University of Minho
School of Engineering

Bruna Filipa Martins Salgado

A Metric Equational System for Quantum Computation

May 2025

University of Minho
School of Engineering

Bruna Filipa Martins Salgado

A Metric Equational System for Quantum Computation

Master's Dissertation
Master in Physics Engineering

Work carried out under the supervision of
Renato Jorge Araújo Neves

May 2025

Copyright and Terms of Use for Third Party Work

License granted to users of this work:

Acknowledgements

Write your acknowledgements here. Do not forget to mention the projects and grants that you have benefited from while doing your research, if any. Ask your supervisor about the specific textual format to use. (Funding agencies are quite strict about this.)

Statement of Integrity

I hereby declare having conducted this academic work with integrity.
I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.
I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, May 2025

Bruna Filipa Martins Salgado

Abstract

Noisy intermediate-scale quantum (NISQ) computers are expected to operate with severely limited hardware resources. Precisely controlling qubits in these systems comes at a high cost, is susceptible to errors, and faces scarcity challenges. Therefore, error analysis is indispensable for the design, optimization, and assessment of NISQ computing. Nevertheless, the analysis of errors in quantum programs poses a significant challenge. The overarching goal of the M.Sc. project is to provide a fully-fledged quantum programming language on which to study metric program equivalence in various scenarios, such as in quantum algorithmics and quantum information theory.

Keywords approximate equivalence, λ-calculus, metric equations

Resumo

Escrever aqui o resumo (pt)

Palavras-chave palavras, chave, aqui, separadas, por, vírgulas

Contents

List of Figures

Acronyms

NISQ Noisy Intermediate-Scale Quantum 1, 2, 3

CPTP Completely Positive Trace-Preserving 25, 26, 28, 51, 52

OSR Operator Sum Representation 26

BNF Backus-Naur Form 39, 40

Notation

$\mathbb{C}^n$ n-dimensional complex space. 6

$\mathbb{C}^{n\times m}$ Space of complex matrices of dimension $n \times m$. 6

R, V, W Typical names for vector spaces. 6

dim (V) Dimension of vector space V. 7

$\langle\cdot,\cdot\rangle$ Inner product. 8

$(-)^*$ Complex conjugate operation. 9

$\|\cdot\|$ Norm of an arbitrary vector. 9

$d(x, y)$ distance between vectors x and y. 10

I Identity operator 11

$(-)^\dagger$ Adjoint operation. 11

U Typical name for a unitary operator. 12

ρ, σ Typical designations for density matrices. 12

$\|.\|_2$ Euclidean norm. 12

$\|.\|_1$ Trace norm. 13

$\|\cdot\|_\infty$ Spectral norm. 13

$\|.\|_\sigma$ Operator norm. 13

$(-)^{\otimes n}$ N-fold tensor product. 15

$|\psi\rangle$ Quantum state. Also known as ket. 15

$\langle\psi|$ $|\psi\rangle^\dagger$. Also known as bra. 15

$\langle\psi|\phi\rangle$ Inner product between states $|\psi\rangle$ and $|\phi\rangle$. 15

$|\psi\rangle \otimes |\phi\rangle$ Tensor product of states $|\psi\rangle$ and $|\phi\rangle$. 17

$|\psi\rangle\,|\phi\rangle$ Tensor product of states $|\psi\rangle$ and $|\phi\rangle$. 17

$|\psi\phi\rangle$ Tensor product of states $|\psi\rangle$ and $|\phi\rangle$. 17

X Pauli operator σ_x. 19

Z Pauli operator σ_z. 19

H Hadamard gate 20

P Phase-shift gate 20

CNOT Controlled Not gate 20

$\|\cdot\|_\diamond$ Diamond norm. 27

$\mathcal{M}\mathbb{R}$ Banach space of finite Borel measures on $\mathbb{R}$. 35

$FV(v)$ Set of free variables of a term v. 40

$v : \mathbb{A}$ Typed term. 41

Γ, Δ, E Typical names for typing contexts. 41

$\Gamma \triangleright v : \mathbb{A}$ Typing judgement. 41

$v[w/x]$ Substitution of a variable x for a term w in a term v. 44

$\Gamma \triangleright v = w : \mathbb{A}$ Equation-in-context. 46

$t =_\epsilon s$ Metric equation. 47

Chapter 1

Introduction

1.1 Motivation and Context

Quantum computing dates back to 1982 when Nobel laureate Richard Feynman proposed the idea of constructing computers based on quantum mechanics principles to efficiently simulate quantum phenomena [1].

The field has since evolved into a multidisciplinary research area that combines quantum mechanics, computer science, and information theory. Quantum information theory, in particular, is based on the idea that if there are new physics laws, there should be new ways to process and transmit information. In classical information theory, all systems (computers, communication channels, etc.) are fundamentally equivalent, meaning they adhere to consistent scaling laws. These laws, therefore, govern the ultimate limits of such systems. For instance, if the time required to solve a particular problem, such as the factorization of a large number, increases exponentially with the size of the problem, this scaling behavior remains true irrespective of the computational power available. Such a problem, growing exponentially with the size of the object, is known as a ”difficult problem”. However, as demonstrated by Peter Shor, the use of a quantum computer with a sufficient number of quantum bits (qubits) could significantly accelerate the factorization of large numbers [2]. This advancement poses a significant threat to the security of confidential data transmitted over the Internet, as the RSA algorithm is based on the computational difficulty of factorizing large numbers.

The quantum computing paradigm holds immense promise, as evidenced by this compelling result in computational complexity theory. While hardware advancements have brought the scientific community closer to realizing this potential, the ultimate goal is yet to be accomplished. A **Noisy Intermediate-Scale Quantum** (**NISQ**) computer equipped with 50-100 qubits may surpass the capabilities of current classical computers, yet the impact of quantum noise, such as decoherence in entangled states, imposes limitations on the size of quantum circuits that can be executed reliably [3]. Unfortunately, general-purpose error correction techniques [4–6] consume a substantial number of qubits, making it difficult for **NISQ** devices to make use of them in the near term. For instance, the implementation of a single logical qubit may require between 10^3 and 10^4 physical qubits [7]. As a result, it is unreasonable to expect that the idealized quantum algorithm will run perfectly on a quantum device, instead only a mere approximation will be observed.

To reconcile quantum computation with **NISQ** computers, quantum compilers perform transformations for error mitigation [8] and noise-adaptive optimization [9]. Additionally, current quantum computers only support a restricted, albeit universal, set of quantum operations. As a result, nonnative operations must be decomposed into sequences of native operations before execution [10], [11]. In general, perfect computational universality is not sought, but only the ability to approximate any quantum algorithm, with a preference for minimizing the use of additional gates beyond the original requirements. The assessment of these compiler transformations necessitates a comparison of the error bounds between the source and compiled quantum programs. Additionally, in quantum information theory, it is essential to account for errors arising from malicious attacks or noisy channels [12].

This suggests the development of appropriate notions of approximate program equivalence, *in lieu* of the classical program equivalence and underlying theories that typically hinge on the idea that equivalence is binary, *i.e.* two programs are either equivalent or they are not [13].

As previously noted, Shor's algorithm has played a pivotal role in sparking heightened interest within the scientific community toward quantum computing research. Several quantum programming languages have surfaced over the past 25 years [14, 15]. These include imperative languages such as Qiskit [16] and Silq [17], as well as functional languages such as Quipper [18] and Q# [19]. On one hand, the design of quantum programming languages is strongly oriented towards implementing quantum algorithms. On the other hand, the definition of functional paradigmatic languages or functional calculi serves as a valuable tool for delving into theoretical aspects of quantum computing, particularly exploring the foundational basis of quantum computation [20].

Most of the current research on algorithms and programming languages assumes that addressing the challenge of noise during program execution will be resolved either by the hardware or through the implementation of fault-tolerant protocols designed independently of any specific application [21]. As previously stated, this assumption is not realistic in the **NISQ** era. Nonetheless, there have been efforts to address the challenge of approximate program equivalence in the quantum setting.

[22] and [23] reason about the issue of noise in a quantum while-language by developing a deductive system to determine how similar a quantum program is from its idealised, noise-free version. The former introduces the (Q,λ)-diamond norm which analyzes the output error given that the input quantum state satisfies some quantum predicate Q to degree λ. However, it does not specify any practical method for obtaining non-trivial quantum predicates. In fact, the methods used in [[22]] cannot produce any post conditions other than $(I, 0)$ (*i.e.*, the identity matrix I to degree 0, analogous to a "true" predicate) for large quantum programs. The latter specifically addresses and delves into this aspect.

An alternative approach was explored in [24], using linear λ-calculus as basis – *i.e* programs are written as linear λ-terms – which has deep connections to both logic and category theory [25], [26]. A notion of approximate equivalence is then integrated in the calculus via the so-called diamond norm , which induces a metric (roughly, a distance function) on the space of quantum programs (seen semantically as completely positive trace-preserving super-operators) [[12]]. The authors argue that their deductive system allows to compute an approximate distance between two quantum programs easily as opposed to computing an exact distance "semantically" which tends to involve quite complex operators. Some positive results were achieved in this setting, but much remains to be done.

Dar mais ênfase ao lambda calculus

1.2 Goals

1.3 Document Structure

Chapter 2

Preliminaries

Mudar esta descrição

The first two sections of this chapter present the mathematical and quantum computing preliminaries necessary for understanding the theory of quantum computation. It should be noted that the concept of a norm, as well as the properties of some norms, are relevant here, as the existence of a metric system implies that operators have a well-defined norm. The introduction to quantum computing is primarily based on [12, 27], while the mathematical foundations are also based on [28] and [29]. The subsequent section delves into how the terms of quantum lambda calculus, which are constructed using Figure 2, are interpreted in the "quantum world".

2.1 Linear Algebra

Tirar a descrição do que são vetores e coisas desse tipo, que eu não vou andar com este tipo de promenores para o resto

Tirar as normas que eu não precisar

Definir trace class operators e traço de maneira mais geral

It is impossible to present the theory of quantum computation without introducing some concepts of linear algebra within finite-dimensional spaces. This section provides a brief overview of the aspects of linear algebra that are most pertinent to the study of quantum computation.

2.1.1 Complex vector spaces

The basic objects of linear algebra are vector spaces. The vector spaces of interest in this work are the complex vector spaces, such as $\mathbb{C}^n$ and $\mathbb{C}^{n\times m}$ therefore, the following definition is given in the context of complex vector spaces.

Definition 2.1.1. A *vector space* (over $\mathbb{C}$) consists of a set V whose elements are called vectors, together with two operations:

- An operation called vector addition that takes two vectors $v, w \in V$, and results in a third vector, written $v + w \in V$;
- An operation called scalar multiplication that takes a scalar $a \in \mathbb{C}$ and a vector $v \in V$, and results in a new vector, written $r \cdot v \in V$

which satisfy the following conditions (called axioms), for all $u, v, w \in V$ and $a_1, a_2 \in \mathbb{C}$:

1. Vector addition is commutative: $u + v = v + u$;
2. Vector addition is associative: $(u + v) + w = u + (v + w)$;
3. There is a zero vector $0 \in V$ such that $v + 0 = v$ for all $v \in V$;
4. Each $v \in V$ has an additive inverse $w \in V$ such that $w + v = 0$;
5. Scalar multiplication distributes over scalar addition, $(a_1 + a_2) \cdot v = a_1 \cdot v + a_2 \cdot v$;
6. Scalar multiplication distributes over vector addition, $a_1 \cdot (v + w) = a_1 \cdot v + a_1 \cdot w$;
7. Ordinary multiplication of scalars associates with scalar multiplication, $(a_1a_2) \cdot v = a_1 \cdot (a_2 \cdot v)$;
8. Multiplication by the scalar 1 is the identity operation, $1 \cdot v = v$.

The letters R, V, W will often be used to refer to vector spaces. All vector spaces are assumed to be finite dimensional, unless otherwise noted.

In the case of the vector space $\mathbb{C}^n$, the space of all n-tuples of complex numbers, $(a_1, \ldots, a_n)$, addition and scalar multiplication are defined in the following standard way:

- Addition: for vectors $u = (a_1, \ldots, a_n), v = (b_1, \ldots, b_n) \in \mathbb{C}^n$, the vector $u + v \in \mathbb{C}^n$ is defined by the equation $(u + v) = (a_1 + b_1, \ldots, a_n + b_n)$.

- Scalar multiplication: for a scalar $a \in \mathbb{C}$ and a vector $v = (b_1, \ldots, b_n) \in \mathbb{C}^n$, the vector $a \cdot v \in \mathbb{C}^n$ is defined by the equation $a \cdot v = (a \cdot b_1, \ldots, a \cdot b_n)$.

The zero vector in $\mathbb{C}^n$ is the vector with all entries equal to zero.
Sometimes, the column matrix notation is used to represent vectors in $\mathbb{C}^n$, that is, a vector $v \in \mathbb{C}^n$ is written as a column matrix

$$v = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}.$$

Other times for readability the format $(a_1, \ldots, a_n)$ is used. The latter should be interpreted as a shorthand for a column vector.
$\mathbb{C}^{n\times m}$ is a vector space when addition is defined as matrix addition and scalar multiplication is defined as multiplication of each component by the scalar. The zero vector is the zero matrix, *i.e.*, the matrix with all entries equal to zero.

Definition 2.1.2. A *vector subspace* of a vector space V is a subset W of V such that W is also a vector space, that is, W must be closed under scalar multiplication and addition.

Definition 2.1.3. A *spanning set* of a vector space is a set of vectors $v_1, \ldots, v_n$ such that any vector v in the vector space can be written as a linear combination $v = \sum_i a_i v_i$ of vectors in that set, where a_i are scalars.

Definition 2.1.4. A set of non-zero vectors $v_1, \ldots, v_n$ are *linearly dependent* if there exists a set of complex numbers $a_1, \ldots, a_n$ with $a_i \neq 0$ for at least one value of i, such that

$$a_1 v_1 + a_2 v_2 + \ldots a_n v_n = 0.$$

Definition 2.1.5. A set of vectors $v_1, \ldots, v_n$ are *linearly independent* if they are not linearly dependent.

Definition 2.1.6. A *basis* for a vector space is a sequence of vectors that is linearly independent and that spans the space.

Definition 2.1.7. The number of elements in the basis is defined to be the *dimension* of V, denoted **dim** (V).

2.1.2 Linear operators

A *linear operator* between vector spaces V and W is defined to be any function $A : V \to W$ which is linear in its inputs, *i.e.*

$$A\left(\sum_i a_i v_i\right) = \sum_i a_i A(v_i)$$

Usually $A(v)$ is just denoted Av.

Suppose $V, W,$ and R are vector spaces, and $A : V \to R$ and $B : W \to R$ are linear operators. Then the notation BA is used to denote the *composition* of B with A, defined by $(BA)(v) \equiv B(A(v))$. Once again, $(BA)(v)$. is abbreviated as BAv.

For any choice of complex spaces $V \in \mathbb{C}^n$ and $W \in \mathbb{C}^m$, there is a bijective linear correspondence between the set of operators from V to W and the set of $n \times m$ matrices. The claim that the matrix $A \in \mathbb{C}^{m\times n}$ is a linear operator just means that

$$A\left(\sum_i a_i v_i\right) = \sum_i a_i A(v_i)$$

is true as an equation where the operation is matrix multiplication of A by a collumn vector in $\mathbb{C}^n$. Clearly, this is true! On the other hand, suppose $A : V \to W$ is a linear operator between vector spaces V and W, such that $V \in \mathbb{C}^n$ and $W \in \mathbb{C}^m$. Suppose $v_1, \ldots, v_n$ is a basis for V and $w_1, \ldots, w_n$ is a basis for W. Then for each j in the range $1, \ldots, m$, there exist complex numbers A_{1j} through A_{nj} such that

$$Av_j = \sum_i A_{ij} w_i.$$

The matrix whose entries are the values A_{ij} is said to form a *matrix representation* of the operator A. This matrix representation of A is completely equivalent to the operator A. As a result, when considering operators on vector spaces of the form $\mathbb{C}^n$ it is common to refer to the operator A and its matrix representation interchangeably.

2.1.3 Inner product

Definition 2.1.8. The *inner product* $\langle\cdot,\cdot\rangle$ is a function from a vector space V to the complex numbers, $\langle\cdot,\cdot\rangle : V \times V \to \mathbb{C}$, that satisfies the following properties for all $v, w \in V$ and $r \in \mathbb{C}$:

1. Linearity in the second argument,

$$\left\langle v, \sum_i a_i w_i \right\rangle = \sum_i a_i \langle v, w_i \rangle.$$

2. $\langle v, w \rangle = \langle w, v \rangle^\dagger$, where $(-)^*$ is the complex conjugate operation.

3. $\langle v, w \rangle \geq 0$ with equality if and only if $v = 0$.

The inner product $\langle v, w \rangle$ of two vectors $v = (a_1, \dots, a_n), w = (b_1, \dots, b_n) \in \mathbb{C}^n$ is defined as

$$\langle v, w \rangle = \sum_i a_i^* b_i.$$

Trace

In order to define inner product of a matrix, it is necessary to first define the trace of a matrix. The trace of a square matrix $A \in \mathbb{C}^{n \times n}$ defined to be the sum of its diagonal elements,

$$\text{tr}(A) = \sum_i A_{ii}.$$

The trace is *cyclic*, that is, $\text{tr}(AB) = \text{tr}(BA)$, and *linear*, $\text{tr}(A+B) = \text{tr}(A)+\text{tr}(B)$, $\text{tr}(a \cdot A) = a \cdot \text{tr}(A)$, where matrices $A, B \in \mathbb{C}^{n \times n}$, and a is a complex number.
By means of the trace, one defines the inner product of two operators $A, B \in \mathbb{C}^{m \times n}$ as follows

$$\langle A, B \rangle = \text{tr}(A^\dagger B). \tag{2.1}$$

In the *finite* dimensional complex vector spaces relevant to quantum computation and quantum information, a *Hilbert space* is is equivalent to an inner product space. As a result, both $\mathbb{C}^n$ and $\mathbb{C}^{n \times m}$ are Hilbert spaces.

2.1.4 Norm and normed spaces

Definition 2.1.9. A *norm* $\|\cdot\|$ is a function that associates an element of a vector space V with a non-negative real number, such that the following properties hold:

1. Positive definiteness: $\|v\| \geq 0$ for all $v \in V$, with $\|v\| = 0$ if and only if $v = 0$;

2. Positive scalability: $\|av\| = |a|\|v\|$ for all $v \in V$ scalar a;

3. The triangle inequality: $\|v + w\| \leq \|v\| + \|w\|$ for all $v, w \in V$.

Definition 2.1.10. A vector space together with a norm is called a *normed vector space*.

Every normed space may be regarded as a metric space, in which the distance $d(x, y)$ between vectors x and y is $\|x - y\|$. The relevant properties of d are

1. $0 < d(x, y) < \infty$ for all x and y,
2. $d(x, y) = 0$ if and only if $x = y$,
3. $d(x, y) = d(y, x)$ for all x and y,
4. $d(x, z) < d(x, y) + d(y, z)$ for all x, y, z.

Every inner product space is a normed space, where the norm of a vector v is defined as $\|v\| = \sqrt{\langle v, v\rangle}$.

Definition 2.1.11. Two vector u, v are said to be *orthogonal* if $\langle v, u\rangle$. An *orthogonal set* is a set of orthogonal vectors of the same vector space.

Definition 2.1.12. A *unit vector* is a vector v such that $\|v\| = 1$. It is also said that v is *normalized* if $\|v\| = 1$

Definition 2.1.13. An orthogonal set of unit vectors is called an *orthonormal set*, and when such a set forms a basis it is called an *orthonormal basis*.

2.1.5 Eigenvectors and eigenvalues

Definition 2.1.14. An *n-permutation* is a function on the first n positive integers $\pi = \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ that is one-to-one and onto. In a permutation each number $1, \ldots, n$ appears as output for one and only one input.

Definition 2.1.15. The *sign* of a permutation sgn(π) is -1 if the number of inversions in π is odd and is $+1$ if the number of inversions is even.

Definition 2.1.16. The *determinant* of a square matrix $A \in \mathbb{C}^{n\times n}$ is defined as

$$\det(A) = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^{n} A_{i\sigma(i)},$$

Here S_n is the set of all n-permutations $\pi = \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$, and sgn$(\pi)$ denotes the sign of the permutation π.

Definition 2.1.17. An *eigenvector* of a linear operator A on a vector space is a non-zero vector v such that $Av = \lambda v$, where λ is a complex number known as the *eigenvalue* of A corresponding to v.

The *characteristic polynomial* of a square operator A is the polynomial $p(\lambda) = \det(A - \lambda I)$, where I is the identity operator

$$\begin{aligned} I_V :& V \to V \\ & v \mapsto v. \end{aligned}$$

where V is a vector space. The subscript will be omitted unless ambiguity arises. It can be shown that the characteristic function depends only upon the operator A, and not on the specific matrix representation used for A. By the fundamental theorem of algebra, every polynomial has at least one complex root, so every operator A has at least one eigenvalue, and a corresponding eigenvector. The solutions of the *characteristic equation* $c(\lambda) = 0$ are the eigenvalues of the operator A.

A *diagonal representation* of an operator A on a vector space V is an expression of the form $A = \sum_i \lambda_i v_i v_i^\dagger$, where the vectors v_i form an orthonormal set of eigenvectors for A, with corresponding eigenvalues λ_i, and $(-)^\dagger$ is the adjoint operation.

