Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Titulo de la tesis en español

Tesis presentada para optar al título de Doctor de la Universidad de Buenos Aires en el área de Ciencias de la Computación

Nombre y Apellido de quien escribe esta tesis

Directora/s de tesis:	Dra. Nombre y Apellido
Directora Asistente:	Dra. Nombre y Apellido
Consejera de estudios:	Dra. Nombre y Apellido
Lugar de trabajo:	Departamento de Computación Facultad de Cs. Exactas y Naturales Universidad de Buenos Aires

Buenos Aires, 2018
Fecha de defensa: 18 de Abril de 2018

Titulo de la tesis en español

Resumen:

Palabras clave: palabra1, palabra2, palabra3, palabra4, palabra5

Título de la tesis en inglés

Abstract:

Keywords: keyword1,keyword2,keyword3,keyword4

Agradecimientos

Contents

Chapter 1

A note on confluence in typed probabilistic lambda calculi

1.1 Introduction

When dealing with probabilistic lambda calculus, we can find two different sources of divergence.

- A *single redex* may reduce in two different ways via a probabilistic reduction.
- A term with *multiple redexes* and no strategy, could be reduced in different ways.

For example, we can consider a lambda calculus extended with a coin $\bigcirc$ reducing to 0 or 1 with probability $\frac{1}{2}$ each. Then, taking just the coin, we are in the first case of divergence. While taking, for example, $(\lambda x.\lambda y.yxx)\bigcirc$, we are in the second case, since we can either beta reduce, or reduce the coin.

There is no point in trying to achieve confluence in the first case: the coin is non-confluent by design. However, we can analyse the branching paths and verify that the probability of reducing to a particular term stays the same, regardless of the reduction sequence. This is what we call *probabilistic confluence*.

To study this kind of cases, Bournez and Kirchner developed the notion of PARS [**?**], later refined in [**?**]. Using the techniques described in [**?**, **?**] we can define the rewriting rules over the distributions.

If we denote by $[(p_1, t_1), \dots, (p_n, t_n)]$ the probability distribution where t_i has probability p_i, the possible reductions from the previous example are depicted in Figure 1.1. The resulting distributions are not only different, but also divergent, since in the left branch, the probability to arrive, for example, to $\lambda y.y01$ is 0, while it is one of the possible results in the right branch.

In this chapter we consider a simply typed lambda calculus extended with a coin, and show different possibilities for achieving some sort of confluence, without giving preference to any of them. This analysis will be helpful to set up the design choices of later chapters, since quantum measurement is an inherent probabilistic operation.

In Section 3.2 we introduce the calculus to be studied, without any restrictions either in the rewriting rules, nor in the typing rules. As we argued above, this naive definition is not confluent (cf. Figure 1.1), unless a strategy is defined (in which case it becomes trivially probabilistically confluent, as will be discussed in Section 1.3.1). In Section 1.3.2 we show that we can achieve confluence by internalizing the probabilistic reductions in the terms. In Section 1.3.3 we show that we can achieve probabilistic confluence by taking an affine-linear type system. Then, in Section 1.4, we show that we can relax the type system in an if-then-else branching, obtaining a probabilistic confluence result modulo a computational equivalence.

$$\begin{tikzcd}
 & (\lambda x.\lambda y.yxx)\bigcirc \arrow[dl] \arrow[dr] & \\
\begin{bmatrix} (\frac{1}{2}, (\lambda x.\lambda y.yxx)0), \\ (\frac{1}{2}, (\lambda x.\lambda y.yxx)1) \end{bmatrix} \arrow[d] & & [(1, \lambda y.y\bigcirc\bigcirc)] \arrow[d, "*"] \\
\begin{bmatrix} (\frac{1}{2}, \lambda y.y00), \\ (\frac{1}{2}, \lambda y.y11) \end{bmatrix} & & \begin{bmatrix} (\frac{1}{4}, \lambda y.y00), \\ (\frac{1}{4}, \lambda y.y01), \\ (\frac{1}{4}, \lambda y.y10), \\ (\frac{1}{4}, \lambda y.y11) \end{bmatrix}
\end{tikzcd}$$

Figure 1.1: *Counterexample of confluence*

1.2 The $\lambda_\circ$ calculus

In this section we present $\lambda_\circ$ (read "lambda coin"), which is the simply typed lambda calculus extended with booleans (1 and 0), an if-then-else construction, and a coin $\bigcirc$. Terms are inductively defined by

$$t := x \mid \lambda x.t \mid tt \mid 1 \mid 0 \mid \textsf{if } t \textsf{ then } t \textsf{ else } t \mid \bigcirc$$

