I Matriz identidad $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

S Compuerta de fase, cambia: $\alpha |0\rangle + \beta |1\rangle$ por $\alpha |0\rangle + \beta i |1\rangle$.

X Compuerta X o inversor $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Y Compuerta de Paulli, cambia: $\alpha |0\rangle + \beta |1\rangle$ por $\beta |0\rangle - \alpha |1\rangle$.

C(X) Compuerta controlada X, aplica para cualquier compuerta.

Z Matriz de cambio de fase.

XOR Compuerta OR exclusivo.

T Transformada de Fourier Cuántica.

cnot Negación controlada. $C(X) |ab\rangle = |a\rangle |b \oplus a\rangle$,

Nota D.1. Tabla de verdad del operador clásico XOR:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0