2.1.6 Spectral theorem

Theorem 2.1.18. *[27] Every normal operator $A \in \mathbb{C}^{n\times n}$ can be expressed as a linear combination $\sum_i \lambda_i b_i b_i^\dagger$ where the set $\{b_i, \dots, bn\}$ is an orthonormal basis on $\mathbb{C}^n$.*

Using this last result any function $f : \mathbb{C} \to \mathbb{C}$, can be extended to normal matrices via,

$$f(A) = \sum_i f(\lambda_i) b_i b_i^\dagger \tag{2.2}$$

where $A = \sum_i \lambda_i b_i b_i^\dagger$ is the spectral decomposition of A.

2.1.7 Important classes of operators/matrices

Linear operators mapping a complex space $\mathbb{C}^n$ or $\mathbb{C}^{n\times n}$ to itself will be called *square operators* due to the fact that their matrix representations are square matrices. Therefore, those definitions given in the context of square operators are also valid for square matrices.

The following classes of operators are of particular interest in quantum information theory.

Definition 2.1.19. *Normal operators.* A square operator A is *normal* if $AA^\dagger = A^\dagger A$.

Definition 2.1.20. *Hermitian operators.* A square operator A is *hermitian* if $A = A^\dagger$. Every Hermitian operator is a normal operator.

Definition 2.1.21. *Positive (semidefinite) operators*. A square operator $A : \mathbb{C}^n \to \mathbb{C}^{\ltimes}$ is *positive*, denoted $A \geq 0$, if $\langle v, Av \rangle \geq 0$ for all $v \in \mathbb{C}^n$. The sum of two positive semidefinite operators is positive semidefinite. Positive matrices are hermitian, and consequently, by the spectral decomposition, have diagonal representation $A = \sum_i \lambda_i v_i v_i^\dagger$, with non-negative eigenvalues λ_i [27].

Definition 2.1.22. *Unitary operators.* A square operator U is *unitary* if $U^\dagger U = UU^\dagger = I$. The letter U will often be used to refer to unitary operators.

Geometrically, unitary operators are important because they preserve inner products between vectors, $\langle Uv, Uw \rangle = \langle v, w \rangle$ for any two vectors v and w. Let $S_1 = \{v_i\}$ be any orthonormal basis set. Define $s_2 = \{w_i\} = \{Uv_i | v_i \in S\}$, so S_2 is also an orthonormal basis set, since unitary operators preserve inner products. Note that $U = \sum_i w_i v_i^\dagger$. Conversely, if $\{v_i\}$ are any two orthonormal bases, then it is easily checked that the operator U defined by $U = \sum_i w_i^\dagger v_i$ is unitary.

Definition 2.1.23. *Density operator*. Positive semidefinite operators that have trace equal to 1 are called *density operators*. Lowercase Greek letters, such as ρ, σ are conventionally used to denote density operators.

Definition 2.1.24. *Isometries*. An operator $A : \mathbb{C}^n \to \mathbb{C}^m$ is as isometry if $||Av|| = ||v||$ for all elements all elements $v \in \mathbb{C}^n$.

Definition 2.1.25. *Projectors*. A positive semidefinite operator P is a projector if $P^2 = P$. Equivalently, a projection operator is a Hermitian operator whose only eigenvalues are 0 and 1. The *orthogonal complement* of P is the operator $Q = I - P$.

Definition 2.1.26. *Diagonal operators*. A square operator A is *diagonal* if $A_{ij} = 0$ for all $i \neq j$.

2.1.8 Useful norms

Definition 2.1.27. The *euclidean norm*, $||.||_2$, of a vector $v \in \mathbb{C}^n$ is defined as:

$$||v||_2 = \sqrt{\langle v, v \rangle}.$$

Definition 2.1.28. The *trace norm*, $\|.\|_1$, of a matrix A is defined as:

$$\|A\|_1 = \text{tr}\sqrt{A^\dagger A} \tag{2.3}$$

This norm is also known as the Schatten 1-norm. The trace norm induces a metric on the set of density matrices which is defined by $d(\rho, \sigma) = \|\rho - \sigma\|_1$.

Alternatively, the trace norm of a square matrix $A \in \mathbb{C}^{n\times n}$ can be defined in the following way:

$$\|A\|_1 = \max\{|\langle U, A\rangle| \mid U \in \mathbb{C}^{n\times n} \text{ is a unitary operator}\}. \tag{2.4}$$

Proposition 2.1.29. *For all square matrices A and B, this norm enjoys the following properties:*

- *Submultiplicativity with respect to compositions: $\|AB\|_1 \leq \|A\|_1\|B\|_1$;*
- *Multiplicativity with respect to the tensor products: $\|A \otimes B\|_1 = \|A\|_1\|B\|_1$.*
- *Isometric invariance (and therefore unitarily invariance): for any unitary operators U_0 and U_1, $\|UAU_1^\dagger\|_1 = \|A\|_1$.*

Definition 2.1.30. The *spectral norm*, $\|\cdot\|_\infty$, of an operator $A : \mathbb{C}^n \to \mathbb{C}^m$ is defined as:

$$\|A\|_\infty = \max\{\|Av\|_2 \mid \|v\|_2 = 1\}.$$

Definition 2.1.31. Two norms on operators, $\| - \|_p$ and $\| - \|_{p*}$, are said to be *dual*, when for every operator $A : \mathbb{C}^n \to \mathbb{C}^m$, it holds that

$$\|A\|_p = \max\{|\langle B, A\rangle| \mid B : \mathbb{C}^n \to \mathbb{C}^m, \|B\|_{p*} = 1\}.$$

The trace norm and the spectral norm are dual.

Definition 2.1.32. Given normed vector spaces V and W, the operator,$\|.\|_\sigma$, norm for an operator $E : V \to W$ is defined as

$$\|E\|_\sigma = \sup\{\|E(A)\| \mid \|A\| = 1\}.$$

2.1.9 Tensor Products and Direct Sums of Complex spaces

Definition 2.1.33. The *direct sum* of two vector spaces V and W, denoted $V \oplus W$, is the space of all pairs (v, w) where $v \in V$ and $w \in W$.

The inner product in $V \oplus W$ is defined as follows:

$$\langle (v_1, w_1), (v_2, w_2) \rangle = \langle v_1, v_2 \rangle + \langle w_1, w_2 \rangle.$$

Definition 2.1.34. Consider two finite complex spaces spaces V and W with respective bases $(e_1, \ldots, e_n)$ and $(f_1, \ldots, f_k)$. The tensor product of V and W, denoted $V \otimes W$, is defined as the space generated by the basis of syntactic symbols:

$$(e_1 \otimes f_1, \ldots, e_1 \otimes f_k, \ldots, e_n \otimes f_1, \ldots, e_n \otimes f_k).$$

The tensor product of two elements $v = \sum_i \lambda_i \cdot e_i$ and w = $\sum_j \mu_j \cdot f_j$ is:

$$v \otimes w = \sum_{i,j} \lambda_i \mu_j \cdot e_i \otimes f_j.$$

The inner product in $V \otimes W$ is defined as follows

$$\langle e_i \otimes f_j, e_k \otimes f_l \rangle = \langle e_i, e_k \rangle \langle f_j, f_l \rangle.$$

The tensor product of two vectors $v = (v_1, \ldots, v_n) \in \mathbb{C}^n$ and $w = (w_1, \ldots, w_m) \in \mathbb{C}^m$ is defined as the vector $v \otimes w \in \mathbb{C}^{nm}$ such that

$$v \otimes w = \begin{pmatrix} v_1 w_1 \\ \vdots \\ v_1 w_m \\ \vdots \\ v_n w_1 \\ \vdots \\ v_n w_m \end{pmatrix}.$$

The tensor product of two operators $A \in \mathbb{C}^{n \times m}$ and $B \in \mathbb{C}^{p \times q}$ is defined as the operator $A \otimes B \in \mathbb{C}^{np \times mq}$ such that

$$(A \otimes B)(v \otimes w) = Av \otimes Bw.$$

The tensor product of two operators $P : \mathbb{C}^{n\times n} \to \mathbb{C}^{m\times m}$ and $Q \in \mathbb{C}^{o\times o} \to \mathbb{C}^{p\times p}$ is defined as the operator $P \otimes Q \in \mathbb{C}^{n\times n} \otimes \mathbb{C}^{o\times o} \to \mathbb{C}^{m\times m} \otimes \mathbb{C}^{p\times p}$ such that

$$(P \otimes Q)(A \otimes B) = P(A) \otimes Q(B).$$

The tensor product of two vector spaces corresponding to the direct sums of other vector spaces $V = V_1 \oplus \ldots \oplus V_n$ and $W = W_1 \oplus \ldots \oplus W_n$ is defined as

$$V \otimes W = V_1 \otimes W_1 \oplus \ldots \oplus V_1 \otimes W_n \oplus \ldots \oplus V_n \otimes W_1 \oplus \ldots \oplus \ldots \oplus W_n \otimes V_n.$$

The notation $(-)^{\otimes n}$ will be used to denote the tensor product of a vector space, vector, or operator with itself n times.
Considering vector spaces V, W, R, the following equalities hold:

$$v \otimes (w + r) = v \otimes w + v \otimes r, \text{ for all } v, w, r \in V, W, R;$$

$$(v + w) \otimes r = v \otimes r + w \otimes u, \text{ for all } v, w, r \in V, W, R;$$

$$(av) \otimes (bw) = ab(v \otimes w), \text{ for all } v, w \in V, W \text{ and scalars } a, b.$$

2.2 Quantum Computing Preliminaries

The basic unit of information in quantum computation is a quantum bit or qubit [30]. Just as classical bit, which can be in one of two states, 0 or 1, a qubit also has a state. Qubits are represented using *Dirac notation*, where the ket symbol $|\psi\rangle$ is used to denote a quantum state ψ. The corresponding bra symbol $\langle\psi|$ is used to denote the conjugate transpose of the state ψ. In this setting, the inner product of two states $|\psi\rangle$ and $|\phi\rangle$ is denoted $\langle\psi|\phi\rangle$ and is the same as $\langle\psi|\,|\phi\rangle$.

Definition 2.2.1. [27] Each isolated physical system is associated with a Hilbert space, known as the system's *state space*. The system's state is fully characterized by a *state vector*, which is a unit vector within this state space.

2.2.1 The 2-Dimensional Hilbert Space

The state of a single qubit is described by a normalized vector of the 2-dimensional Hilbert space $\mathbb{C}^2$. States $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ are equivalent to the classical states 0 and 1, respectively. These states, known as the *computational basis* states, form an orthonormal basis for this vector space. Unlike classical bits, a qubit is not restricted to the states $|0\rangle$ and $|1\rangle$; it can exist in a linear combination of these states, commonly referred to as a *superposition*. Consequently, a general qubit state can be written as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.5}$$

α and β are known as *amplitudes*. $|\alpha|^2$ and $|\beta|^2$ can be seen as the probabilities of measuring each state. Because $|\alpha|^2 + |\beta|^2 = 1$, Equation 2.5 and be rewritten as

$$|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right), \tag{2.6}$$

where θ, ϕ and γ are real numbers. $e^{i\gamma}$ is known as a *global phase* and is often ignored because it has no observable effects, *i.e.*, it does not affect the probabilities of measurement outcomes. When disregarding the global phase, the quantum state can $|\psi\rangle$ be represented as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.7}$$

which corresponds to a point in the unit sphere where θ marks the latitude (*i.e.* the polar angle) and ϕ marks the longitude (*i.e.* the azimuthal angle). This representation is traditionally called the Bloch sphere representation. A point in the latter representation corresponds to the vector in $\mathbb{R}^3$ defined by $(\cos\phi\sin\theta, \sin\phi\sin\theta, \cos\theta)$ and often called Bloch vector.

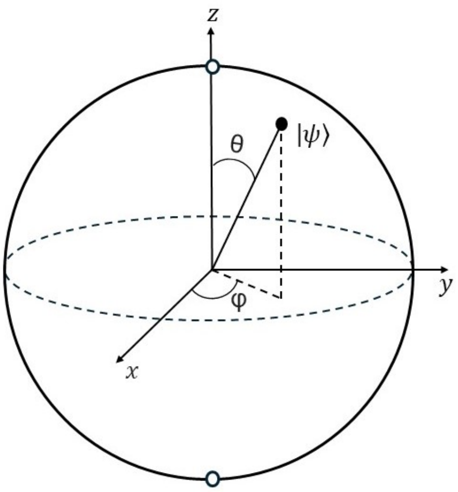


Figure 1: Bloch sphere representation of a qubit

The distance between two quantum states $|\psi\rangle$ and $|\psi'\rangle$ is their Euclidean distance in the Bloch sphere [8, 27].

There are infinite points in the Bloch sphere, which might suggest the possibility of encoding an infinite amount of information in the infinite binary expansion of the angle θ. However, when a qubit is measured, it collapses to one of the basis states, so only one bit of information can be extracted from a qubit. To accurately determine the amplitudes α and β, an infinite number of identical qubit copies would need to be measured. Nevertheless, it is still conceptually valid to think of these amplitudes as “hidden information”. One could say that quantum computation is the art of manipulating this hidden information using phenomena such as interference and superposition to perform tasks that would be impossible or inefficient with classical computers.

2.2.2 Multi-qubit States

The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. As a result, an n-qubit state can be represented by a unit vector in 2^n-dimensional Hilbert space, $\mathbb{C}^{2^n}$. The notations $|\psi\rangle \otimes |\phi\rangle$, $|\psi\rangle\,|\phi\rangle$, and $|\psi\phi\rangle$ are used to denote the tensor product of two states $|\psi\rangle$ and $|\phi\rangle$. As for any complex vector, $|\psi\rangle^{\otimes n}$ denotes the n-fold tensor product of state $|\psi\rangle$ with itself. The computational basis states of an n-qubit system are of the form $|x_1 \ldots x_n\rangle$ and so a quantum state of such a system is specified by 2^n amplitudes. For instance, a two-qubit state can be written as

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle\,.$$

It should be noted that unfortunately, no simple generalization of the Bloch sphere known for multiple qubits.

Entanglement

An interesting aspect of multi-qubit states is the phenomenon of *entanglement*. This term means strong intrinsic correlations between two (or more) particles when the quantum state of each of them cannot be described independently of the state of the other, i.e. cannot be written as a product of states of the individual qubits. Measuring one qubit of the entangled pair affects the state of the other qubit. This must happen even if the particles are far apart. In order to better understand this concept, consider the follow *Bell state* or *EPR pair*:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Upon measuring the first qubit, there are two possible outcomes: 0 with probability $1/2$ and 1 with probability $1/2$. If the first qubit is measured to be 0, the second qubit will also be 0 with probability 1. If the first qubit is measured to be 1, the second qubit will also be 1 with probability 1. Therefore, the measurement outcomes are correlated.

These correlations prompted Einstein, Podolsky, and Rosen to publish a paper [31] questioning the completeness of quantum mechanics in 1935. The EPR paradox presented a dilemma: the existence of entanglement (i.e., correlations that persist regardless of distance) versus local realism and hidden variables. Einstein argued that if two objects, which have interacted in the past but are now separated, exhibit perfect correlation, they must possess a set of properties determined before their separation. These properties would persist in each object, dictating the outcomes of measurements on both sides. Einstein believed that the strong correlations predicted by quantum mechanics necessitate the existence of additional properties not accounted for by the quantum formalism that determine the measurement results. Therefore, he argued that quantum mechanics might require supplementation, as it may not represent a complete or ultimate description of reality.

In 1964, John Bell made a remarkable discovery: the measurement correlations in the Bell state are stronger than those that could ever occur between classical systems [32]. He explored the idea that each entangled particle might possess hidden properties — unaccounted for by quantum mechanics—that determine the measurement outcomes. Then, through mathematical reasoning, Bell demonstrated that the correlations predicted by any local hidden variable theory cannot exceed a specific level. There is an upper limit of correlations fixed by what today is called the “Bell inequalities”. He found that quantum theory sometimes predicts correlations that exceed this limit. Consequently, an experiment could settle the debate by testing whether or not correlations surpass the bounds he had found following Einstein’s position.

In 1982, Alain Aspect conducted an experiment that confirmed the violation of the Bell inequalities [33]. In this experiment, polarizers were placed more than twelve meters apart. This meant that the correlation obtained could not be explained by the fact that the particles carry within them unmeasured properties. Moreover, it proved that the outcome of the measurement is not determined until the moment of measurement. There seemed to be an instantaneous exchange between two particles at the time of measurement when they were twelve meters apart.

Sixteen years later, Nicolas Gisin [34] and Anton Zeilinger [35] conducted similar experiments, demonstrating that entanglement persists over distances of several kilometers. More recently, [36] extended these tests using entangled photon pairs sent from a satellite to verify Bell's inequalities over a distance of one thousand kilometers, further confirming that, regardless of the distance, entangled particles behave as an indivisible, inseparable whole. The connection between them is so profound that it appears to challenge the principles of relativity. This phenomenon is known as *quantum nonlocality*.

2.2.3 Unitary operators and Measurements

Pauli Matrices

The Pauli matrices are a set of three 2×2 hermitian matrices that are defined as follows:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The eigenvectors and eigenvalues of the Pauli matrices are as follows:

$$\sigma_x \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \sigma_y \begin{pmatrix} 1 \\ i \end{pmatrix} = \begin{pmatrix} 1 \\ i \end{pmatrix}, \qquad \sigma_z \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$\sigma_x \begin{pmatrix} 1 \\ -1 \end{pmatrix} = -\begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \sigma_y \begin{pmatrix} 1 \\ -i \end{pmatrix} = -\begin{pmatrix} 1 \\ -i \end{pmatrix}, \quad \sigma_z \begin{pmatrix} 0 \\ 1 \end{pmatrix} = -\begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The normalized eigenvectors of σ_x are $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, and normalized eigenvectors of σ_y are $|+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$ and $|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle)$. The eigenvectors of σ_z are $|0\rangle$ and $|1\rangle$. These eigenvectors correspond to the $\hat{x}, \hat{y}$ and $\hat{z}$ axes of the Bloch sphere in Figure 1, respectively.

When matrices σ_x, σ_y or σ_z are applied to a state on the Bloch sphere, they rotate the state by π radians around the $\hat{x}, \hat{y}$ or $\hat{z}$ axis, respectively. For example, the action of σ_x on the state $|0\rangle$ is to rotate it to $|1\rangle$, and vice versa. Note that for the eigenstates of these matrices with eigenvalue -1, this still applies if considering a global phase of $-1 = e^{i\pi}$, given that two quantum states $|\psi\rangle$ and $e^{i\phi}\,|\psi\rangle$ are indistinguishable by any quantum measurement.

Matrices σ_x and σ_z will also be referred to as X and Z, respectively.

Unitary operators

Closed systems, i.e., systems that do not interact with other systems evolve according to unitary operators. In quantum computation, these unitary operators are also known as *gates*. For a state $|\psi\rangle$, a unitary operator U describes an evolution from $|\psi\rangle$ to $U|\psi\rangle$.

Pauli matrices are examples of unitary operators. The X and Z gates are often referred to as the *not* and *phase flip* gates, respectively. Other important unitary operators include *Hadamard gate*, denoted H, which maps $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$, and the *phase-shift gate*, denoted P, which leaves $|0\rangle$ unaltered applies a phase shift of θ to the state $|1\rangle$:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}.$$

When the Pauli matrices are exponentiated, they result in three valuable classes of unitary matrices, corresponding to the rotation operators around the $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes, which are defined as follows:

$$R_x(\theta) = e^{-i\theta\sigma_x/2} = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right)\sigma_x = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix},$$

$$R_y(\theta) = e^{-i\theta\sigma_y/2} = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right)\sigma_y = \begin{pmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix},$$

$$R_z(\theta) = e^{-i\theta\sigma_z/2} = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right)\sigma_z = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}.$$

Theorem 2.2.2. *[27] Suppose U is a unitary operation on a single qubit. Then there exist real numbers α, β, γ and δ such that*

$$U = e^{i\alpha}R_z(\beta)R_y(\gamma)R_z(\delta).$$

There are also multi-qubit gates, such as the *controlled-not* gate, denoted ***CNOT***, which leaves the states $|00\rangle$ and $|01\rangle$ unchanged, and maps $|10\rangle$ and $|11\rangle$ to each other:

$$\mathit{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

In this case, as the state of the first qubit determines if the X gate is applied to the second qubit, the first qubit is called the *control qubit* and the second qubit the *target qubit*.
There is an "extension" of the controlled-not gate, the controlled-U gate, where U is a unitary gate acting on a single qubit. This gate applies the gate U to the target qubit if the control qubit is in state $|1\rangle$ and does nothing otherwise. It is defined as:

$$\begin{aligned}\mathit{CU}(|0\rangle \otimes |\psi\rangle) &= |0\rangle \otimes |\psi\rangle \\ \mathit{CU}(|1\rangle \otimes |\psi\rangle) &= |1\rangle \otimes U|\psi\rangle .\end{aligned}$$

It should be noted that no completely closed systems exist in the universe. Nevertheless, for many systems, the approximation of a closed system is valid.