The rewrite system is given in Table 1.1. The rules $t \to_p r$ mean that t reduces with probability p in one step to r, where the sum of probabilities of reducing one redex is 1. In particular, every non-contextual rule has probability 1, since there is only one rule per redex, except for the coin, which reduces with probability $\frac{1}{2}$ to 0 and probability $\frac{1}{2}$ to 1. If $t \to_{p_1} r_1 \dots \to_{p_n} r_n$ we may write $t \to^*_{\prod_{i=1}^{n} p_i} r_n$.

The reduction rules are intentionally as permissive as possible (even the branches of an if-then-else can be reduced) in order to analyse its (lack of) confluence. The type system is given in Table 1.2.

Strong normalization for this calculus follows trivially from Tait's proof for the simply typed λ-calculus, extended with booleans. The only reduction added is the probabilistic coin toss and it takes at most one step for each operator. Hence, using the rewriting over probabilistic distributions techniques from [**?**], we only need to show local confluence in order to achieve global confluence altogether. This is an adaptation of Newman's lemma for probabilistic calculi (see [**?**] for a longer discussion about probabilistic confluence).

Clearly, $\lambda_\circ$ is not confluent, as already seen in the introduction (see Figure 1.1). It is easy to see that these distributions represent different results.

The divergence stems from three characteristics of the calculus:

1. Lack of a reduction strategy.
2. Probabilistic reductions.
3. Duplication of variables.

Removing just one of these elements renders the system confluent, however each modification comes with its own trade-off. We will examine each case, one by one, in the following section.

$$(\lambda x.t)r \to_1 t[r/x] \qquad \mathsf{if}\ 1\ \mathsf{then}\ t\ \mathsf{else}\ u \to_1 t \qquad \bigcirc \to_{\frac{1}{2}} 1$$

$$\mathsf{if}\ 0\ \mathsf{then}\ t\ \mathsf{else}\ u \to_1 u \qquad \bigcirc \to_{\frac{1}{2}} 0$$

$$\frac{t \to_p r}{\lambda x.t \to_p \lambda x.r} \qquad \frac{t \to_p r}{ts \to_p rs} \qquad \frac{t \to_p r}{st \to_p sr}$$

$$\frac{t \to_p r}{\mathsf{if}\ t\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \to_p \mathsf{if}\ r\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2}$$

$$\frac{s_1 \to_p r_1}{\mathsf{if}\ t\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \to_p \mathsf{if}\ t\ \mathsf{then}\ r_1\ \mathsf{else}\ s_2}$$

$$\frac{s_2 \to_p r_2}{\mathsf{if}\ t\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2 \to_p \mathsf{if}\ t\ \mathsf{then}\ r_1\ \mathsf{else}\ r_2}$$

Table 1.1: *Rewrite system for* $\lambda_\circ$.

$$A := \mathbb{B} \mid A \to A$$

$$\frac{}{\Gamma, x : A \vdash x : A}\ \mathsf{ax} \qquad \frac{}{\Gamma \vdash 1 : \mathbb{B}}\ \mathsf{ax}_1 \qquad \frac{}{\Gamma \vdash 0 : \mathbb{B}}\ \mathsf{ax}_0$$

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

$$\frac{}{\Gamma \vdash \bigcirc : \mathbb{B}}\ \mathsf{ax}_{\bigcirc} \qquad \frac{\Gamma \vdash t : \mathbb{B} \quad \Gamma \vdash u : A \quad \Gamma \vdash v : A}{\Gamma \vdash \mathsf{if}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v : A}\ \mathsf{if}$$

Table 1.2: *Type system for* $\lambda_\circ$.

1.3 Removing the divergence sources

1.3.1 Defining a strategy

The definition of a strategy is the easiest modification. Choosing a reduction strategy makes all critical pairs disappear, since there is only one possible reduction rule to be applied for each term distribution. For example, in Figure 1.1 a *call-by-name* strategy would take the right path, where *call-by-value* would take the left one.