Measurements

There are times when it necessary to observe the system to extract information. This interection leaves the system no longer closed and, consequently, the evolution of the system is no longer unitary.
The act of measuring a qubit is represented by a set of operators called *measurement operators*, denoted $\{M_m\}$. These operators act on the state space of the system being measured. The index m refers possible measurement outcomes. These measurement operators must satisfy the completeness equation $\sum_m M_m^\dagger M_m = I$, which ensures that the probabilities of all possible outcomes sum to 1. If a measurement M_m is performed on a state $|\psi\rangle$ the outcome m is observed with probability $p_m = \langle\psi| M_m^\dagger M_m |\psi\rangle$ for each m. Moreover, after a measurement yielding outcome m, the state collapses to $\frac{M_m|\psi\rangle}{\sqrt{p_m}}$.
In the case of the computational basis, the measurement operators are the projectors onto the basis states $|0\rangle$ and $|1\rangle$ denoted by $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$, respectively. Considering an arbitrary state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the probabilities of measuring 0 and 1 are $p_0 = \langle\psi| M_0 M_0^\dagger |\psi\rangle = \langle\psi| M_0 |\psi\rangle = |\alpha|^2$, and $p_1 = \langle\psi| M_1 M_1^\dagger |\psi\rangle = \langle\psi| M_1 |\psi\rangle = |\beta|^2$. Consequently the states after measurement are $\frac{M_0|\psi\rangle}{|\alpha|} = \frac{\alpha}{|\alpha|}|0\rangle = |0\rangle$ (with $p = p_0$) and $\frac{M_1|\psi\rangle}{|\beta|} = \frac{\beta}{|\beta|}|1\rangle = |1\rangle$ (with $p = p_1$).
From now on, unless stated otherwise, any reference to measurement should be understood as pertaining to the computational basis.
As previously mentioned, any states $|\psi\rangle$ and $e^{i\phi}|\psi\rangle$ are indistinguishable by any quantum measurement. Consider a measurement operator M_m, the probabilities of obtaining out-

come m are $\langle\psi| M_m^\dagger M_m |\psi\rangle$ and $\langle\psi| e^{-i\theta} M_m^\dagger M_m e^{i\theta} |\psi\rangle = \langle\psi| M_m^\dagger M_m |\psi\rangle$. For this reason, it is said that these states are equal from an observational point of view.

2.2.4 Density operators

Until now the state vector formalism was used. However there is an alternative formulation using density operators. The density operator is often known as the *density matrix*, the two terms will be used interchangeably.

A quantum state $|\psi\rangle$ is said to be a *pure state* if it is completely known, i.e. if it can be written as a ket. In this case, the state can be written in the density operator formalism as $\rho = |\psi\rangle\langle\psi|$. On the other hand, a state that is a probabilistic mixture of pure states is designated a *mixed state*. A mixed state $\sum_i \alpha_i |\psi_i\rangle$ can be represented by a density operator $\rho = \sum_i |\alpha_i|^2 |\psi_i\rangle\langle\psi_i|$. Note that $|\alpha_i|^2$ is the probability of the system being in state $|\psi_i\rangle$.

With respect to density operators, when applying a unitary operator U to a state ρ, the resulting state is $U\rho U^\dagger$. Regarding measurements, given a collection of measurement operators, $\{M_m\}$, the probability of obtaining outcome m is $p(m) = \text{Tr}(M_m \rho M_m^\dagger)$, and after a measurement yielding outcome m, the state collapes to $\frac{M_m^\dagger \rho M_m}{\text{Tr}(M_m \rho M_m^\dagger)}$.

In subsection 2.2.1 it was shown how to determine the cartesian coordenates of a pure state in the Bloch Sphere from the state vector. For an arbitrary 2×2 density matrix, the following holds

$$\rho = \frac{1}{2}(I + r_x\sigma_x + r_y\sigma_y + r_z\sigma_z), \tag{2.8}$$

where $\vec{r} = (r_x, r_y, r_z)$ is a real three-dimensional vector such that $\|\vec{r}\|_2 \leq 1$. This vector is known as the Bloch vector for the state ρ. Since ρ is Hermitian. r_x, r_y and r_z are always real. The inverse map of Equation 2.8 is

$$r_\mu = \text{Tr}(\rho\sigma_\mu) \tag{2.9}$$

Note that given that the trace is linear and matrix multiplication distributes over matrix addition, the cartesian coordinates of an operator consisting of the sum or subtraction of density operators can also be determined by Equation 2.9.

Reduced density operator

Density operators are particularly well-suited for describing individual subsystems of a composite quantum system. This type of description is provided by the *reduced density operator*

Given physical systems A and B whose composite system is given by the density operator ρ_{AB}, the reduced density operator for subsystem A is

$$\rho_A = \text{Tr}_B(\rho_{AB}),$$

where Tr_B is the partial trace over the Hilbert space of subsystem B, defined as

$$\begin{aligned}
\text{Tr}_B(|a_1\rangle \langle a_2| \otimes |b_1\rangle \langle b_2|) &= |a_1\rangle \langle a_2| \, \text{Tr}(|b_1\rangle \langle b_2|) \\
&= |a_1\rangle \langle a_2| \sum_{\mu} \langle \mu| \, |b_1\rangle \langle b_2| \, |\mu\rangle \\
&= |a_1\rangle \langle a_2| \sum_{\mu} \langle \mu|b_1\rangle \langle b_2|\mu\rangle \\
&= |a_1\rangle \langle a_2| \sum_{\mu} \langle b_2|\mu\rangle \langle \mu|b_1\rangle \\
&= |a_1\rangle \langle a_2| \langle b_2|b_1\rangle
\end{aligned}$$

where $|a_1\rangle$ and $|a_2\rangle$ are any two vectors in the state space of A, $|b_1\rangle$ and $|b_2\rangle$ are any two vectors in the state space of B, and $\{|\mu\rangle\}$, span the state space of B. Therefore, by linearly, the partial trace of $\rho_{AB} = \sum_{ijkl} p_{ijkl} |a_i\rangle \langle a_j| \otimes |b_k\rangle \langle b_l|$ is

$$\begin{aligned}
\text{Tr}_B(\rho_{AB}) &= \sum_{ijkl} p_{ijkl} \text{Tr}_B \left(|a_i\rangle \langle a_j| \otimes |b_k\rangle \langle b_l|\right) \\
&= \sum_{ijkl} p_{ijkl} |a_i\rangle \langle a_j| \sum_{\mu} \langle b_l|\mu\rangle \langle \mu|b_k\rangle = \sum_{ijkl} p_{ijkl} |a_i\rangle \langle a_j| \langle b_l|b_k\rangle
\end{aligned}$$

To demonstrate that this operator is a density operator, it must be shown that it possesses unit trace and is positive semidefinite. Given that the trace of ρ_{AB} corresponds to the sum of the diagonal elements of the density matrix, $\sum_i ikp_{iikk}$, and that that is also the case for the reduced density operator ρ_A, the sum of the diagonal elements is preserved by the partial trace. Thus, ρ_a has trace equal to 1. Moreover, considering that

$$\langle |\psi\rangle , \rho_{AB} |\psi\rangle\rangle = \langle \psi| \rho_{AB} |\psi\rangle = \langle \psi| \sum_{ijkl} p_{ijkl} |a_i\rangle \langle a_j| \otimes |b_k\rangle \langle b_l| \, |\psi\rangle$$

$$\begin{aligned}
&= \sum_{ijkl} p_{ijkl} \langle \psi| \, |a_i\rangle \langle a_j| \otimes |b_k\rangle \langle b_l| \, |\psi\rangle \\
&= \sum_{ijkl} p_{ijkl} \sum_{m} (\alpha_m^* \langle a_m| \otimes \langle b_m|) \, |a_i\rangle \langle a_j| \otimes |b_k\rangle \langle b_l| \sum_{m} (\alpha_m |a_m\rangle \otimes |b_m\rangle) \\
&= \sum_{ijklm} p_{ijkl} |\alpha_m|^2 \langle a_m|a_i\rangle \langle a_j|a_m\rangle \langle b_m|b_k\rangle \langle b_l|b_m\rangle
\end{aligned}$$

$$=\sum_{ijkl} p_{ijkl} \langle a_j|a_i\rangle \langle b_l|b_k\rangle$$

and that,

$$\langle |\psi\rangle, \rho_A |\psi\rangle\rangle = \langle\psi| \rho_A |\psi\rangle = \langle\psi| \sum_{ijkl} p_{ijkl} |a_i\rangle \langle a_j| \langle b_l|b_k\rangle |\psi\rangle$$

$$=\sum_{ijkl} p_{ijkl} \langle b_l|b_k\rangle \langle\psi| |a_i\rangle \langle a_j| |\psi\rangle =$$

$$=\sum_{ijklm} p_{ijkl} \langle b_l|b_k\rangle \sum_m (\alpha_m^* \langle a_m|) |a_i\rangle \langle a_j| \sum_m (\alpha_m |a_m\rangle)$$

$$=\sum_{ijklm} p_{ijkl} \langle b_l|b_k\rangle |\alpha_m|^2 \langle a_m|a_i\rangle \langle a_j|a_m\rangle = \sum_{ijkl} p_{ijkl} \langle b_l|b_k\rangle \langle a_j|a_i\rangle$$

$$=\sum_{ijkl} p_{ijkl} \langle a_j|a_i\rangle \langle b_l|b_k\rangle$$

where, $\{|a_m\rangle\}$ span the space of A and $\{|b_m\rangle\}$ span the space of B, it is possible to conclude that $\langle |\psi\rangle, \rho_{AB} |\psi\rangle\rangle = \langle |\psi\rangle, \rho_A |\psi\rangle\rangle$. Thus, since $\langle |\psi\rangle, \rho_{AB} |\psi\rangle\rangle \geq 0$, the same applies to $\langle |\psi\rangle, \rho_A |\psi\rangle\rangle$. Therefore, ρ_A is a positive semidefinite, and, consequently, a density operator. In certain situations it is more advantageous to consider the reduced density operator for subsystem A defined as:

$$\rho_A = \text{Tr}_B(\rho_{AB}) = \sum_\mu \langle\mu| \rho_{AB} |\mu\rangle,$$

where $\{|\mu\rangle\}$, span the state space of B and act only in the state space of B.
Similarly, the reduced density operator for subsystem B is $\rho_B = \text{Tr}_A(\rho_{AB})$.

2.2.5 Quantum Channels

Thus far, only two types of quantum operations have been discussed: unitary operators, which describe the evolution of a closed quantum system, and measurements, which describe the act of observing a quantum system. Now, a new type of quantum operation that accounts for the more realistic notion of interaction between a quantum system and an environment will be introduced. Nonetheless, it is necessary to first introduce a few key concepts.

Definition 2.2.3. A *super-operator* Q is a linear map between the space of operators on a Hilbert space.

Definition 2.2.4. A super-operator Q is called *positive* if it sends positive matrices to positive matrices, *i.e.* $A \geq 0 \Rightarrow QA \geq 0$.

Definition 2.2.5. A super-operator Q is said to be *completely positive* if for all k,

$$Q \otimes I_{\mathbb{C}^{k\times k}} : \mathbb{C}^{n\times n} \otimes \mathbb{C}^{k\times k} \to \mathbb{C}^{m\times m} \otimes \mathbb{C}^{k\times k}$$

is positive.

Definition 2.2.6. A super-operator Q is called *trace-preserving* if Tr $(QA) = \text{Tr}(A)$.

Since density matrices are positive, any physically allowed transformation must be represented by a positive operator. Moreover, this is not sufficient on its own: since one can always extend the space $\mathbb{C}^{n\times n}$ to $\mathbb{C}^{n\times n} \otimes \mathbb{C}^{k\times k}$ by adjoining a new quantum system, any physically allowed transformation must be completely positive. Finally, since the trace of a density matrix is always 1, any physically allowed transformation must be trace-preserving. A **Completely Positive Trace-Preserving** (**CPTP**) operator is traditionally called a *quantum channel*.

Kraus operator sum representation

Assume that there is a quantum system S of interest which is a subsystem of a larger system which also includes an environment E. These systems have a joint unitary evolution described by a unitary operator U acting on the composite system, $U = \rho_{SE} \mapsto U\rho_{SE}U^\dagger$. Given that density matrices are positve operators, by (Definition 2.1.21), the density operator of the environment ρ_E initially can be written as

$$\rho_E = \sum_i p_i \left|i\right\rangle \left\langle i\right|$$

where $|i\rangle$ form an orthonormal basis for the state space of E and p_i are positive.
The state of the subsystem S after the unitary evolution corresponds to the partial trace of the joint state over the environment,

$$\begin{aligned}\rho_S =& \text{Tr}_E(U\rho_{SE}U^\dagger)\\ =& \sum_\mu \left\langle \mu\right| U\rho_{SE}U^\dagger \left|\mu\right\rangle\end{aligned}$$

where $\{|\mu\rangle\}$ span the state space of E.

Considering that initially both systems are completely decoupled, the initial state of the system can be written as $\rho_{SE} = \rho_S \otimes \rho_E$. Thus,

$$\begin{aligned}\rho_S &= \sum_\mu \langle\mu| U\rho_S \otimes \sum_i p_i |i\rangle \langle i| U^\dagger |\mu\rangle \\ &= \sum_{\mu i} \sqrt{p_i} \langle\mu| U |i\rangle \rho_S \sqrt{p_i} \langle i| U^\dagger |\mu\rangle \\ &= \sum_{\mu i} \mathsf{K}_{\mu i}\rho_S \mathsf{K}_{\mu i}^\dagger\end{aligned}$$

where the set of operators $\{\mathsf{K}_{\mu i}\}$ is designated *Kraus operators* and $\mathsf{K}_{\mu i} = \sqrt{p_i} \langle\mu| U |i\rangle$. Note that $\{|\mu\rangle\}$ and $\{|i\rangle\}$, act only in the state space of E. The equation $\rho_S = \sum_{\mu i} \mathsf{K}_{\mu i}\rho_S \mathsf{K}_{\mu i}^\dagger$ is called an ***Operator Sum Representation*** (***OSR***).

OSR can be thought of as a quantum channel that maps ρ_S to $\sum_{\mu i} \mathsf{K}_{\mu i}\rho_S \mathsf{K}_{\mu i}^\dagger$, given this map is **CPTP** [37].

Non-selective measurements

In the previously presented formalism to represent all the possible outcomes of a measurement, described by a set of operators $\{M_m\}$, on a state ρ, it would be necessary to write that state ρ collapse to state $\rho_m = \frac{M_m^\dagger \rho M_m}{\mathrm{Tr}(M_m \rho M_m^\dagger)}$ with probability $p(m) = \mathrm{Tr}(M_m \rho M_m^\dagger)$, for each possible outcome m. However, this is not appropriate for calculations. Consequently, to better represent the effect of a measurement on a quantum system *non-selective measurements* are used. In this case, the possible outcomes are not explicitly stated, and the state after the measurement is as follows:

$$\rho = \sum_m p_m \rho_m = \sum_m M_m \rho M_m^\dagger$$

This last equality corresponds to an Kraus operator sum representation, where the set of Kraus operators is $\{M_m\}$.

Norms on quantum channels

Definition 2.2.7. The norm of a super-operator $Q : \mathbb{C}^{n\times n} \to \mathbb{C}^{m\times m}$ is defined:

$$\|Q\|_1 = \max\{\|QA\|_1 \mid \|A\|_1 = 1\}, \tag{2.10}$$

where $A \in \mathbb{C}^{n\times n}$

Unfortunately, this norm is not stable under tensoring , given that the inequation $\|Q \otimes I_{\mathbb{C}^{n\times n}}\|_1 \leq \|Q\|_1$ does not hold [12]. As a result, the diamond norm, which is based on the trace norm, is used instead in the context of quantum channels.

Definition 2.2.8. Given a super-operator $Q : \mathbb{C}^{n\times n} \to \mathbb{C}^{m\times m}$, the diamond norm, $\| \cdot \|_\diamond$, is defined as:

$$\|Q\|_\diamond = \|Q \otimes I_{\mathbb{C}^{n\times n}}\|_1 \tag{2.11}$$

For this norm it holds that for all super-operators $Q : \mathbb{C}^{n\times n} \to \mathbb{C}^{m\times m}$ and $S : \mathbb{C}^{m\times m} \to \mathbb{C}^{o\times o}$, if Q is a quantum channel then $\|SQ\|_\diamond \leq \|S\|_\diamond$ [12, Proposition 3.44 and Proposition 3.48], and if S is a quantum channel, then $\|SQ\|_\diamond \leq \|Q\|_\diamond$. This is a desireble property, as is guarantees that quantum operations do not increase the distance between states, and as a consequence, composition of programs is valid.

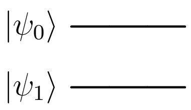
Consider an operator r : $(\mathbb{C}^n \to \mathbb{C}^m) \to (\mathbb{C}^{n\times n} \to \mathbb{C}^{m\times m})$ that sends an operator T to the mapping $A \mapsto TAT^\dagger$. The exact calculation of distances induced by $\|\|_\diamond$ tends to be quite complicated, but a useful property for calculating the distance between quantum channels in the image of r is provided in [12]. Consider two operators $T, S : n \to m$. There exists a unit vector $v \in \mathbb{C}^n$ such that,

$$\|r(T)(vv^\dagger) - r(S)(vv^\dagger)\|_1 = \|r(T) - r(S)\|_\diamond \tag{2.12}$$

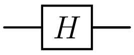
2.2.6 Quantum circuits

As quantum computation remains in its early stages of development, programming is primarily based on the use of *quantum circuits*. A quantum circuit consists of wires and *quantum gates*, which serve to transmit and manipulate quantum information. Each wire corresponds to a qubit, while the gates represent operations that can be applied to these qubits.

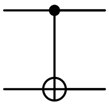
In this subsection the notation for the quantum gates used in this work will be introduced. Wires in parallel represent the tensor product of the respective qubits. For instance, $\psi_0 \otimes \psi_1$ corresponds to



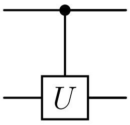
The single bit gates presented in Section 2.2.3 are represented as a box with the symbom of the gate inside. For example, the Hadamard gate is represented as



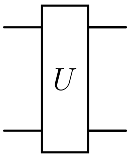
The controlled-not gate, which is a two-qubit gate, is represented as



Similarly, the controlled-U gate, where U is an unitary single-qubit gate, is represented as

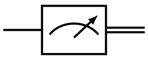


An arbitrary unitary operator acting on n qubits is represented as a box acting on n wires. For instance, the operator U acting on two qubits is represented as



CPTP maps are depicted as boxes containing the corresponding map symbols. The relevant **CPTP** operators will be introduced in **??**.

The measurement operation is representes by a "meter" symbol. Given that output of a measurement is a classical bit, the wire representing the output of a measurement is a classical wire, representes by a double line.



2.2.7 No-cloning theorem

The no-cloning theorem states that it is impossible to duplicate an unknown quantum bit [38]. In this subsection, an elementary proof of this theorem will be presented.

Suppose that there exists a unitary operator U that recieves a qubit $|\psi\rangle$ and some standard pure state $|s\rangle$ as input and oututs the state $|\psi\rangle \otimes |\psi\rangle$. The action of U can be written as

$$U\left(|\psi\rangle \otimes |s\rangle\right) = |\psi\rangle \otimes |\psi\rangle$$

Consider the aplication of U to two pure states $|\psi_0\rangle$ and $|\psi_1\rangle$,

$$U\left(|\psi_0\rangle \otimes |s\rangle\right) = |\psi_0\rangle \otimes |\psi_0\rangle$$
$$U\left(|\psi_1\rangle \otimes |s\rangle\right) = |\psi_1\rangle \otimes |\psi_1\rangle .$$

Given that unitary operators preserve inner products, the following equality should hold:

$$\langle\psi_0|\psi_1\rangle = \left(\langle\psi_0|\psi_1\rangle\right)^2$$

This equation is only satisfied if $\langle\psi_0|\psi_1\rangle = 0$ or $\langle\psi_0|\psi_1\rangle = 1$. The first case implies that $|\psi_0\rangle$ and $|\psi_1\rangle$ are orthogonal, and the second case implies that they are in the same state. Therefore, it is only possible to clone orthogonal states. These are the states perfectly distinguishable by measurement and thus are equivalent to copying classical information. For instance, it is impossible to the clone qubits $\psi_0 = |0\rangle$ and $\psi_1 = |-\rangle$, since they are not orthogonal.

It should be noted that this principle is upheld by the type system outlined in Figure 2, which does not allow the repeated use of a variable (seen as a quantum resource).

2.3 Functional Analysis

Definition 2.3.1. (*Cauchy sequence*) Suppose d is a metric on a set X. A sequence $\{x_n\} \subset X$ is called a *Cauchy sequence* if, for every $\varepsilon > 0$, there exists an integer $N \in \mathbb{N}$ such that $d(x_m, x_n) < \varepsilon$ for all $m, n > N$. The metric d is said to be *complete* if every Cauchy sequence in X converges to a point in X

Definition 2.3.2. A *Banach space* is a normed vector space that is complete with respect to the metric induced by its norm. In other words, every Cauchy sequence in the space must converge to a limit within the space.

Espaços de Banach, duais, bounded operators, norm topology, weak topology ?

Se calhar vou abolir isto e meter as coias relacionadas com pp em pp e as com as op algebras nas op algebras

2.4 Probabilistic Programming/Measure theory

This section introduces the fundamentals of measure theory, a key component in defining the semantics of probabilistic programs, drawing inspiration primarily from [39] and [40]. For a more in-depth exploration of lambda calculus theory, see reference [41].

In this work, we only consider finite measures; hence, the term ”measure” implicitly refers to a finite measure unless stated otherwise.

2.4.1 Probabilistic Programming and Measure Theory

Probabilistic programs offer a structured approach to drawing statistical conclusions from uncertain data or real-world observations. They generalize probabilistic graphical models beyond the capabilities of Bayesian networks and are expected to have broader applications in machine intelligence. These programs are are employed across various domains. They control autonomous systems, verify security protocols, and implement randomized algorithms for solving computationally intractable problems. As a result, they are becoming increasingly central to AI development. At their core, they aim to democratize probabilistic modeling by providing programmers with expressive, high-level abstractions for machine learning and statistical reasoning [42].

Probabilistic computation involves running programs incorporating randomness, leading to output behaviors characterized by probability distributions rather than deterministic outcomes. To effectively understand and analyze these programs, it is essential to have a solid foundation in reasoning about probability distributions. That is where measure theory comes into play.

2.4.2 What is measure theory?

Throughout history, mathematicians sought to extend the ideas of length, area, and volume. The most effective way to generalize these concepts is through the idea of a measure. Abstractly, a measure is a function defined on sets with additive properties mirroring length, area, and volume.

We begin with a simple example inspired by [40] to develop an intuition for the concepts of measure and measure space. Imagine an open field S covered in snow after a storm. Suppose we wish to measure the amount of snow accumulated in as many field regions as possible. Assume we have accurate tools for measuring snow over standard geometric shapes like triangles, rectangles, and circles. We can approximate irregularly shaped regions using combinations of these standard shapes and then apply a limiting process to assign a consistent measure to such regions. Let $\mathcal{B}$ denote the collection of subsets of S that are deemed *measurable*, let $\lambda(A)$ represent the amount of snow in each $A \in \mathcal{B}$, and let A^c denote the complement of a set A.

For this framework to make sense, it is reasonable to require that $\mathcal{B}$ and $\lambda(\cdot)$ satisfy the following properties:

Properties of B:

1. If $A \in B$, then the complement $A^c \in \mathcal{B}$. (*i.e.*, if we can measure the snow on a set A, and we know the total amount on S, then we can determine the snow on the remaining part A^c.)

2. If $A_1, A_2 \in \mathcal{B}$, then $A_1 \cup A_2 \in \mathcal{B}$. (*i.e.*, if we can measure the snow on two regions A_1 and A_2, we should also be able to measure it on their union.)

3. If $\{A_n\}_{n\geq 1} \subset \mathcal{B}$ is an increasing sequence, *i.e.*, $A_n \subset A_{n+1}$ for all n, then $\bigcup_{n=1}^{\infty} A_n \in \mathcal{B}$. (*i.e.*, if each set in a increasing sequence of regions is measurable, then their limit—the union—should also be measurable.)

4. The collection $\mathcal{B}$ contains a base class C of simple, well-behaved sets (e.g., triangles, rectangles, circles) for which measurement is initially defined.

Properties of $\lambda(\cdot)$:

1. $\lambda(A) \geq 0$ for all $A \in \mathcal{B}$ (*i.e.*, the amount of snow on any set must be nonnegative).

2. If $A_1, A_2 \in \mathcal{B}$ and $A_1 \cap A_2 = \emptyset$, then $\lambda(A_1 \cup A_2) = \lambda(A_1) + \lambda(A_2)$ (*i.e.*, The total amount of snow over two non-overlapping regions is just the sum of the snow in each region. This characteristic of λ is known as *finite additivity*.)

3. If $\{A_n\}_{n\geq 1} \subset \mathcal{B}$ is an increasing sequence, *i.e.*, $A_n \subset A_{n+1}$ then $\lambda(\lim_{n\to\infty} A_n) = \lambda\left(\bigcup_{n=1}^{\infty} A_n\right) = \lim_{n\to\infty} \lambda(A_n)$. (*i.e.*, if a set can be approximated by an increasing sequence of measurable sets $\{A_n\}_{n\geq 1}$, then $\lambda(A) = \lim_{n\to\infty} \lambda(A_n)$. This is known as *monotone continuity from below*.)

2.4.3 Measurable spaces and measures

Remarkably, these intuitive conditions give rise to a profoundly versatile and far-reaching theoretical framework. The requirements imposed on $\mathcal{B}$ and λ can equivalently be stated as follows:

Properties of $\mathcal{B}$:

1. $\emptyset \in \mathcal{B}$.
2. $A \in \mathcal{B} \implies A^c \in \mathcal{B}$.
3. $A_1, A_2, \dots \in \mathcal{B} \implies \bigcup_i A_i \in \mathcal{B}$ (this is known as *closure under countable unions*).

Properties of λ:

1. $\lambda(\cdot) \geq 0$ and $\lambda(\emptyset) = 0$.
2. If $\{A_n\}_{n\geq 1} \subset \mathcal{B}$ is a sequence of pairwise disjoint sets (i.e., $A_i \cap A_j = \emptyset$ for $i \neq j$), then $\lambda\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} \lambda(A_n)$ (this is known as *countable additivity*).

A collection $\mathcal{B}$ of subsets of S satisfying the above conditions for B is designated a *σ-algebra*. Similarly, a set function λ defined on a σ-algebra $\mathcal{B}$ that fulfills the above properties for λ qualifies as a *measure*.

Definition 2.4.1. (*σ-algebra*) A *σ-algebra* $\mathcal{B}$ on a set S is a collection of subsets of S that includes the empty set, is closed under complementation with respect to S, and is closed under countable unions.

Definition 2.4.2. (*Measurable space*) The pair $(S, \mathcal{B})$ where $\mathcal{B}$ is a σ-algebra constitutes a *measurable space*. The elements of $\mathcal{B}$ are called the *measurable sets* of the space.

Definition 2.4.3. (*Borel σ-algebra*) The Borel σ-algebra of $\mathbb{R}^n$ is the σ-algebra $\mathcal{B}(\mathbb{R}^n)$ generated by all open sets. The sets in $\mathcal{B}(\mathbb{R}^n)$ are called Borel sets.

Definition 2.4.4. (*Measure*) A *measure* on the measurable space $(S, \mathcal{B})$ is a function $\mu : \mathcal{B} \to [0, \infty]$ that is countably additive and satisfies $\mu(\emptyset) = 0$. A measure on $(S, \mathcal{B})$ is called a *probability measure* if it assigns total measure 1 to the entire space, that is, $\mu(S) = 1$.

In the context of probability theory, such measures are often referred to as *distributions*. In what follows, we will use the terms measure and distribution interchangeably.
One of the most important measures is the Lebesgue measure on the real line (*i.e.*, the length in $\mathbb{R}$), and its generalizations to $\mathbb{R}^n$. It is characterized as the unique measure on the Borel sets, whose value on every interval is its length. That is, for any interval $(a, b) \subset \mathbb{R}$, $\lambda((a, b)) = b - a$.

The Lebesgue measure is *translation-invariant*, meaning that shifting a set does not change its measure, and it is also *σ-finite*.

For any $s \in S$, the *Dirac measure* (also known as the *Dirac delta* or *point mass* at s) is the probability measure defined by

$$\delta_s(B) = \begin{cases} 1, & \text{if } s \in B, \\ 0, & \text{if } s \notin B. \end{cases}$$

A measure is called *discrete* if represented as a countable weighted sum of Dirac measures. In particular, a *convex combination* of Dirac measures yields a discrete probability measure. These are of the form $\sum_{s \in C} a_s \delta_s$, where $a_s \geq 0$ for all $s \in C$, and the weights satisfy $\sum_{s \in C} a_s = 1$.

On the other hand, a measure μ on a measurable space $(S, \mathcal{B})$ is called *continuous* if it assigns zero measure to all singleton sets, i.e., $\mu(\{s\}) = 0$ for every $s \in S$. An example of a continuous measure is the *Lebesgue measure* on $\mathbb{R}^n$ (for $n \in \mathbb{N}$).

Definition 2.4.5. (*Pushforward measure*) Let $(S, \mathcal{B}_S)$ and $(T, \mathcal{B}_T)$ be measurable spaces. Given a function $f : (S, \mathcal{B}_S) \to (T, \mathcal{B}_T)$ and a measure μ on $\mathcal{B}_S$, the *pushforward measure* $f_*(\mu)$ on $\mathcal{B}_T$ is defined by:

$$f_*(\mu)(B) = \mu(f^{-1}(B)), \quad B \in \mathcal{B}_T.$$

Definition 2.4.6. Let $(S, \mathcal{B}_S)$ and $(T, \mathcal{B}_T)$ be measurable spaces. A function $f : S \to T$ is said to be *measurable* if for every measurable subset $B \in \mathcal{B}_T$, the preimage $f^{-1}(B) \in \mathcal{B}_S$.

Definition 2.4.7. (*Lebesgue integral*) The *Lebesgue integral* generalizes the familiar Riemann integral. Consider a measurable space $(S, \mathcal{B})$ and a bounded measurable function $f \colon S \to \mathbb{R}_0^+$, with upper and lower bounds M and m, respectively. The *Lebesgue integral* of f with respect to a measure $\mu \colon \mathcal{B} \to \mathbb{R}_0^+$, denoted $\int f \, d\mu$, is defined as the limit of finite weighted sums of the form:

$$\sum_{i=0}^{n} f(s_i)\mu(B_i),$$

where $\{B_0, \ldots, B_n\}$ forms a measurable partition of S, and within each B_i, the variation of f does not exceed $(M - m)/n$. Here, $s_i \in B_i$ for each i, and the limit is taken over increasingly refined partitions.

In the case of a finite discrete space $n = \{1, 2, \ldots, n\}$, the Lebesgue integral simplifies to a weighted sum:

$$\int f\, d\mu = \sum_{i=1}^{n} f(i)\mu(\{i\}).$$

Given two measurable spaces $(S_1, \mathcal{B}_1)$ and $(S_2, \mathcal{B}_2)$, their product is the measurable space $(S_1 \times S_2, \mathcal{B}_1 \otimes \mathcal{B}_2)$, where $S_1 \times S_2$ is the cartesian product and $\mathcal{B}_1 \otimes \mathcal{B}_2$ is the σ-algebra generated by all measurable rectangles $B_1 \times B_2$ with $B_1 \in \mathcal{B}_1$ and $B_2 \in \mathcal{B}_2$:

$$\mathcal{B}_1 \otimes \mathcal{B}_2 \coloneqq \sigma\left(\{B_1 \times B_2 \mid B_1 \in \mathcal{B}_1,\ B_2 \in \mathcal{B}_2\}\right).$$

Measures on this product space are called *joint distributions*, and are uniquely determined by their values on measurable rectangles due to the inductive structure of the product σ-algebra. *Product measures* are a significant class of joint distributions and are defined from measures

Definition 2.4.8. (*product measures*) Let $(S_1, \mathcal{B}_1)$ and $(S_2, \mathcal{B}_2)$ be measurable spaces, and let μ_1 and μ_2 be measures on these spaces, respectively. The *product measure* $\mu_1 \otimes \mu_2$ is defined on measurable rectangles by

$$(\mu_1 \otimes \mu_2)(B_1 \times B_2) = \mu_1(B_1)\mu_2(B_2).$$

This definition extends uniquely to a joint distribution $\mu_1 \otimes \mu_2 : \mathcal{B}_1 \otimes \mathcal{B}_2 \to \mathbb{R}_0^+$, and reflects the notion of probabilistic independence: sampling from $\mu_1 \otimes \mu_2$ is equivalent to independently sampling from μ_1 and μ_2.

Definition 2.4.9. (*Markov Kernel*) Let $(S, \mathcal{B}_S)$ and $(T, \mathcal{B}_T)$ be measurable spaces. A *Markov kernel* is a mapping $P : S \times \mathcal{B}_T \to [0, 1]$ satisfying the following two properties:

1. For each fixed $s \in S$, the function $P(s, \cdot) : \mathcal{B}_T \to [0, 1]$ is a probability measure on $(T, \mathcal{B}_T)$.

2. For each fixed $A \in \mathcal{B}_T$, the function $P(\cdot, A) : S \to [0, 1]$ is a measurable function on $(S, \mathcal{B}_S)$.

By a slight abuse of terminology, we will also use the term *Markov kernel* to refer to a mapping $P' : S \to (\mathcal{B}_T \to [0, 1])\, P'(s) = \mu$, when $P : S \times \mathcal{B}_T \to [0, 1], , P(s, A) = \mu(A)$ is a Markov kernel in the definition above.

By a slight abuse of terminology, we will also refer to a mapping $P' : S \to (\mathcal{B}_T \to [0, 1])\, P'(s) = \mu$, as a Markov kernel, when there exists a Markov kernel $P : S \times \mathcal{B}_T \to [0, 1]\, P(s, A) = \mu(A)$ for all $s \in S$ and $A \in \mathcal{B}_T$.

Definition 2.4.10. Given a measure μ on S and a Markov kernel $P : S \times \mathcal{B}_T \to [0, 1]$, we can define the *pushforward* of μ under the Markov kernel P as:

$$P_*(\mu)(B) = \int_S P(s, B)\, \mu(ds), \quad \forall B \in \mathcal{B}_T.$$

Any measurable function $f : (S, \mathcal{B}_S) \to (T, \mathcal{B}_T)$ induces a simple Markov kernel $s \mapsto \delta_{f(s)}$. As a result, as introduced earlier, the usual definition of the pushforward measure via f becomes a special case of the general formulation defined above.

2.4.4 Spaces of Measures

The set of all finite measures on a measurable space $(S, \mathcal{B})$ will be denoted by $\mathcal{M}(S, \mathcal{B})$, or simply $\mathcal{M}S$ when the σ-algebra $\mathcal{B}$ is clear from context. In particular, $\mathcal{M}\mathbb{R}$ denotes the Banach space of finite Borel measures on $\mathbb{R}$.

$\mathcal{M}S$ forms a real vector space, where addition and scalar multiplication are defined pointwise. Specifically, for $B \in \mathcal{B}$, $\mu, \nu \in \mathcal{M}S$, and $\alpha \in \mathbb{R}$, the operations are given by

$$(\mu + \nu)(B) = \mu(B) + \nu(B), \quad (a\mu)(B) = \alpha\mu(B).$$

$\mathcal{M}S$ is also a normed space equipped with the total variation norm.

Definition 2.4.11. For a measure μ, the total variation norm is defined as

$$\|\mu\|_{TV} := \sup \left\{ \sum_{i=1}^{n} |\mu(B_i)| : \{B_1, \ldots, B_n\} \text{ is a finite measurable partition of } S \right\}.$$

For positive measures, this reduces to $\mu(S)$, and for probability measures, the norm is 1. The total variation norm turns $\mathcal{M}_S$ into a Banach space, meaning it is complete under this norm. The following alternative definition is useful to compute the total variation norm between measures.

Definition 2.4.12. [43] Let let P and Q be two probability measures on $\mathcal{M}\mathbb{R}$. Define

$$\nu = P + Q, \quad p = \frac{dP}{d\nu}, \quad q = \frac{dQ}{d\nu},$$

where $\frac{dP}{d\nu}$ denotes the the Radon–Nikodym derivative of the measure P with respect to the measure ν. It holds that:

$$\|P - Q\| = \sup_{A \in \mathcal{B}} \left\{ \left| \int_A (p - q)\, d\nu \right| \right\}$$

2.5 C* and W*-Algebras

2.6 Category theory

Category theory originated as an effort to connect and unify two distinct areas of mathematics. The goal was to study and classify specific geometric structures—such as topological spaces, manifolds, and bundles—by associating them with corresponding algebraic structures like groups, rings, and abelian groups. It became clear that a language was needed to connect geometric and algebraic objects—one not explicitly tailored to geometry or algebra. Only a language of such generality could allow meaningful discussion across both fields. This is the birth of category theory. Described as ”a language about nothing, and therefore about everything,” category theory provides a highly general way of discussing mathematical concepts. It was invented by Samuel Eilenberg and Saunders MacLane [44]. They organized various mathematical structures into categories called geometric others algebraic. To connect these categories, they defined functors, which map objects and morphisms from one category to another, much like functions do. They further introduced natural transformations, which provide a way to compare functors, translating the results of one functor into those of another. [45]

Só cenas básicas: Qual é o interesse, o que é uma cat, funtores, tranformações naturais, produto, coproducto, tensor, categoria monoidal simetria, adjoint, coherence theorem for symmetric monoidal categories

Chapter 3

Metric Lambda Calculus

The Lambda Calculus, developed by Church and Curry in the 1930s, serves as a formal language capturing the key attribute of higher-order functional languages, treating functions as first-class citizens, allowing them to be passed as arguments [46]. Moreover, lambda calculus has been proven to be universal in the sense that any computable function can be represented as an expression within the language [47] . Beyond its foundational aspects, this calculus incorporates extensions for modeling side effects, including probabilistic or nondeterministic behaviors and shared memory. Higher-order functions form a pivotal abstraction in practical programming languages such as LISP, Scheme, ML, and Haskell.
This chapter introduces the metric lambda calculus as presented in [24]. The metric lambda calculus integrates notions of approximation into the equational system of affine lambda calculus, a variant of lambda calculus that restricts each variable to being used at most once. The metric lambda calculus incorporates a metric equational system, enabling reasoning about approximate program equivalence. This chapter offers a brief insight into lambda calculus and an overview of the syntax and metric equational system of the metric lambda calculus. For a more detailed study of lambda calculus theory, the reader is referred to [46].

3.1 The Lambda Calculus

The concept of a function takes a central role in the lambda calculus. But what exactly is a function? In most mathematics, the “functions as graphs” paradigm the “functions as graphs” paradigm is the most elegant and appropriate framework for understanding functions. Within this paradigm, each function f has a fixed domain X and a fixed codomain Y. The function f is then a subset of $X \times Y$ that satisfies the property that for each $x \in X$ there is a unique $y \in Y$ such that $(x, y) \in f$. Two functions f and g are equal if they yield the same output on each input, that is if $f(x) = g(x)$ for all $x \in X$. This perspective is known as the extensional view of functions, as it emphasizes that the only observable property of a function is how it maps inputs to outputs.

On the other hand, the “functions as rules” paradigm is more appropriate within computer science. In this context, defining a function involves specifying a rule or procedure for computing the function. Such a rule is often expressed in the form of a formula, for example, $f(x) = x^2$. As with the mathematical paradigm, two functions are considered extensionally equal if they exhibit the same input-output behavior. However, this view also introduces the notion of intensional equality: two functions are intensionally equal if they are defined by (essentially) the same formula.

In the lambda calculus, functions are described explicitly as formulae. The function $f : x \mapsto f(x)$ is represented as $\lambda x.f(x)$. Applying a function to an argument is done by juxtaposing the two expressions. For instance consider the function $f : x \mapsto x + 1$, to compute $f(2)$ one writes $(\lambda x.x + 1)(2)$.

The expression of higher-order functions - functions whose inputs and/or outputs are themselves functions- in a simple manner is an essential feature of lambda calculus. For example, the composition operator $f, g \mapsto f \circ g$ is written as $\lambda f.\lambda g.\lambda x.f(g(x))$. Considering the functions $f : x \mapsto x^2$ and $g : x \mapsto x + 1$, to compute $(f \circ g)(2)$ one writes

$$(\lambda f.\lambda g.\lambda x.f(g(x)))(\lambda x.x^2)(\lambda x.x + 1)(2).$$

As mentioned above, within the “functions as rules” paradigm, is not always necessary to specify the domain and codomain of a function in advance. For instance, the identity function $f : x \mapsto x$, can have any set X as its domain and codomain, provided that the domain and codomain are the same. In this case, one says that f has type $X \to X$. In the case of the composition operator, $h = \lambda f.\lambda g.\lambda x.f(g(x))$, the domain and codomain of the functions f and g must match. Specifically, f can have any set X as its domain and any set Y as its codomain, provided that Y is the domain of g. Similarly, g can have any set Z as its codomain. Thus, h has type

$$(X \to Y) \to (Y \to Z) \to (X \to Z).$$

This flexibility regarding domains and codomains enables operations on functions that are not possible in ordinary mathematics. For instance, if $f = \lambda x.x$ is the identity function, then one has that $f(x) = x$ for any x. In particular, by substituting f for x, one obtains $f(f) =$

$(\lambda x.x)(f) = f$. Note that the equation $f(f) = f$ is not valid in conventional mathematics, as it is not permissible, due to set-theoretic constraints, for a function to belong to its own domain.

Nevertheless, this remarkable aspect of lambda calculus, this work focuses on a more restricted version of the lambda calculus, known as the simply-typed lambda calculus. Here, each expression is always assigned a type, which is very similar to the situation in mathematics. A function may only be applied to an argument if the argument's type aligns with the function's expected domain. Consequently, terms such as $f(f)$ are not allowed, even if f represents the identity function.

3.2 Syntax

The grammar and term formation rules of the affine lambda calculus, discussed in [24], are presented in this subsection.

3.2.1 Type system

As previously mentioned, this work focuses on the simply-typed lambda calculus, this work focuses on the simply-typed lambda calculus, where each lambda term is assigned a *type*. Unlike sets, types are *syntactic* objects, meaning they can be discussed independently of their elements. One can conceptualize types as names or labels for set. The definition of the grammar of types for affine lambda calculus is as follows, where G represents a set of ground types, is given by the following **Backus-Naur Form** (**BNF**) [48].

$$\mathbb{A} ::= X \in G \mid \mathbb{I} \mid \mathbb{A} \otimes \mathbb{A} \mid \mathbb{A} \multimap \mathbb{A} \tag{3.1}$$

Note that this is an inductive definition. Ground types are things such as booleans, integers, and so forth. The type $\mathbb{I}$ is the unit/empty type, which has only one element. The type $\mathbb{A} \otimes \mathbb{A}$ correponds to the tensor of two types, while the type $\mathbb{A} \multimap \mathbb{B}$ is the type of linear maps one type to another.

3.2.2 (Raw)Terms

The expressions of the lambda calculus are called lambda terms. In the simply-typed lambda calculus, each lambda term is assigned a type. The terms without the specification of a type are called *raw typed lambda terms*. The grammar of *raw typed lambda terms* is given by the **BNF** below.

$$\begin{aligned} v, v_1, \ldots, v_n, w \quad ::= \quad & x \mid f(v_1, \ldots, v_n) \mid \ * \ \mid (\lambda x : \mathbb{A}.v) \mid (vw) \mid v \otimes w \mid \\ & \text{pm } v \text{ to } x \otimes y.w \mid v \ \text{ to } \ * .w \mid \text{dis}(v) \end{aligned}$$

Here x ranges over an infinite set of variables. $f \in \Sigma$, where Σ corresponds to a class of sorted operation symbols, and $f(v_1, \ldots, v_n)$ corresponds to the aplication of the function f to the arguments $v_1, \ldots, v_n$. The symbol $*$ is the unit element of the type $\mathbb{I}$. The term $(\lambda x : \mathbb{A}.v)$ is the lambda abstraction term, which represents a function that takes an argument of type $\mathbb{A}$ and returns the value of v. The term (vw) is the application term, which applies the function v to the argument w. The term $v \otimes w$ is the tensor product of v and w. The term pm v to $x \otimes y.w$ is the pattern-matching construct, which is used to deconstruct a tensor product into components x and y. The term v to $*.w$ is used to discard a variable v of the unit type. The term dis(v) is the discard term, which is used to discard a term v.

Ver o que por antes do ::= porque v1,..., vn tb são termos

3.2.3 Free and Bound Variables

An occurrence of a variable x within a term of the form $\lambda x.v$ is referred to as *bound*. Similarly, the variables x and y in the term pm v to $x \otimes y.w$ are also bound. A variable occurrence that is not bound is said to be *free*. For example, in the term $\lambda x.xy$, the variable y is free, whereas the variable x is bound.

The set of free variables of a term v is denoted by $FV(v)$, and is defined inductively as follows:

$$\begin{aligned} FV(x) &= \{x\}, & FV(*) &= \emptyset, \\ FV(f(v_1, \ldots, v_n)), &= FV(v_1) \cup \ldots \cup FV(v_n) & FV(\lambda x : \mathbb{A}.v) &= FV(v) \backslash \{x\}, \\ FV(vw) &= FV(v) \cup FV(w), & FV(v \otimes w) &= FV(v) \cup FV(w), \\ FV(\text{pm } v \text{ to } x \otimes y.w), &= FV(v) \cup (FV(w) \backslash \{x, y\}) & FV(\text{dis}(v)) &= FV(v), \\ FV(v \text{ to } *.w) &= FV(v) \cup FV(w). \end{aligned}$$

3.2.4 Term formation rules

To prevent the formation of nonsensical terms within the context of lambda calculus, such as $(v \otimes w)(u)$, the *typing rules* are imposed.

A *typed* term is a pair consisting of a term and its corresponding type. The notation $v : \mathbb{A}$ denotes that the term v has type $\mathbb{A}$. Typing rules are formulated using *typing judgments*. A typing judgment is an expression of the form $x_1 : \mathbb{A}_1, \ldots, x_n : \mathbb{A}_n \triangleright v : \mathbb{A}$ (where $n \geq 1$), which asserts that the term v is a well-typed term of type $\mathbb{A}$ under the assumption that each variable variable x_i has type $\mathbb{A}_i$, for $1 \leq i \leq n$. The list $x_1 : \mathbb{A}_1, \ldots, x_n : \mathbb{A}_n$ of typed variables is called the *typing context* of the judgment, and it might be empty. Each variable x_i (where $1 \leq i \leq n$) must occur at most once in $x_1, \ldots, x_n$. The typing contexts are denoted by Greek letters Γ, Δ, E, and from now on, when referring to an abstract judgment, the notation $\Gamma \triangleright v : \mathbb{A}$ will be employed. The empty context is denoted by $-$. Note that in the affine lambda calculus, different contexts do not share variables. For example, if $\Gamma = x : \mathbb{A}, y : \mathbb{B}$ none of these variables can appear in any other context.

The concept of *shuffling* is employed to construct a linear typing system that ensures the admissibility of the exchange rule and enables unambiguous reference to judgment's denotation $[\![\Gamma \triangleright v : \mathbb{A}]\!]$. An admissible rule is not explicitly included in the formal definition of type theory, but its validity can be proven by demonstrating that whenever the premises can be derived, it is possible to construct a derivation of its conclusion. Shuffling is defined as a permutation of typed variables in a sequence of contexts, $\Gamma_1, \ldots, \Gamma_n$, preserving the relative order of variables within each Γ_i [49]. For instance, if $\Gamma_1 = x : \mathbb{A}, y : \mathbb{B}$ and $\Gamma_2 = z : \mathbb{D}$, then $z : \mathbb{D}, x : \mathbb{A}, y : \mathbb{B}$ is a valid shuffle of Γ_1, Γ_2. On the other hand, $y : \mathbb{B}, x : \mathbb{A}, z : \mathbb{D}$ is not a shuffle because it alters the occurrence order of x and y in Γ_1. The set of shuffles in $\Gamma_1, \ldots, \Gamma_n$ is denoted as Sf$(\Gamma_1, \ldots, \Gamma_n)$. A valid typing derivation is constructed using the inductive rules shown in Figure 2.

$$\frac{\Gamma_i \triangleright v_i : \mathbb{A}_i \quad f : \mathbb{A}_1, \ldots, \mathbb{A}_n \to \mathbb{A} \in \Sigma \quad E \in \mathsf{Sf}(\Gamma_1; \ldots; \Gamma_n)}{E \triangleright f(v_1, \ldots, v_n) : \mathbb{A}}\,(\mathsf{ax}) \qquad \frac{}{x : \mathbb{A} \triangleright x : \mathbb{A}}\,(\mathsf{hyp})$$

$$\frac{}{- \triangleright * : \mathbb{I}}\,(\mathbb{I}_i) \quad \frac{\Gamma \triangleright v : \mathbb{A} \otimes \mathbb{B} \quad \Delta, x : \mathbb{A}, y : \mathbb{B} \triangleright w : \mathbb{D} \quad E \in \mathsf{Sf}(\Gamma; \Delta)}{E \triangleright \mathsf{pm}\ v\ \mathsf{to}\ x \otimes y.w : \mathbb{D}}\,(\otimes_e) \quad \frac{\Gamma \triangleright v : \mathbb{A}}{\Gamma \triangleright \mathsf{dis}(v) : \mathbb{I}}\,(\mathsf{dis})$$

$$\frac{\Gamma \triangleright v : \mathbb{A} \quad \Delta \triangleright w : \mathbb{B} \quad E \in \mathsf{Sf}(\Gamma; \Delta)}{E \triangleright v \otimes w : \mathbb{A} \otimes \mathbb{B}}\,(\otimes_i) \qquad \frac{\Gamma \triangleright v : \mathbb{I} \quad \Delta \triangleright w : \mathbb{A} \quad E \in \mathsf{Sf}(\Gamma; \Delta)}{E \triangleright v\ \mathsf{to}\ *.w : \mathbb{A}}\,(\mathbb{I}_e)$$

$$\frac{\Gamma, x : \mathbb{A} \triangleright v : \mathbb{B}}{\Gamma \triangleright \lambda x : \mathbb{A}.v : \mathbb{A} \multimap \mathbb{B}}\,(\multimap_i) \qquad \frac{\Gamma \triangleright v : \mathbb{A} \multimap \mathbb{B} \quad \Delta \triangleright w : \mathbb{A} \quad E \in \mathsf{Sf}(\Gamma; \Delta)}{E \triangleright vw : \mathbb{B}}\,(\multimap_e)$$

Figure 2: Term formation rules of affine lambda calculus.

The rule (ax) states that if there is a function $f \in \Sigma$ that has type $\mathbb{A}_1, \ldots, \mathbb{A}_n \to \mathbb{A}$ and a set of variables $v_1, \ldots, v_n$ whose types match the type of the arguments of f, then if that function is applied to v_1, ..., v_n the respective result is of type $\mathbb{A}$. The rule (hyp) is a tautology: under the assumption that x has type $\mathbb{A}$, x has type $\mathbb{A}$. The rule ($\mathbb{I}_i$) asserts that the unit element $*$ always has type $\mathbb{I}$. The rule ($\multimap_i$) expresses that if v is a term of type $\mathbb{B}$ with a variable x of type $\mathbb{A}$, then $\lambda x : \mathbb{A}.v$ is a function of type $\mathbb{A} \multimap \mathbb{B}$. The rule $(\multimap_e)$ states that a function of type $\mathbb{A} \multimap \mathbb{B}$ can be applied to an argument of type $\mathbb{A}$ to produce a result of type $\mathbb{B}$. The rule $(\otimes_i)$ asserts that if there is a term v of type $\mathbb{A}$ and a term w of type $\mathbb{B}$, then the tensor of these terms is of type $\mathbb{A} \otimes \mathbb{B}$. The rule $(\otimes_e)$ expresses if there is a term w of type $\mathbb{D}$ with variables x and y of types $\mathbb{A}$ and $\mathbb{B}$, respectively, and a term v of type $\mathbb{A} \otimes \mathbb{B}$, then v can be deconstructed into $x \otimes y$. The rule $(\mathbb{I}_e)$ states that if there is a term w of type $\mathbb{A}$ and a term v of type $\mathbb{I}$, then v can be discarded, and only the term w remains. Finally, the rule (dis) asserts that a term v of type $\mathbb{A}$ can be discarded, resulting in a term of type $\mathbb{I}$.

For a better understanding of the rules, a few straightforward programming examples are provided. For instance, the program that swaps the elements of a tensor product can be written as follows:

$$x : \mathbb{A}, y : \mathbb{B} \triangleright \mathsf{pm}\ x \otimes y\ \mathsf{to}\ a \otimes b.b \otimes a : \mathbb{B} \otimes \mathbb{A}$$

Now, to prove that this program is well-typed one can write the following typing derivation:

$$\begin{array}{lll}
1 & x : \mathbb{A} \triangleright x : \mathbb{A} & (\text{hyp}) \\
2 & y : \mathbb{B} \triangleright y : \mathbb{B} & (\text{hyp}) \\
3 & x : \mathbb{A}, y : \mathbb{B} \triangleright x \otimes y : \mathbb{A} \otimes \mathbb{B} & (1, 2, \otimes_i) \\
4 & b : \mathbb{B} \triangleright b : \mathbb{B} & (\text{hyp}) \\
5 & a : \mathbb{A} \triangleright a : \mathbb{A} & (\text{hyp}) \\
6 & b : \mathbb{B}, a : \mathbb{A} \triangleright b \otimes a : \mathbb{B} \otimes \mathbb{A} & (4, 5, \otimes_i) \\
7 & x : \mathbb{A}, y : \mathbb{B} \triangleright \text{pm } x \otimes y \text{ to } a \otimes b . b \otimes a : \mathbb{B} \otimes \mathbb{A} & (3, 6, \otimes_e)
\end{array}$$

Observe that in the notation of the third column, the numbers correspond to the premises utilized in the application of the rule.

Another example is the function that recieves a tensor product and returns first element and discards the second:

$$- \triangleright \lambda x : \mathbb{A} \otimes \mathbb{B}.\text{pm } x \text{ to } a \otimes b.\text{dis}(b) \ \text{ to } * .a : \mathbb{A}$$

To prove that this program is well-typed one can write the following typing derivation:

$$\begin{array}{lll}
1 & b : \mathbb{B} \triangleright b : \mathbb{B} & (\text{hyp}) \\
2 & b : \mathbb{B} \triangleright \text{dis}(b) : \mathbb{I} & (1, \text{dis}) \\
3 & a : \mathbb{A} \triangleright a : \mathbb{A} & (\text{hyp}) \\
4 & a : \mathbb{A}, b : \mathbb{B} \triangleright \text{dis}(b) \ \text{ to } * .a & (2, 3, \mathbb{I}_e) \\
5 & x : \mathbb{A} \otimes \mathbb{B} \triangleright x : \mathbb{A} \otimes \mathbb{B} & (\text{hyp}) \\
6 & x : \mathbb{A} \otimes \mathbb{B} \triangleright \text{pm } x \text{ to } a \otimes b.\text{dis}(b) \ \text{ to } * .a : \mathbb{A} & (4, 5, \otimes I_e) \\
7 & - \triangleright \lambda x : \mathbb{A} \otimes \mathbb{B}.\text{pm } x \text{ to } a \otimes b.\text{dis}(b) \ \text{ to } * .a : \mathbb{A} & (6, \multimap_i)
\end{array}$$

Also fala-se de Type inference algorithm? Tipo existe...

3.2.5 α-equivalence

A natural notion of equivalence definition stems from the fact that terms that differ only in the names of their bound variables represent the same program. For instance, the functions $\lambda x : \mathbb{A}.x$ and $\lambda y : \mathbb{A}.y$ have the same input-output behavior, despite being represented by different lambda terms. This equivalence is called *α-equivalence*.

Definition 3.2.1. The α-equivalence is an equivalence relation on lambda terms that is used to rename bound variables. To rename a variable x as y in a term v, denoted by $v\{y/x\}$, is to replace all occurrences of x in v by y. Two terms v and w are α-equivalent, written $=_\alpha$, if one can be derived from the other by a series of changes of bound variables

Convention 3.2.2. Terms are considered up to α-equivalence from now on.

3.2.6 Substitution

The substitution of a variable x for a term w in a term v is denoted by $v[w/x]$. It is only permitted to replace free variables. For instance, $\lambda x.x\,[v/x]$ is $\lambda x.x$ and not $\lambda x.v$. Moreover, it is necessary to avoid the unintended binding of free variables. For example,

$$(\text{pm } x \otimes y \text{ to } a \otimes b.b \otimes a \otimes z)\,[z/\text{pm } c \otimes d \text{ to } e \otimes f.f \otimes e \otimes a]$$

is not the same as

$$\text{pm } x \otimes y \text{ to } a \otimes b.b \otimes a \otimes (\text{pm } c \otimes d \text{ to } e \otimes f.f \otimes e \otimes a).$$

Instead, the bounded variable a must be renamed before the substitution, and in this case, the proper substitution is

$$(\text{pm } x \otimes y \text{ to } t \otimes b.b \otimes t \otimes z)\,[z/\text{pm } c \otimes d \text{ to } e \otimes f.f \otimes e \otimes a]$$

which is equal to

$$\text{pm } x \otimes y \text{ to } t \otimes b.b \otimes t \otimes (\text{pm } c \otimes d \text{ to } e \otimes f.f \otimes e \otimes a).$$

Note that a simple way of ensuring these restrictions are satisfied is not allowing the variable x to occur in the context of w in $v[w/x]$. Since x is in the context of v, this is always the case in the affine lambda calculus.

Definition 3.2.3. Given the typings judgments $\Gamma, x : \mathbb{A} \triangleright v : \mathbb{B}$ and $\Delta \triangleright w : \mathbb{A}$, the substitution

$\Gamma, \Delta \triangleright v[w/x] : \mathbb{B}$ is defined below. The types of judgments are omitted as no ambiguity arises.

$$\begin{aligned}
\Gamma, \Delta \triangleright y[w/x] &= \Gamma, \Delta \triangleright y, \\
\Delta \triangleright *[w/x] &= \Delta \triangleright *, \\
\Gamma, \Delta \triangleright (\lambda y : \mathbb{B}.v)[w/x] &= \Gamma, \Delta \triangleright \lambda y : \mathbb{B}.v[w/x], \\
(\mathsf{dis}(v))[w/x] &= \mathsf{dis}(v[w/x]),
\end{aligned}$$

In the next three cases, $\Gamma, x : \mathbb{A} \in$Sf$(\Gamma_1, \dots, \Gamma_i, \dots, \Gamma_n)$ and $\Gamma_i \triangleright v_i$

$$\begin{aligned}
\Gamma, \Delta \triangleright (f(v_1, \dots, v_n))[w/x] &= \Gamma, \Delta \triangleright f(v_1[w/x], \dots, v_n), && (\text{ if } x : \mathbb{A} \in \Gamma_1) \\
\Gamma, \Delta \triangleright (f(v_1, \dots, v_i, \dots, v_n))[w/x] &= \Gamma, \Delta \triangleright f(v_1, \dots, v_i[w/x], \dots, v_n), && (\text{ if } x : \mathbb{A} \in T_i) \\
\Gamma, \Delta \triangleright (f(v_1, \dots, v_n))[w/x] &= \Gamma, \Delta \triangleright f(v_1, \dots, v_n[w/x]), && (\text{ if } x : \mathbb{A} \in \Gamma_n)
\end{aligned}$$

In the next two cases, $\Gamma, x : \mathbb{A} \in$ Sf$(\Gamma_1, \Gamma_2), \Gamma_1 \triangleright v$, and $\Gamma_2 \triangleright u$

$$\begin{aligned}
\Gamma, \Delta \triangleright (vu)[w/x] &= \Gamma, \Delta \triangleright (v[w/x]u), && (\text{ if } x : \mathbb{A} \in \Gamma_1) \\
\Gamma, \Delta \triangleright (vu)[w/x] &= \Gamma, \Delta \triangleright (vu[w/x]), && (\text{ if } x : \mathbb{A} \in \Gamma_2)
\end{aligned}$$

In the next two cases, $\Gamma, x : \mathbb{A} \in$ Sf$(\Gamma_1, \Gamma_2), \Gamma_1 \triangleright v$, and $\Gamma_2 \triangleright u$

$$\begin{aligned}
\Gamma, \Delta \triangleright (v \otimes u)[w/x] &= \Gamma, \Delta \triangleright v[w/x] \otimes u, && (\text{ if } x : \mathbb{A} \in \Gamma_1) \\
\Gamma, \Delta \triangleright (v \otimes u)[w/x] &= \Gamma, \Delta \triangleright v \otimes u[w/x], && (\text{ if } x : \mathbb{A} \in \Gamma_2)
\end{aligned}$$

In the next two cases, $\Gamma, x : \mathbb{A} \in$ Sf$(\Gamma_1, \Gamma_2), \Gamma_1 \triangleright v$, and $\Gamma_2, y : \mathbb{D}, z : \mathbb{E} \triangleright u$

$$\begin{aligned}
\Gamma, \Delta \triangleright (\mathsf{pm}\ v \ \mathsf{to}\ y \otimes z.u)[w/x] &= \Gamma, \Delta \triangleright \mathsf{pm}\ v[w/x] \ \mathsf{to}\ y \otimes z.u, && (\text{ if } x : \mathbb{A} \in \Gamma_1) \\
\Gamma, \Delta \triangleright (\mathsf{pm}\ v \ \mathsf{to}\ y \otimes z.u)[w/x] &= \Gamma, \Delta \triangleright \mathsf{pm}\ v \ \mathsf{to}\ y \otimes z.u[w/x], && (\text{ if } x : \mathbb{A} \in \Gamma_2)
\end{aligned}$$

In the next two cases, $\Gamma, x : \mathbb{A} \in$ Sf$(\Gamma_1, \Gamma_2), \Gamma_1 \triangleright v$, and $\Gamma_2 \triangleright u$

$$\begin{aligned}
\Gamma, \Delta \triangleright (v \ \mathsf{to}\ * .u)[w/x] &= \Gamma, \Delta \triangleright v[w/x] \ \mathsf{to}\ * .u && (\text{ if } x : \mathbb{A} \in \Gamma_1), \\
\Gamma, \Delta \triangleright (v \ \mathsf{to}\ * .u)[w/x] &= \Gamma, \Delta \triangleright v \ \mathsf{to}\ * .u[w/x] && (\text{ if } x : \mathbb{A} \in \Gamma_2).
\end{aligned}$$

The sequential substitutions $M[M_i/x_i] \dots [M_n/x_n]$ are writen as $M[M_i/x_i, \dots, M_n/x_n]$.

3.2.7 Properties

The calculus defined in Figure 2 possesses several desirable properties, which are listed below. Before proceeding, it is necessary to introduce some auxiliary notation. Given a context Γ, $te(\Gamma)$ denotes context Γ with all types erased. The expression $\Gamma \simeq_\pi \Gamma'$ denotes that the contexts Γ is a permutation of context Γ'.This notation also applies to non-repetitive lists of untyped variables $te(\Gamma)$. Additionally, a judgment $\Gamma \triangleright v : \mathbb{A}$ will often be abbreviated into $\Gamma \triangleright v$ or even just v when no ambiguities arise.

The properties are as follows:

1. for all judgements $\Gamma \triangleright v$ and $\Gamma' \triangleright v$, $\mathrm{te}(\Gamma) \simeq_\pi \mathrm{te}(\Gamma')$;
2. additionally if $\Gamma \triangleright v : \mathbb{A}, \Gamma' \triangleright v : \mathbb{A}'$, and $\Gamma \simeq_\pi \Gamma'$, then $\mathbb{A}$ must be equal to $\mathbb{A}'$;
3. all judgements $\Gamma \triangleright v : \mathbb{A}$ have a unique derivation.
4. (exchange) For every judgement $\Gamma, x : \mathbb{A}, y : \mathbb{B}, \Delta \triangleright v : \mathbb{D}$ it is possible to derive $\Gamma, y : \mathbb{B}, x : \mathbb{A}, \Delta \triangleright v : \mathbb{D}$.
5. (substitution) For all judgements $\Gamma, x : \mathbb{A} \triangleright v : \mathbb{B}$ and $\Delta \triangleright w : \mathbb{A}$ it is possible to derive $\Gamma, \Delta \triangleright v[w/x] : \mathbb{B}$.

3.2.8 Equations-in-context

The simply typed lambda calculus is a formal language that captures operations like the application of a function to an argument and the elimination of variables. To express these operations there is a set of rules known as reduction rules. These rules fall into two primary categories: the *β-reductions*, which perform operations and enforce the implicit meaning of the term, and *η-reductions*, which simplify terms by exploiting the extensionality of functions. There is also a secondary class of reductions known as *commuting conversions*, which serve to disambiguate terms that, while equivalent, have different representations. As a result, affine λ-calculus comes equipped with the so-called equations-in-context $\Gamma \triangleright v = w : \mathbb{A}$, depicted in Figure 3.

$$
\begin{array}{|clcl|}
\hline
(\beta) & \Gamma, \Delta \triangleright (\lambda x : \mathbb{A}.\, v)\, w = v[w/x] : \mathbb{B} & (\eta) & \Gamma \triangleright \lambda x : \mathbb{A}.(v x) = v : \mathbb{A} \multimap \mathbb{B} \\
(\beta_{\mathbb{I}_e}) & \Gamma \triangleright * \,\mathsf{to}\, * .\, v = v : \mathbb{A} & (\eta_{\mathbb{I}_e}) & \Delta, \Gamma \triangleright v \,\mathsf{to}\, * .\, w[*/z] = w[v/z] : \mathbb{A} \\
(\beta_{\otimes_e}) & \multicolumn{3}{c|}{E, \Gamma, \Delta \triangleright \mathsf{pm}\, v \otimes w \,\mathsf{to}\, x \otimes y.\, u = u[v/x, w/y] : \mathbb{A}} \\
(\eta_{\otimes_e}) & \multicolumn{3}{c|}{\Delta, \Gamma \triangleright \mathsf{pm}\, v \,\mathsf{to}\, x \otimes y.\, u[x \otimes y/z] = u[v/z] : \mathbb{A}} \\
(c_{\mathbb{I}_e}) & \multicolumn{3}{c|}{\Delta, \Gamma, E \triangleright u[v \,\mathsf{to}\, * .w/z] = v \,\mathsf{to}\, * .u[w/z] : \mathbb{A}} \\
(c_{\otimes_e}) & \multicolumn{3}{c|}{\Delta, \Gamma, E \triangleright u[\mathsf{pm}\, v \,\mathsf{to}\, x \otimes y.\, w/z] = \mathsf{pm}\, v \,\mathsf{to}\, x \otimes y.\, u[w/z] : \mathbb{A}} \\
(\eta_{\mathsf{dis}}) & \multicolumn{3}{c|}{x_1 : \mathbb{A}_1, \ldots, x_n : \mathbb{A}_n \triangleright v = \mathsf{dis}(x_1) \,\mathsf{to}\, * \ldots .\, \mathsf{dis}(x_{n-1}) \,\mathsf{to}\, *\; \mathsf{dis}(x_n) : \mathbb{I}} \\
\hline
\end{array}
$$

Figure 3: Equations-in-context for affine lambda calculus

It is evident that, for example, equation (β) enforces the meaning of $(\lambda x : \mathbb{A}.\ v)w$, which is interpreted as "v with w in place of x". The equation (η), on the other hand, is a simplification rule that states that a function that applies another function v to an argument x can be simplified to the function v itself. The remaining β e η equations follow similar reasoning. The commuting conversion $(c_{\mathbb{I}_e})$ expresses that substituting a variable z by a term that maps a term v to the unit element $*$ in a term w is equivalent to mapping a term v to the unit element $*$ and then replacing z by w. The other commuting conversion has a similar interpretation.

3.3 Metric equational system

Mete-se contexto e tipo nas eqs métricas?

Metric equations [50], [51] are a strong candidate for reasoning about approximate program equivalence. These equations take the form of $t =_\epsilon s$, where ϵ is a non-negative rational representing the "maximum distance" between the two terms t and s. The metric equational system for linear lambda calculus is depicted in Figure 4.

$$\frac{}{v =_0 v}\,\text{(refl)} \qquad \frac{v =_q w \quad w =_r u}{v =_{q+r} u}\,\text{(trans)} \qquad \frac{v =_q w \quad r \geq q}{v =_r w}\,\text{(weak)}$$

$$\frac{\forall r > q.\ v =_r w}{v =_q w}\,\text{(arch)} \qquad \frac{\forall i \leq n.\ v =_{q_i} w}{v =_{\wedge q_i} w}\,\text{(join)} \qquad \frac{v =_q w}{w =_q v}\,\text{(sym)}$$

$$\frac{v =_q w \quad v' =_r w'}{v \otimes v' =_{q+r} w \otimes w'} \qquad \frac{\forall i \leq n.\ v_i =_{q_i} w_i}{f(v_1, ..., v_n) =_{\Sigma q_i} f(w_1, ..., , w_n)} \qquad \frac{v =_q w}{\lambda x : \mathbb{A}.\ v =_q \lambda x : \mathbb{A}.\ w}$$

$$\frac{v =_q w \quad v' =_r w'}{\text{pm } v \text{ to } x \otimes y.\ v' =_{q+r} \text{pm } w \text{ to } x \otimes y.\ w'} \qquad \frac{v =_q w}{\text{dis}(v) =_q \text{dis}(w)} \qquad \frac{v =_q w \quad v' =_r w'}{v v' =_{q+r} w w'}$$

$$\frac{\Gamma \triangleright v =_q w : \mathbb{A} \quad \Delta \in \text{perm}(\Gamma)}{\Delta \triangleright v =_q w : \mathbb{A}} \qquad \frac{v =_q w \quad v' =_r w'}{v \text{ to } *.v' =_{q+r} w \text{ to } *.w'} \qquad \frac{v =_q w \quad v' =_r w'}{v[v'/x] =_{q+r} w[w'/x]}$$

Figure 4: Metric equational system

Here, perm(Γ) denotes the set of possible permutions of context Γ. The rules (refl), (trans), and (sym) generalize the properties of reflexivity, transitivity, and symmetry of equality.

Rule (weak) asserts that if two terms are at a maximum distance q from each other, then they are also separated by any $r \geq q$. Rule (arch) states that if $v =_r w$ for all approximations r of q, then it necessarily follows that $v =_q w$. The rule (join) expresses that if several maximum distances between two terms are known, the actual maximum distance between them is the minimum of these distances. The rule that follows conveys that if the maximum distance between two terms v and w is q, and the maximum distance between terms v' and w' is r, then the maximum distance between the tensor products $v \otimes v'$ and $w \otimes w'$ is $q + r$. The remaining rules follow similar reasoning.

To ilustrate the usefulness of these equations, consider the program P that recieves a tensor product, swaps its elements and then applies a function f to to the new second element of the tensor pair:

$$P = x : \mathbb{A}, y : \mathbb{B} \triangleright \text{pm } x \otimes y \text{ to } a \otimes b.b \otimes f(a) : \mathbb{D} \otimes \mathbb{A}$$

Now, consider the case where f is an idealized version of function f^{ϵ} mapping a to $f(a)^{\epsilon}$. The program that applies the “real” function f to the first element of the tensor pair is P^{ϵ}:

$$P^{\epsilon} = x : \mathbb{A}, y : \mathbb{B} \triangleright \text{pm } x \otimes y \text{ to } a \otimes b.b \otimes f(a)^{\epsilon} : \mathbb{D} \otimes \mathbb{A}$$

Knowing that $f(a)^{\epsilon} =_{\epsilon} f(a)$, it is possible to show that $P^{\epsilon} =_{\epsilon} P$ using the metric equational system. The prove is as follows. The types and contexts are omitted for brevity as no ambiguity arises.

1 $f(a)^{\epsilon} =_{\epsilon} f(a)$

2 $b =_0 b$ (refl)

3 $b \otimes f(a)^{\epsilon} =_{\epsilon} b \otimes f(a)$ $(1, 2, \otimes_i)$

4 $x \otimes y =_0 x \otimes y$ (refl)

5 pm $x \otimes y$ to $a \otimes b.b \otimes f(a)^{\epsilon} =_{\epsilon}$ pm $x \otimes y$ to $a \otimes b.b \otimes f(a)$ $(3, 4, \otimes_e)$

3.4 Interpretation

Interpretação geral com categorias

3.4.1 Quantum Lambda Calculus

Quantum lambda calculus integrates quantum computation with higher-order functions, thereby emerging as a powerful tool for formal reasoning about quantum programs within a functional programming framework. This functional paradigm, with a static type system, offers the significant advantage of ensuring the absence of run-time errors, *i.e.*, potential errors can be detected at compile-time, when the program is written, rather than during execution.

The principal distinction between the quantum lambda calculus introduced in this section and the formulation proposed by Selinger [52, 53] lies in the handling of data duplication. In this aproach, as dictated by the type system in Figure 2, duplication of any data is strictly prohibited. In contrast, Selinger's approach permits the duplication of classical data while strictly forbidding the duplication of quantum data.

The syntax of the metric lambda calculus was presented in Chapter 3. Now, it is necessary to define the *model* of the "reality" of interest, *i.e.*, what the terms of the calculus mean within the "reality" considered. In the case of quantum lambda calculus, the model is based on quantum channels. The interpretation introduced in this subsection is based on the work of [24] with the addition of the measurement operation.

In order to define the interpretation of judgments $\Gamma \triangleright v : \mathbb{A}$, it is necessary to establish some notation first. Considering $v \in V$, $w \in W$, and $r \in R$ where V, W, R represent vector spaces, $\mathsf{sw}_{V,W} : V \otimes W \to W \otimes V$, denotes the swap operator, defined as $\mathsf{sw}_{V,W} = v \otimes w \mapsto w \otimes v$; $\lambda_V : \mathbb{C} \otimes V \to V$ is the left unitor defined as $\lambda_V = 1 \otimes v \mapsto v$; $\rho_V : V \otimes \mathbb{C} \to V$ is the right unitor defined as $\rho_V = v \otimes 1 \mapsto v$; and $\alpha_{V,W,R} : V \otimes (W \otimes R) \to (V \otimes W) \otimes R$ is the left associator, defined as $\alpha_{V,W,R} = v \otimes (w \otimes r) \mapsto (v \otimes w) \otimes r$. Moreover, for all operators $f : V \otimes W \to R$, the operator $\overline{f} : V \to (W \multimap R)$ denotes the corresponding curried version, defined as $\overline{f}(v) = w \mapsto f(v, w)$. On the other hand, the application operator denoted $\mathsf{app} : (V \multimap W) \otimes V \to W$, is defined as $f\,v = w$, where $f : V \multimap W$. The subscripts in these operators will be omitted unless ambiguity arises.

For all ground types $X \in G$ the interpretation of $[\![X]\!]$ is postulated as a vector space V. Types are interpreted inductively using the unit $\mathbb{I}$, the tensor $\otimes$, and the linear map $\multimap$. Given a non-empty context $\Gamma = \Gamma', x : \mathbb{A}$, its interpretation is defined by $[\![\Gamma', x : \mathbb{A}]\!] = [\![\Gamma']\!] \otimes [\![\mathbb{A}]\!]$ if Γ' is non-empty and $[\![\Gamma', x : \mathbb{A}]\!] = [\![\mathbb{A}]\!]$ otherwise. The empty context $-$ is interpreted as $[\![-]\!] = [\![\mathbb{I}]\!] = \mathbb{C}$. Given $X_1, ..., X_n \in V$, the n-tensor $(\ldots(X_1 \otimes X_2) \otimes \ldots) \otimes X_n$ is denoted as $X_1 \otimes \ldots \otimes X_n$, and similarly for operators.

“Housekeeping” operators are employed to handle interactions between context interpretation and the vectorial model. Given $\Gamma_1, \ldots, \Gamma_n$, the operator that splits $\llbracket \Gamma_1, \ldots, \Gamma_n \rrbracket$ into $\llbracket \Gamma_1 \rrbracket \otimes \ldots \otimes \llbracket \Gamma_n \rrbracket$ is denoted by $\mathsf{sp}_{\Gamma_1;\ldots;\Gamma_n} : \llbracket \Gamma_1, \ldots, \Gamma_n \rrbracket \to \llbracket \Gamma_1 \rrbracket \otimes \ldots \otimes \llbracket \Gamma_n \rrbracket$. For $n = 1$, $\mathsf{sp}_{\Gamma_1} = \mathsf{id}$. Let Γ_1 and Γ_2 be two contexts, $\mathsf{sp}_{\Gamma_1,\Gamma_2} \to \Gamma_1 \otimes \Gamma_2$ is defined as:

$$\mathsf{sp}_{-;\Gamma_2} = \lambda^{-1} \qquad \mathsf{sp}_{\Gamma_1;-} = \rho^{-1} \qquad \mathsf{sp}_{\Gamma_1;x:\mathbb{A}} = I \qquad \mathsf{sp}_{\Gamma_1;\Delta,x:\mathbb{A}} = \alpha \cdot (\mathsf{sp}_{\Gamma_1;\Delta} \otimes I)$$

For $n > 2$, $\mathsf{sp}_{\Gamma_1;\ldots;\Gamma_n}$ is is defined recursively based on the previous definition, using induction on n:

$$\mathsf{sp}_{\Gamma_1;\ldots;\Gamma_n} = (\mathsf{sp}_{\Gamma_1;\ldots;\Gamma_{n-1}} \otimes I) \cdot \mathsf{sp}_{\Gamma_1,\ldots,\Gamma_{n-1};\Gamma_n}$$

On the other hand, $\mathsf{jn}_{\Gamma_1;\ldots;\Gamma_n}$ denotes the inverse of $\mathsf{sp}_{\Gamma_1;\ldots;\Gamma_n}$. Next, given $\Gamma, x : \mathbb{A}, y : \mathbb{B}, \Delta$, the operator permuting x and y is denoted by $\mathsf{exch}_{\Gamma,x:\mathbb{A},y:\mathbb{B},\Delta} : \llbracket \Gamma, \underline{x : \mathbb{A}, y : \mathbb{B}}, \Delta \rrbracket \to \llbracket \Gamma, y : \mathbb{B}, x : \mathbb{A}, \Delta \rrbracket$ and defined as:

$$\mathsf{exch}_{\Gamma,\underline{x:\mathbb{A},y:\mathbb{B}},\Delta} = \mathsf{jn}_{\Gamma;y:\mathbb{B},x:\mathbb{A};\Delta} \cdot (I \otimes \mathsf{sw} \otimes I) \cdot \mathsf{sp}_{\Gamma;x:\mathbb{A},y:\mathbb{B};\Delta}$$

The shuffling operator $\mathsf{sh}_E : \llbracket E \rrbracket \to \llbracket \Gamma_1, \ldots, \Gamma_n \rrbracket$ is defined as a suitable composition of exchange operators.

For every operation symbol $f : \mathbb{A}_1, \ldots, \mathbb{A}_n \to \mathbb{A}$ it is assumed the existence of an operator $\llbracket f \rrbracket : \llbracket \mathbb{A}_1 \rrbracket \otimes \ldots \otimes \llbracket \mathbb{A}_n \rrbracket \to \llbracket \mathbb{A} \rrbracket$. The interpretation of judgments is defined by induction over derivations according to the rules in Figure 5 [24].

$$\frac{\llbracket \Gamma_i \rhd v_i : \mathbb{A}_i \rrbracket = m_i \quad f : \mathbb{A}_1, \ldots, \mathbb{A}_n \to \mathbb{A} \in \Sigma \quad E \in \mathsf{Sf}(\Gamma_1; \ldots; \Gamma_n)}{\llbracket E \rhd f(v_1, \ldots, v_n) : \mathbb{A} \rrbracket = \llbracket f \rrbracket \cdot (m_1 \otimes \ldots \otimes m_n) \cdot \mathsf{sp}_{\Gamma_1;\ldots;\Gamma_n} \cdot \mathsf{sh}_E} \qquad \frac{}{\llbracket x : \mathbb{A} \rhd x : \mathbb{A} \rrbracket = \mathsf{I}_{\llbracket \mathbb{A} \rrbracket}}$$

$$\frac{}{\llbracket - \rhd * : \mathbb{I} \rrbracket = \mathsf{I}_{\llbracket \mathbb{I} \rrbracket}} \qquad \frac{\llbracket \Gamma \rhd v : \mathbb{A} \otimes \mathbb{B} \rrbracket = m \quad \llbracket \Delta, x : \mathbb{A}, y : \mathbb{B} \rhd w : \mathbb{D} \rrbracket = n \quad E \in \mathsf{Sf}(\Gamma; \Delta)}{\llbracket E \rhd \mathsf{pm}\ v\ \mathsf{to}\ x \otimes y.w : \mathbb{D} \rrbracket = n \cdot \mathsf{jn}_{\Delta;\mathbb{A};\mathbb{B}} \cdot \alpha \cdot \mathsf{sw} \cdot (m \otimes \mathsf{I}) \cdot \mathsf{sp}_{\Gamma;\Delta} \cdot \mathsf{sh}_E}$$

$$\frac{\llbracket \Gamma \rhd v : \mathbb{A} \rrbracket = m \quad \llbracket \Delta \rhd w : \mathbb{B} \rrbracket = n \quad E \in \mathsf{Sf}(\Gamma; \Delta)}{\llbracket E \rhd v \otimes w : \mathbb{A} \otimes \mathbb{B} \rrbracket = (m \otimes n) \cdot \mathsf{sp}_{\Gamma;\Delta} \cdot \mathsf{sh}_E}$$

$$\frac{\llbracket \Gamma \rhd v : \mathbb{I} \rrbracket = m \quad \llbracket \Delta \rhd w : \mathbb{A} \rrbracket = n \quad E \in \mathsf{Sf}(\Gamma; \Delta)}{\llbracket E \rhd v\ \mathsf{to}\ *.w : \mathbb{A} \rrbracket = n \cdot \lambda \cdot (m \otimes \mathsf{I}) \cdot \mathsf{sp}_{\Gamma;\Delta} \cdot \mathsf{sh}_E} \qquad \frac{\llbracket \Gamma, x : \mathbb{A} \rhd v : \mathbb{B} \rrbracket = m}{\llbracket \Gamma \rhd \lambda x : \mathbb{A}. v : \mathbb{A} \multimap \mathbb{B} \rrbracket = \overline{m \cdot \mathsf{jn}_{\Gamma;\mathbb{A}}}}$$

$$\frac{\llbracket \Gamma \rhd v : \mathbb{A} \multimap \mathbb{B} \rrbracket = m \quad \llbracket \Delta \rhd w : \mathbb{A} \rrbracket = n \quad E \in \mathit{Sf}(\Gamma; \Delta)}{\llbracket E \rhd vw : \mathbb{B} \rrbracket = \mathsf{app} \cdot (m \otimes n) \cdot \mathsf{sp}_{\Gamma;\Delta} \cdot \mathsf{sh}_E} \qquad \frac{\llbracket \Gamma \rhd v : \mathbb{A} \rrbracket = f}{\llbracket \Gamma \rhd \mathsf{dis}(v) : \mathbb{I} \rrbracket = \mathsf{Tr} \cdot f}$$

Figure 5: Judgment interpretation

The following diagrams are useful to better understand the interpretation of judgements given in Figure 5.

$$
\begin{array}{lll}
\llbracket \mathsf{ax} \rrbracket : & & \llbracket E \rrbracket \xrightarrow{\mathsf{sh}_E} \llbracket \Gamma_1, \ldots, \Gamma_n \rrbracket \xrightarrow{\mathsf{sp}_{\Gamma;\Delta}} \llbracket \Gamma_1 \rrbracket \otimes \ldots \otimes \llbracket \Gamma_n \rrbracket \\
 & & \xrightarrow{m_1 \otimes \ldots \otimes m_n} \llbracket \mathbb{A}_1 \rrbracket \otimes \ldots \otimes \llbracket \mathbb{A}_n \rrbracket \xrightarrow{\llbracket f \rrbracket} \llbracket \mathbb{A} \rrbracket \\
\llbracket \mathsf{hyp} \rrbracket : & & \llbracket \mathbb{A} \rrbracket \xrightarrow{I_{\llbracket \mathbb{A} \rrbracket}} \llbracket \mathbb{A} \rrbracket \\
\llbracket \mathbb{I}_i \rrbracket : & & \llbracket \mathbb{I} \rrbracket \xrightarrow{I_{\llbracket \mathbb{I} \rrbracket}} \llbracket \mathbb{I} \rrbracket \\
\llbracket \otimes_e \rrbracket : & & \llbracket E \rrbracket \xrightarrow{\mathsf{sh}_E} \llbracket \Gamma, \Delta \rrbracket \xrightarrow{\mathsf{sp}_{\Gamma;\Delta}} \llbracket \Gamma \rrbracket \otimes \llbracket \Delta \rrbracket \xrightarrow{m \otimes \mathsf{I}} (\llbracket \mathbb{A} \rrbracket \otimes \llbracket \mathbb{B} \rrbracket) \otimes \llbracket \Delta \rrbracket \\
 & & \xrightarrow{\mathsf{sw}} \llbracket \Delta \rrbracket \otimes (\llbracket \mathbb{A} \rrbracket \otimes \llbracket \mathbb{B} \rrbracket) \xrightarrow{\alpha} (\llbracket \Delta \rrbracket \otimes \llbracket \mathbb{A} \rrbracket) \otimes \llbracket \mathbb{B} \rrbracket \xrightarrow{\mathsf{jn}_{\Delta;\mathbb{A};\mathbb{B}}} \llbracket \Delta, \mathbb{A}, \mathbb{B} \rrbracket \\
 & & \xrightarrow{n} \llbracket \mathbb{D} \rrbracket \\
\llbracket \otimes_i \rrbracket : & & \llbracket E \rrbracket \xrightarrow{\mathsf{sh}_E} \llbracket \Gamma, \Delta \rrbracket \xrightarrow{\mathsf{sp}_{\Gamma;\Delta}} \llbracket \Gamma \rrbracket \otimes \llbracket \Delta \rrbracket \xrightarrow{m \otimes n} \llbracket \mathbb{A} \rrbracket \otimes \llbracket \mathbb{B} \rrbracket \\
\llbracket \mathbb{I}_e \rrbracket : & & \llbracket E \rrbracket \xrightarrow{\mathsf{sh}_E} \llbracket \Gamma, \Delta \rrbracket \xrightarrow{\mathsf{sp}_{\Gamma;\Delta}} \llbracket \Gamma \rrbracket \otimes \llbracket \Delta \rrbracket \xrightarrow{m \otimes \mathsf{I}} \llbracket \mathbb{I} \rrbracket \otimes \llbracket \Delta \rrbracket \xrightarrow{\lambda} \llbracket \Delta \rrbracket \xrightarrow{n} \llbracket \mathbb{A} \rrbracket \\
\llbracket \multimap_i \rrbracket : & \llbracket \Gamma \rrbracket \otimes \llbracket \mathbb{A} \rrbracket & \xrightarrow{\overline{m \cdot \mathsf{jn}_{\Gamma;\mathbb{A}}}} \llbracket \mathbb{A} \rrbracket \multimap \llbracket \mathbb{B} \rrbracket \qquad\qquad (\llbracket \Gamma \rrbracket \otimes \llbracket \mathbb{A} \rrbracket \xrightarrow{\mathsf{jn}_{\Gamma;\mathbb{A}}} \llbracket \Gamma, \mathbb{A} \rrbracket \xrightarrow{m} \llbracket \mathbb{B} \rrbracket) \\
\llbracket \multimap_e \rrbracket : & & \llbracket E \rrbracket \xrightarrow{\mathsf{sh}_E} \llbracket \Gamma, \Delta \rrbracket \xrightarrow{\mathsf{sp}_{\Gamma;\Delta}} \llbracket \Gamma \rrbracket \otimes \llbracket \Delta \rrbracket \xrightarrow{m \otimes n} \llbracket \mathbb{A} \rrbracket \multimap \llbracket \mathbb{B} \rrbracket \otimes \llbracket \mathbb{A} \rrbracket \xrightarrow{\mathsf{app}} \llbracket \mathbb{B} \rrbracket \\
\llbracket \mathsf{dis} \rrbracket : & & \llbracket \Gamma \rrbracket \xrightarrow{f} \llbracket \mathbb{A} \rrbracket \xrightarrow{\mathsf{Tr}} \llbracket \mathbb{I} \rrbracket
\end{array}
$$

Regarding the interpretation of the exhange and substitution properties, for any judgements $\Gamma, x : \mathbb{A}, y : \mathbb{B}, \Delta \rhd v : \mathbb{D}$, $\Gamma, x : \mathbb{A} \rhd v : \mathbb{B}$, and $\Delta \rhd w : \mathbb{A}$, the following equations hold:

$$
\begin{aligned}
\llbracket \Gamma, x : \mathbb{A}, y : \mathbb{B}, \Delta \rhd v : \mathbb{D} \rrbracket &= \llbracket \Gamma, y : \mathbb{B}, x : \mathbb{A}, \Delta \rhd v : \mathbb{D} \rrbracket \cdot \mathsf{exch}_{\Gamma,\underline{x:\mathbb{A},y:\mathbb{B}},\Delta} \\
\llbracket \Gamma, \Delta \rhd v[w/x] : \mathbb{B} \rrbracket &= \llbracket \Gamma, x : \mathbb{A} \rhd v : \mathbb{B} \rrbracket \cdot \mathsf{jn}_{\Gamma;\mathbb{A}} \cdot (\mathsf{I} \otimes \llbracket \Delta \rhd w : \mathbb{A} \rrbracket) \cdot \mathsf{sp}_{\Gamma;\Delta}
\end{aligned}
\tag{3.2}
$$

Types and operations in quantum lambda calculus

In the case of quantum lambda calculus, it is natural to consider a type *qbit* of quantum bits. The interpretation of this type is defined as $\llbracket$*qbit*$\rrbracket = \mathbb{C}^{2\times 2}$.

The following operations are considered: the creation of a new qubit in the state $|0\rangle$, *new* $0 : \mathbb{I} \multimap$ *qbit*, the creation of a new qubit in the state $|1\rangle$ (*new* $1 : \mathbb{I} \multimap$ *qbit*), measuring a qubit, *meas* : *qbit* $\multimap$ *qbit*, applying a unitary operation to a qubit, *U* : *qbit*$^{\otimes n}$ $\multimap$ *qbit*$^{\otimes n}$, and performing a **CPTP** operation on a qubit, *CPTP* : *qbit*$^{\otimes n}$ $\multimap$ *qbit*$^{\otimes n}$. These operations are defined in Figure 6.

$$\begin{aligned} [\![\mathit{new}\,0]\!] &: \mathbb{C} \multimap [\![\mathit{bit}]\!] \\ 1 &\mapsto \left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right) \end{aligned} \qquad \begin{aligned} [\![\mathit{new}\,1]\!] &: \mathbb{C} \multimap [\![\mathit{bit}]\!] \\ 1 &\mapsto \left(\begin{smallmatrix}0&0\\0&1\end{smallmatrix}\right) \end{aligned} \qquad \begin{aligned} [\![\mathit{meas}]\!] &: [\![\mathit{qbit}]\!] \multimap [\![\mathit{bit}]\!] \\ \rho &\mapsto M_0\rho M_0^\dagger + M_1\rho M_1^\dagger \end{aligned}$$

$$\begin{aligned} [\![U]\!] &: [\![\mathit{qbit}]\!]^{\otimes n} \multimap [\![\mathit{qbit}]\!]^{\otimes n} \\ \rho &\mapsto U\rho U^\dagger \end{aligned} \qquad \begin{aligned} [\![\mathit{CPTP}]\!] &: [\![\mathit{qbit}]\!]^{\otimes n} \multimap [\![\mathit{qbit}]\!]^{\otimes n} \\ \rho &\mapsto \mathit{CPTP}(\rho) \end{aligned}$$

Figure 6: Interpretation of the operations in quantum lambda calculus.

The chosen model requires that all operators considered are **CPTP** maps. The identity I, the swap operator sw, the left and right unitors ρ and λ, the left associator α,the trace operation !, the split operator sp, the join operator jn, the exhange exch and shuffle sh are all **CPTP** maps [24]. However, the operators curry $\overline{f}$ and application app are not **CPTP** maps, which means that they are not allowed in this model. On the other hand the measurement operator $[\![\mathit{meas}]\!]$ and the unitary operator $[\![U]\!]$ are **CPTP** maps (Section 2.2.5, [12, page 73]).

Proposition 3.4.1. *The operators $[\![\mathit{new}\,0]\!]$ and $[\![\mathit{new}\,1]\!]$ are trace-preserving completely positive super-operators.*

Proof. It is straightforward to verify that the operators $[\![\mathit{new}\,0]\!]$ and $[\![\mathit{new}\,1]\!]$ are trace-preserving. Given that matrices $\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)$ and $\left(\begin{smallmatrix}0&0\\0&1\end{smallmatrix}\right)$ correspond to the projectors $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$, one has that

$$\langle\psi|\,|0\rangle\langle 0| \otimes I^{\otimes n}\,|\psi\rangle = \langle\psi|\,|0\rangle\langle 0| \otimes \left(\frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2}\right)^{\otimes n} |\psi\rangle \tag{3.3}$$

$$= \sum_i |\alpha_i|^2 \langle\psi_{0i}|0\rangle\langle 0|\psi_{0i}\rangle \langle\psi_{1i}| \left(\frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2}\right) |\psi_{1i}\rangle \dots \langle\psi_{ni}| \left(\frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2}\right) |\psi_{ni}\rangle \tag{3.4}$$

$$= \sum_i |\alpha_i|^2 |\langle 0|\psi_{0i}\rangle|^2 \left(\frac{(|\langle 0|\psi_{1i}\rangle|^2 + |\langle 1|\psi_{1i}\rangle|^2)\dots(|\langle 0|\psi_{ni}\rangle|^2 + |\langle 1|\psi_{ni}\rangle|^2)}{2}\right) \tag{3.5}$$

$$\tag{3.6}$$

Considering that the squared absolute value of a real number is always non-negative, the expression above is always non-negative. Therefore, the operator $[\![\mathit{new}\,0]\!]$ is positive. The same reasoning can be applied to the operator $[\![\mathit{new}\,1]\!]$. Therefore, attending to Definition 2.2.5 and Definition 2.2.4, it is possible to conclude that both operators $[\![\mathit{new}\,0]\!]$ and $[\![\mathit{new}\,1]\!]$ are completely positive super-operators. As a result, both operators are **CPTP** maps.

□

Falar que aqui vamos usar a categoria do paper do professor Renato (que não é fechada) e q lá se prova que a cat e o functor $\otimes$ são met enriched .

3.4.2 Example: Deutsch's Algorithm

Preciso de alterar o erro -> coloacr um bit flip antes da medição (As contas estão erradas (estão para a trace norm) e com a diamond norm este exemplo ficaria muito complicado em termos de contas)

In 1985, David Deutsch presented an algorithm that determines whether a function f is constant for a single-bit input (*i.e.*, either equal to 1 for all x or equal to 0 for all x) or balanced (*i.e.*, equal to 1 for half of the values of x and equal to 0 for the other half) [54]. Classically, to determine which case holds requires running f twice. Quantumly, it suffices to run f once. The Deutsch-Jozsa Algorithm is a simple example of a quantum algorithm that outperforms its classical counterpart. The algorithm is based on the concept of a quantum oracle, which is a black box that implements a unitary transformation U_f such that $U_f\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x)\rangle$, where $\oplus$ denotes addition modulo 2. The quantum circuit implementing Deutsch's algorithm is presented in Figure 7.

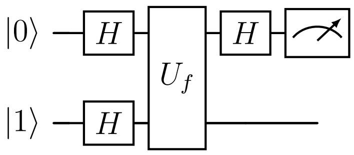


Figure 7: Quantum circuit implementing Deutsch's algorithm

Using lambda calculus, the Deutsch-Jozsa Algorithm can be expressed as:

$$\text{Deutsch} = -\triangleright \text{ pm } U_f(H\,(\textit{new}\,0\,(*)) \otimes H\,(\textit{new}\,1\,(*)))\ \textit{to}\ q_1 \otimes q_2\,.$$
$$\textit{meas}(H(q_1)) \otimes q_2 : \textit{qbit} \otimes \textit{qbit}$$

The proper interpretation of this term will be provided, but before that, an interpretation based on the quantum circuit in Figure 7 will be presented.

Attending to the circuit in Figure 7, one has that

$$\begin{aligned} & |0\rangle \otimes |1\rangle \\ \xmapsto{H\otimes H} \quad & \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |-\rangle \end{aligned} \tag{3.7}$$

With respecto to quantum oracle U_f, it is possible to show that:

$$\begin{aligned} & |x\rangle \otimes |-\rangle = |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}(|x\rangle \otimes |0\rangle - |x\rangle \otimes |1\rangle)) \\ \xmapsto{U_f} \; & \frac{1}{\sqrt{2}}(|x\rangle \otimes |0 \oplus f(x)\rangle - |x\rangle \otimes |1 \oplus f(x)\rangle) && \{\text{Defn. of } U_f\} \\ = \; & \frac{1}{\sqrt{2}}(|x\rangle\,|f(x)\rangle - |x\rangle\,|\neg f(x)\rangle) && \{0 \oplus x = x, 1 \oplus x = \neg x\} \\ = \; & \frac{1}{\sqrt{2}}(|x\rangle \otimes (|f(x)\rangle - |\neg f(x)\rangle)) \end{aligned} \tag{3.8}$$

Proceding by case distinction:

$$\frac{1}{\sqrt{2}}(|x\rangle \otimes (|f(x)\rangle - |\neg f(x)\rangle) = \begin{cases} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & \text{if } f(x) = 0 \\ |x\rangle \otimes \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) & \text{if } f(x) = 1 \end{cases} \tag{3.9}$$

It follows that:

$$|x\rangle \otimes \frac{1}{\sqrt{2}}(|f(x)\rangle - |\neg f(x)\rangle) = (-1)^{f(x)}\,|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = (-1)^{f(x)}\,|x\rangle \otimes |-\rangle \tag{3.10}$$

Returning to the interpretation of the Deutsch Algorithm, one has that:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |-\rangle \tag{3.11}$$

$$\xmapsto{U_f} \quad \frac{1}{\sqrt{2}}(U_f\,|0\rangle \otimes |-\rangle + U_f\,|1\rangle \otimes |-\rangle)) \tag{3.12}$$

$$= \quad \frac{1}{\sqrt{2}}((-1)^{f(0)}\,|0\rangle \otimes |-\rangle + (-1)^{f(1)}\,|1\rangle \otimes |-\rangle) \tag{3.13}$$

$$= \quad \begin{cases} (\pm 1)\,|+\rangle \otimes |-\rangle & \text{if } f(0) = f(1) \\ (\pm 1)\,|-\rangle \otimes |-\rangle & \text{if } f(0) \neq f(1) \end{cases} \tag{3.14}$$

$$\xmapsto{H\otimes I} \quad \begin{cases} (\pm 1)\,|0\rangle \otimes |-\rangle & \text{if } f(0) = f(1) \\ (\pm 1)\,|1\rangle \otimes |-\rangle & \text{if } f(0) \neq f(1) \end{cases} \tag{3.15}$$

Ignoring the global phase, the final state of the system is:

$$\xmapsto{meas\otimes I} \quad \begin{cases} |0\rangle \otimes |-\rangle & \text{if } f(0) = f(1) \\ |1\rangle \otimes |-\rangle & \text{if } f(0) \neq f(1) \end{cases} \tag{3.16}$$

Now, regarding the interpretation of the lambda term Deutch,

$$\begin{aligned}
&\llbracket \mathsf{Deutch} \rrbracket \\
={}& \llbracket - \triangleright \ \mathsf{pm}\ U_f(H\,(\mathit{new}\,0\,(*)) \otimes H\,(\mathit{new}\,1\,(*)))\ \mathit{to}\ q_1 \otimes q_2\,. \\
& \mathit{meas}(H(q_1)) \otimes q_2 : \mathit{qbit} \otimes \mathit{qbit} \rrbracket \\
={}& \llbracket q_1 : \mathit{qbit} \otimes q_2 : \mathit{qbit} \triangleright \mathit{meas}(H(q_1)) \otimes q_2 : \mathit{qbit} \otimes \mathit{qbit} \rrbracket \cdot \mathsf{jn}_{-;\mathit{qbit};\mathit{qbit}} && \{\llbracket \otimes_e \rrbracket\} \\
& \cdot \alpha \cdot \mathsf{sw} \cdot (\llbracket - \triangleright U_f(H\,(\mathit{new}\,0\,(*)) \otimes H\,(\mathit{new}\,1\,(*))) : \mathit{qbit} \otimes \mathit{qbit} \rrbracket \\
& \otimes I_{\mathbb{C}}) \cdot \mathsf{sp}_{-;-} \cdot \mathsf{sh}_{-;-} \\
={}& (\llbracket q_1 : \mathit{qbit} \triangleright \mathsf{meas}(H(q_1)) : \mathit{qbit} \rrbracket \otimes \llbracket q_2 : \mathit{qbit} \triangleright q_2 : \mathit{qbit} \rrbracket) && \{\llbracket \otimes_i \rrbracket, \\
& \cdot \mathsf{sp}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{sh}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{jn}_{-;\mathit{qbit};\mathit{qbit}} \cdot \alpha \cdot \mathsf{sw} \cdot (\llbracket - \triangleright U_f(H\,(\mathit{new}\,0\,(*)) && \text{Def. sp and sh}\} \\
& \otimes H\,(\mathit{new}\,1\,(*))) : \mathit{qbit} \otimes \mathit{qbit} \rrbracket \otimes I_{\mathbb{C}}) \cdot I_{\mathbb{C}} \otimes I_{\mathbb{C}} \cdot I_{\mathbb{C}} \otimes I_{\mathbb{C}} \\
={}& (\llbracket \mathsf{meas} \rrbracket \cdot \llbracket H \rrbracket \cdot \llbracket q_1 : \mathit{qbit} \triangleright q_1 : \mathit{qbit} \rrbracket \cdot \mathsf{sp}_{\mathit{qbit}} \cdot \mathsf{sh}_{\mathit{qbit}} \ \otimes I_{\llbracket \mathit{qbit} \rrbracket}) && \{\llbracket \mathsf{ax} \rrbracket, \llbracket \mathsf{hyp} \rrbracket\} \\
& \cdot \mathsf{sp}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{sh}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{jn}_{-;\mathit{qbit};\mathit{qbit}} \cdot \alpha \cdot \mathsf{sw} \cdot (\llbracket U_f \rrbracket \cdot (\llbracket H \rrbracket \otimes \llbracket H \rrbracket \\
& \cdot \llbracket \mathit{new}\,0 \rrbracket \cdot \llbracket - \triangleright * : \mathbb{I} \rrbracket \cdot \llbracket \mathit{new}\,1 \rrbracket \cdot \llbracket - \triangleright * : \mathbb{I} \rrbracket) \cdot \mathsf{sp}_{-;-} \cdot \mathsf{sh}_{-} \otimes I_{\mathbb{C}}) \\
={}& (\llbracket \mathsf{meas} \rrbracket \cdot \llbracket H \rrbracket \cdot I_{\llbracket \mathit{qbit} \rrbracket} \cdot I_{\llbracket \mathit{qbit} \rrbracket} \otimes I_{\llbracket \mathit{qbit} \rrbracket}) \cdot \mathsf{sp}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{sh}_{\mathit{qbit};\mathit{qbit}} && \{\llbracket \mathsf{hyp} \rrbracket, \llbracket \mathbb{I}_i \rrbracket, \\
& \cdot \mathsf{jn}_{-;\mathit{qbit};\mathit{qbit}} \cdot \alpha \cdot \mathsf{sw} \cdot ((\llbracket U_f \rrbracket \cdot (\llbracket H \rrbracket \cdot \llbracket \mathit{new}\,0 \rrbracket \cdot I_{\mathbb{C}} \otimes \llbracket H \rrbracket \cdot \llbracket \mathit{new}\,1 \rrbracket && \text{Def. sp and sh}\} \\
& \cdot I_{\mathbb{C}}) \cdot I_{\mathbb{C}} \otimes I_{\mathbb{C}} \cdot I_{\mathbb{C}} \otimes I_{\mathbb{C}}) \otimes I_{\mathbb{C}}) \\
={}& (\llbracket \mathsf{meas} \rrbracket \cdot \llbracket H \rrbracket \otimes I_{\llbracket \mathit{qbit} \rrbracket}) \cdot \mathsf{sp}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{sh}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{jn}_{-;\mathit{qbit};\mathit{qbit}} \cdot \alpha \cdot \mathsf{sw} && \{\text{Figure 6}\} \\
& \cdot (U_f\,(H\,|0\rangle\,\langle 0|\,H^{\dagger} \otimes H\,|1\rangle\,\langle 1|\,H^{\dagger})U_f^{\dagger} \otimes I_{\mathbb{C}}) \\
={}& (\llbracket \mathsf{meas} \rrbracket \cdot \llbracket H \rrbracket \otimes I_{\llbracket \mathit{qbit} \rrbracket}) \cdot \mathsf{sp}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{sh}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{jn}_{-;\mathit{qbit};\mathit{qbit}} \cdot \alpha \cdot \mathsf{sw} && \{\text{Equation 3.7}, \\
& \cdot \Bigg(\left(\frac{1}{\sqrt{2}}((-1)^{f(0)}\,|0\rangle\,\langle 0| \otimes |-\rangle\,\langle -| + (-1)^{f(1)}\,|1\rangle\,\langle 1| \otimes |-\rangle\,\langle -|) \right) && \text{Equation 3.13}\} \\
& \otimes I_{\mathbb{C}} \Bigg) \\
={}& (\llbracket \mathsf{meas} \rrbracket \cdot \llbracket H \rrbracket \otimes I_{\llbracket \mathit{qbit} \rrbracket}) \cdot \mathsf{sp}_{\mathit{qbit};\mathit{qbit}} \cdot \mathsf{sh}_{\mathit{qbit};\mathit{qbit}} \cdot \lambda \otimes I_{\mathbb{C}} \cdot \alpha && \{\text{Def. jn}\} \\
& \cdot \Bigg(I_{\mathbb{C}} \otimes \Bigg(\frac{1}{\sqrt{2}}((-1)^{f(0)}\,|0\rangle\,\langle 0| \otimes |-\rangle\,\langle -| + (-1)^{f(1)}\,|1\rangle\,\langle 1| \\
& \otimes |-\rangle\,\langle -|) \Bigg) \Bigg) \\
={}& (\llbracket \mathsf{meas} \rrbracket \cdot \llbracket H \rrbracket \otimes I_{\llbracket \mathit{qbit} \rrbracket}) \cdot \Bigg(\frac{1}{\sqrt{2}}((-1)^{f(0)}\,|0\rangle\,\langle 0| \otimes |-\rangle\,\langle -| && \{\text{Def. sp and sh}\} \\
& + (-1)^{f(1)}\,|1\rangle\,\langle 1| \otimes |-\rangle\,\langle -|) \Bigg) \Bigg)
\end{aligned}$$

$$= \begin{cases} (M_0 \otimes I_{[\![qbit]\!]}) \cdot (H\,|+\rangle\langle +|\,H^\dagger \otimes |-\rangle\langle -|) & \text{if } f(0) = f(1) \\ \cdot (M_0^\dagger \otimes I_{[\![qbit]\!]}) + (M_1 \otimes I_{[\![qbit]\!]}) & \\ \cdot (H\,|+\rangle\langle +|\,H^\dagger \otimes |-\rangle\langle -|)(M_1^\dagger \otimes I_{[\![qbit]\!]}) & \\ (M_0 \otimes I_{[\![qbit]\!]}) \cdot (H\,|-\rangle\langle -|\,H^\dagger \otimes |-\rangle\langle -|) & \text{if } f(0) \neq f(1) \\ \cdot (M_0^\dagger \otimes I_{[\![qbit]\!]}) + (M_1 \otimes I_{[\![qbit]\!]}) & \\ \cdot (H\,|-\rangle\langle -|\,H^\dagger \otimes |-\rangle\langle -|)(M_1^\dagger \otimes I_{[\![qbit]\!]}) & \end{cases} \qquad \{\text{Equation 3.14, Figure 6}\}$$

$$= \begin{cases} |0\rangle\langle 0| \otimes |-\rangle\langle -| & \text{if } f(0) = f(1) \\ |1\rangle\langle 1| \otimes |-\rangle\langle -| & \text{if } f(0) \neq f(1) \end{cases} \qquad \{\text{Equation 3.15, Equation 3.16}\}$$

Deutsch's Algorithm with Measurement Errors

A measurement error is characterized by reading a "1" as a "0" or vice versa. Measurement errors do not impact all states uniformly [55]. Consequently, there is a discrepancy in how frequently the state "1" is incorrectly read as "0" compared to how often the state "0" is measured as "1" or vice versa.

Given probabilities p_1 and p_2 of measuring a "0" as a "1" and a "1" as a "0", respectively, a measurement featuring this type of error, denoted meas$^\epsilon$, is defined as follows:

$$\begin{aligned} \text{meas}^\epsilon :& [\![qbit]\!] \to [\![qbit]\!] \\ & \rho \mapsto (1-p_1)M_0\rho M_0^\dagger + (1-p_2)M_1\rho M_1^\dagger + p_1 M_1 X\rho X^\dagger M_1^\dagger + p_2 M_0 X \rho X^\dagger M_0^\dagger \end{aligned} \tag{3.17}$$

Considering an arbitrary state $\rho = |\alpha|^2|0\rangle\langle 0| + \alpha\beta^*|0\rangle\langle 1| + \alpha^*\beta|1\rangle\langle 0| + |\beta|^2|1\rangle\langle 1|$, the resulting state after measurement is:

$$\begin{aligned} \text{meas}^\epsilon(\rho) &= (1-p_1)|\alpha|^2|0\rangle\langle 0| + (1-p_2)|\beta|^2|1\rangle\langle 1| + p_1|\alpha|^2|1\rangle\langle 1| + p_2|\beta|^2|0\rangle\langle 0| \\ &= ((1-p_1)|\alpha|^2 + p_2|\beta|^2)|0\rangle\langle 0| + ((1-p_2)|\beta|^2 + p_1|\alpha|^2)|1\rangle\langle 1| \end{aligned} \tag{3.18}$$

Given that

$$\begin{aligned} \text{Tr}(\text{meas}^\epsilon(\rho)) =& \text{Tr}(((1-p_1)|\alpha|^2 + p_2|\beta|^2)|0\rangle\langle 0| + ((1-p_2)|\beta|^2 + p_1|\alpha|^2)|1\rangle\langle 1|) \\ =&(1-p_1)|\alpha|^2 + p_2|\beta|^2 + (1-p_2)|\beta|^2 + p_1|\alpha|^2 = |\alpha|^2 + |\beta|^2 = \text{Tr}(\rho) \end{aligned} \tag{3.19}$$

this operator is trace-preserving. Attending to Equation 3.3 and to the fact that the sum of positive semidefinite matrices is semidefinite (Definition 2.1.21), it is straightforward to prove that the operator meas$^\epsilon$ is completely positive. As a result, meas$^\epsilon$ is a quantum channel and, consequently, a valid operation in the model considered.

For example, considering $p_1 = 0.1$ and $p_2 = 0.3$ the resulting state after measurement is:

$$\rho' = \begin{cases} 0.9 \left|0\right\rangle \left\langle 0\right| \left|-\right\rangle \left\langle -\right| + 0.1 \left|1\right\rangle \left\langle 1\right| \left|-\right\rangle \left\langle -\right| & \text{if } f(0) = f(1) \\ 0.3 \left|0\right\rangle \left\langle 0\right| \left|-\right\rangle \left\langle -\right| + 0.7 \left|1\right\rangle \left\langle 1\right| \left|-\right\rangle \left\langle -\right| & \text{if } f(0) \neq f(1) \end{cases} \tag{3.20}$$

As a result, the discrepancy between the ideal and actual measurement results,$[\![\rho - \rho']\!]_\diamond$, corresponds to:

$$\begin{aligned} & [\![\rho - \rho']\!]_\diamond \qquad (3.21) \\ = & \begin{cases} \|(\left|0\right\rangle \left\langle 0\right| \left|-\right\rangle \left\langle -\right|) - (0.9 \left|0\right\rangle \left\langle 0\right| \left|-\right\rangle \left\langle -\right| + 0.1 \left|1\right\rangle \left\langle 1\right| \left|-\right\rangle \left\langle -\right|)\|_\diamond & \text{if } f(0) = f(1) \\ \|(\left|1\right\rangle \left\langle 1\right| \left|-\right\rangle \left\langle -\right|) - (0.3 \left|0\right\rangle \left\langle 0\right| \left|-\right\rangle \left\langle -\right| + 0.7 \left|1\right\rangle \left\langle 1\right| \left|-\right\rangle \left\langle -\right|)\|_\diamond & \text{if } f(0) \neq f(1) \end{cases} \\ = & \begin{cases} \|0.1 \left|0\right\rangle \left\langle 0\right| \left|-\right\rangle \left\langle -\right| - 0.1 \left|1\right\rangle \left\langle 1\right| \left|-\right\rangle \left\langle -\right|\|_\diamond & \text{if } f(0) = f(1) \\ \|-0.3 \left|0\right\rangle \left\langle 0\right| \left|-\right\rangle \left\langle -\right| + 0.3 \left|1\right\rangle \left\langle 1\right| \left|-\right\rangle \left\langle -\right|\|_\diamond & \text{if } f(0) \neq f(1) \end{cases} \\ = & \begin{cases} \|0.1 (\left|0\right\rangle \left\langle 0\right| - \left|1\right\rangle \left\langle 1\right|) \left|-\right\rangle \left\langle -\right|\|_\diamond & \text{if } f(0) = f(1) \\ \|0.3 (\left|1\right\rangle \left\langle 1\right| - \left|0\right\rangle \left\langle 0\right|) \left|-\right\rangle \left\langle -\right|\|_\diamond & \text{if } f(0) \neq f(1) \end{cases} \end{aligned}$$

Attending to Definition 2.1.9 and the multiplicativity of the diamond norm with respect to tensor products, it follows that:

$$[\![\rho - \rho']\!]_\diamond = \begin{cases} 0.1 \|\left|0\right\rangle \left\langle 0\right| - \left|1\right\rangle \left\langle 1\right|\|_\diamond \|\left|-\right\rangle \left\langle -\right|\|_\diamond & \text{if } f(0) = f(1) \\ 0.3 \|(\left|1\right\rangle \left\langle 1\right| - \left|0\right\rangle \left\langle 0\right|)\|_\diamond \|\left|-\right\rangle \left\langle -\right|\|_\diamond & \text{if } f(0) \neq f(1) \end{cases} \tag{3.22}$$

Employing Equation 2.9, it is easily concluded that the Bloch vectors of the states $\left|0\right\rangle \left\langle 0\right|$, $\left|1\right\rangle \left\langle 1\right|$, and $\left|-\right\rangle \left\langle -\right|$ are $(-1, 0, 0)$, $(0, 0, 1)$, and $(0, 0, -1)$, respectively. Consequently, and considering Definition 2.1.9 the discrepancy between the ideal and actual measurement results is:

$$
\begin{aligned}
[\![\rho - \rho']\!]_\diamond &= \begin{cases} 0.1\|(0,0,1)\|_2\|(1,0,0)\|_2 & \text{if } f(0) = f(1) \\ 0.3\|(0,0,-2)\|_2\|(1,0,0)\|_2 & \text{if } f(0) \neq f(1) \end{cases} \\
&= \begin{cases} 0.2\sqrt{1^2+0^2+0^2} \cdot \sqrt{1^2+0^2+0^2} & \text{if } f(0) = f(1) \\ 0.6\sqrt{(-1)^2+0^2+0^2} + \sqrt{1^2+0^2+0^2} & \text{if } f(0) \neq f(1) \end{cases} \\
&= \begin{cases} 0.2 & \text{if } f(0) = f(1) \\ 0.6 & \text{if } f(0) \neq f(1) \end{cases}
\end{aligned} \tag{3.23}
$$

For an arbitray distance ϵ, between the ideal post-measurement state ρ he erroneous post-measurement stat ρ', using the metric deductive system in Figure 4 one can write:

$$
\begin{aligned}
&- \triangleright \text{ pm } U_f(H\,(\textit{new}\,0\,(*)) \otimes H\,(\textit{new}\,1\,(*)))\ \textit{to}\ q_1 \otimes q_2\,. \\
&\quad \textit{meas}(H(q_1)) \otimes q_2 : \textit{qbit} \otimes \textit{qbit} \\
&=_\epsilon \\
&- \triangleright \text{ pm } U_f(H\,(\textit{new}\,0\,(*)) \otimes H\,(\textit{new}\,1\,(*)))\ \textit{to}\ q_1 \otimes q_2\,. \\
&\quad \textit{meas}^\epsilon(H(q_1)) \otimes q_2 : \textit{qbit} \otimes \textit{qbit}
\end{aligned}
$$

Therefore, Deutsch $=_\epsilon$ Deutsch$^{\textit{meas}^\epsilon}$, and consequently, for scenario under consideration, if f is a constant function, Deutsch $=_{0.2}$ Deutsch$^{\textit{meas}^\epsilon}$; otherwise, Deutsch $=_{0.6}$ Deutsch$^{\textit{meas}^\epsilon}$.

3.4.3 Example: Proving an equivalence using equations-in-context

This subsection aims to illustrate how to prove that creating a new qubit, discarding it, and then creating a new qubit is equivalent to just creating a new qubit, *i.e.*,

$$
- \triangleright \text{disc } \textit{new}\,0\,(*)\,\text{to} * .\ \textit{new}\,0\,(*) : \textit{qbit} = - \triangleright \textit{new}\,0\,(*) : \textit{qbit} \tag{3.24}
$$

syntactically, using equations-in-context.

The discard equation in the bottom line in Figure 3 states that all judgements $\Gamma \triangleright v : \mathbb{I}$ (with $\Gamma = x_1 : \mathbb{A}_1, ..., x_n : \mathbb{A}_n$) carry no different information than that of just discarding all variables available in context Γ. Therefore,

$$
- \triangleright * = \text{disc } \textit{new}\,0\,(*)\,\text{to} * : \mathbb{I} \tag{3.25}
$$

Consequently,

$$- \rhd \text{disc } \textit{new}0(*) \text{to} * . \textit{new}0(*) : \textit{qbit} = - \rhd * \text{to} * . \textit{new}0(*) : \textit{qbit} \tag{3.26}$$

Subsequently applying the rule $\beta_{\mathbb{I}_e}$ in Figure 3, it holds that

$$\begin{aligned} - \rhd \text{disc } \textit{new}0(*) \text{to} * . \textit{new}0(*) : \textit{qbit} &= - \rhd * \text{ to } * . \textit{new}0(*) : \textit{qbit} \\ &= - \rhd \textit{new}0(*) : \textit{qbit} \end{aligned} \tag{3.27}$$

3.4.4 Linear λ-calculus meets probabilistic programming

Arranjar exemplo -> devo pegar no do paper e explanar síntaxe e assim

Falar que aqui vamos usar a categoria do paper do professor Renato e q lá se prova que a cat e o functor $\otimes$ e a adjunção são met enriched.

Chapter 4

Conditionals

Adicionar as coisas das notas dq elas estiverem corrigidas + doc booleanos + intro às coisas

Na parte quantica mencionar o artigo do selinger de 2009, questão da linearidade

Por isto no sitio certo -> modelo quantico aqui mudamos a op de meas e de conversão em bit em qb

Within the model of quantum lambda calculus, introduced in section 3.4, the measurement operation is defined in accordance with the standard approach in physics [12]. However, within this definiton the distinction between classical and quantum states is made through the elements of the spaces and not the spaces themselves, in the sense that both the result of measuring a quantum state, which is a classical bit, and the quantum state itself are elements of the same space, $\mathbb{C}^{2\times 2}$.

4.1 Syntax

4.2 Interpretation

4.3 Metric Equations

4.4 Examples

Chapter 5

Conclusions and future work

Conclusions and future work.

5.1 Conclusions

5.2 Prospect for future work

Bibliography

[1] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7), 1982.

[2] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[3] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[4] A Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996.

[5] Daniel Gottesman. *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[6] Andrew M Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77(5):793, 1996.

[7] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.

[8] Joel J Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94(5):052325, 2016.

[9] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, pages 1015–1029, 2019.

[10] Aram W Harrow, Benjamin Recht, and Isaac L Chuang. Efficient discrete approximations of quantum gates. *Journal of Mathematical Physics*, 43(9):4445–4451, 2002.

[11] Lukas Burgholzer and Robert Wille. Advanced equivalence checking for quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(9):1810–1824, 2020.

[12] John Watrous. *The theory of quantum information*. Cambridge university press, 2018.

[13] Glynn Winskel. *The formal semantics of programming languages: an introduction*. MIT press, 1993.

[14] Jianjun Zhao. Quantum software engineering: Landscapes and horizons. *arXiv preprint arXiv:2007.07047*, 2020.

[15] Manuel A Serrano, Jose A Cruz-Lemus, Ricardo Perez-Castillo, and Mario Piattini. Quantum software components and platforms: Overview and quality assessment. *ACM Computing Surveys*, 55(8):1–31, 2022.

[16] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.

[17] Benjamin Bichsel, Maximilian Baader, Timon Gehr, and Martin Vechev. Silq: A high-level quantum language with safe uncomputation and intuitive semantics. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 286–300, 2020.

[18] Alexander S Green, Peter LeFanu Lumsdaine, Neil J Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, pages 333–342, 2013.

[19] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q# enabling scalable quantum computing and development with a high-level dsl. In *Proceedings of the real world domain specific languages workshop 2018*, pages 1–10, 2018.

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

[21] Frederic T Chong, Diana Franklin, and Margaret Martonosi. Programming languages and compiler design for realistic quantum hardware. *Nature*, 549(7671):180–187, 2017.

[22] Shih-Han Hung, Kesha Hietala, Shaopeng Zhu, Mingsheng Ying, Michael Hicks, and Xiaodi Wu. Quantitative robustness analysis of quantum programs. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019.

[23] Runzhou Tao, Yunong Shi, Jianan Yao, John Hui, Frederic T Chong, and Ronghui Gu. Gleipnir: toward practical error analysis for quantum programs. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 48–64, 2021.

[24] Fredrik Dahlqvist and Renato Neves. The syntactic side of autonomous categories enriched over generalised metric spaces. *Logical Methods in Computer Science*, 19, 2023.

[25] Jean-Yves Girard, Yves Lafont, and Laurent Regnier. *Advances in linear logic*, volume 222. Cambridge University Press, 1995.

[26] P Nick Benton. A mixed linear and non-linear logic: Proofs, terms and models. In *International Workshop on Computer Science Logic*, pages 121–135. Springer, 1994.

[27] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[28] Walter Rudin. *Functional Analysis*. McGraw-Hill, 1991.

[29] A Hitchhiker's Guide. *Infinite dimensional analysis*. Springer, 2006.

[30] Simon Perdrix. Quantum entanglement analysis based on abstract interpretation. In *International Static Analysis Symposium*, pages 270–282. Springer, 2008.

[31] Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical review*, 47(10):777, 1935.

[32] John S Bell. On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1(3):195, 1964.

[33] Alain Aspect, Jean Dalibard, and Gérard Roger. Experimental test of bell's inequalities using time-varying analyzers. *Physical review letters*, 49(25):1804, 1982.

[34] Wolfgang Tittel, Jürgen Brendel, Bernard Gisin, Thomas Herzog, Hugo Zbinden, and Nicolas Gisin. Experimental demonstration of quantum correlations over more than 10 km. *Physical Review A*, 57(5):3229, 1998.

[35] Jian-Wei Pan, Dik Bouwmeester, Harald Weinfurter, and Anton Zeilinger. Experimental entanglement swapping: entangling photons that never interacted. *Physical review letters*, 80(18):3891, 1998.

[36] Juan Yin, Yuan Cao, Yu-Huai Li, Sheng-Kai Liao, Liang Zhang, Ji-Gang Ren, Wen-Qi Cai, Wei-Yue Liu, Bo Li, Hui Dai, et al. Satellite-based entanglement distribution over 1200 kilometers. *Science*, 356(6343):1140–1144, 2017.

[37] Daniel A Lidar. Lecture notes on the theory of open quantum systems. *arXiv preprint arXiv:1902.00967*, 2019.

[38] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[39] Fredrik Dahlqvist, Alexandra Silva, and Dexter Kozen. Semantics of Probabilistic Programming: A Gentle Introduction. In Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva, editors, *Foundations of Probabilistic Programming*, pages 1–42. Cambridge University Press, 2020.

[40] Krishna B. Athreya and Soumendra N. Lahiri. *Measure Theory and Probability Theory*. Springer Science & Business Media, 2006.

[41] Charalambos D. Aliprantis and Kim C. Border. In Charalambos D. Aliprantis and Kim C. Border, editors, *Infinite Dimensional Analysis: A Hitchhiker's Guide*, pages 301–330. Springer, 1999.

[42] Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva, editors. *Foundations of Probabilistic Programming*. Cambridge University Press, 2020.

[43] Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[44] Samuel Eilenberg and Saunders MacLane. General Theory of Natural Equivalences. *Transactions of the American Mathematical Society*, 58(2):231–294.

[45] Noson S. Yanofsky. *Monoidal Category Theory: Unifying Concepts in Mathematics, Physics, and Computing*. MIT Press, 2024.

[46] Hendrik P Barendregt et al. *The lambda calculus*, volume 3. North-Holland Amsterdam, 1984.

[47] Paul Bernays. Alonzo church. an unsolvable problem of elementary number theory. american journal of mathematics, vol. 58 (1936), pp. 345–363. *The Journal of Symbolic Logic*, 1(2):73–74, 1936.

[48] John W Backus, Friedrich L Bauer, Julien Green, Charles Katz, John McCarthy, Alan J Perlis, Heinz Rutishauser, Klaus Samelson, Bernard Vauquois, Joseph Henry Wegstein, et al. Report on the algorithmic language algol 60. *Communications of the ACM*, 3(5):299–311, 1960.

[49] Michael Shulman. A practical type theory for symmetric monoidal categories. *Theory and Applications of Categories*, 37(5):863–907, 2021.

[50] Radu Mardare, Prakash Panangaden, and Gordon Plotkin. Quantitative algebraic reasoning. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 700–709, 2016.

[51] Radu Mardare, Prakash Panangaden, and Gordon Plotkin. On the axiomatizability of quantitative algebras. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017.

[52] Peter Selinger and Benoît Valiron. A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science*, 16(3):527–552, 2006.

[53] Peter Selinger, Benoıt Valiron, et al. Quantum lambda calculus. *Semantic techniques in quantum computation*, pages 135–172, 2009.

[54] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[55] Swamit S Tannu and Moinuddin K Qureshi. Mitigating measurement errors in quantum computers by exploiting state-dependent bias. In *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*, pages 279–290, 2019.

Place here information about funding, FCT project, etc. in which the work is framed. Leave empty otherwise.