Ultimately the choice lies in how to interpret the duplication of variables. Reducing via call-by-name means that the probabilistic event is duplicated. Whereas, a call-by-value strategy duplicates the outcome of said event (see [**?**] for a discussion on this choice, and even an alternative combining both approaches).

$$A := \mathbb{B} \mid A \to A$$

$$\frac{}{\Gamma, x : A \vdash x : A}\ \mathsf{ax} \qquad \frac{}{\Gamma \vdash 1 : \mathbb{B}}\ \mathsf{ax}_1 \qquad \frac{}{\Gamma \vdash 0 : \mathbb{B}}\ \mathsf{ax}_0$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ \to_i \qquad \frac{\Gamma \vdash t : A \to B \quad \Delta \vdash r : A}{\Gamma, \Delta \vdash tr : B}\ \to_e$$

$$\frac{}{\Gamma \vdash \bigcirc : \mathbb{B}}\ \mathsf{ax}_{\bigcirc} \qquad \frac{\Gamma \vdash t : \mathbb{B} \quad \Delta_1 \vdash u : A \quad \Delta_2 \vdash v : A}{\Gamma, \Delta_1, \Delta_2 \vdash \mathsf{if}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v : A}\ \mathsf{if}$$

Table 1.3: *Affine type system for $\lambda_\circ$ (different contexts are considered to be disjoint).*

1.3.2 Internalising the probabilities

Following [**?**], we can modify the reduction on $\lambda_\circ$ to internalize the entire distribution of a term. In this particular case, every reduction has probability 1, and the coin toss deterministically reduces to its probability distribution. We write $t \oplus_p r$ for the probability distribution $[(p, t), (1 - p, r)]$, $(t \oplus_p r) \oplus_q s$ for the probability distribution $[(qp, t), (q(1 - p)r), (1 - q, s)]$, etc. Then, we can consider the rewrite rule $\bigcirc \to 0 \oplus_{\frac{1}{2}} 1$. This idea is common in non-probabilistic settings as well, e.g., [**?**].

Following this approach brings confluence to the calculus, since every repetition of a probabilistic event rewrites to the same result, its distribution. For example, the two branches of Figure 1.1 become $(\lambda x.\lambda y.yxx)(0 \oplus_{\frac{1}{2}} 1)$ and $\lambda y.y\bigcirc\bigcirc$, both converging to $\lambda y.y(0 \oplus_{\frac{1}{2}} 1)(0 \oplus_{\frac{1}{2}} 1)$.

Although this is a valid solution, it forces us to consider every possible state of a program at the same time along with its probability of occurrence, making the management of the system more complex. Here we are dealing with a simple coin, but more involved calculi might have several different reductions, each with its own distribution. If not designed correctly, a language that holds every possible state in the probability distribution can easily become too cumbersome to be effective.

1.3.3 Affine variables

The last reasonable solution is to restrict duplication. One way to do this is by controlling the appearance of variables at the type system level, with an affine type system, see Table 1.3.

This type system solves the counterexample from Figure 1.1, since the considered term has no type in this system. In particular, we can prove the following property:

Lemma 1.3.1. *If $r \to_p s$ then $t[r/x] \to_p t[s/x]$.*

Notice that this property is not true in the unrestricted $\lambda_\circ$. For example, while $\bigcirc \to_{\frac{1}{2}} 1$, we have $(\lambda y.yxx)[\bigcirc/x] = \lambda y.y\bigcirc\bigcirc \to^*_{\frac{1}{4}} \lambda y.y11$.

The drawback in this approach is clear, there is a loss in expressivity. Of course, in some cases, this restriction is a desirable quality. For example, in quantum computing it may serve to avoid cloning qubits, a forbidden operation in quantum mechanics.

1.4 Computational confluence with a sub-affine type system

The solution considered in Section 1.3.3 seems quite extreme. In particular, using the same variable in different branches of an if-then-else construction does not actually duplicate it, since only one of those branches will remain. However, changing the rule if from Table 1.3 to if_s given by

$$\frac{\Gamma \vdash t : \mathbb{B} \quad \Delta \vdash u : A \quad \Delta \vdash v : A}{\Gamma, \Delta \vdash \mathsf{if}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v : A}\ \mathsf{if}_s$$

breaks confluence anyway. We call this calculus "sub-affine". Consider the following example. Let $\mathsf{not} = \lambda x.\mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ 1$, then

$$\begin{tikzcd}
 & (\lambda x.\lambda y.\mathsf{if}\ y\ \mathsf{then}\ x\ \mathsf{else}\ \mathsf{not}\ x)\bigcirc \arrow[dl] \arrow[dr] & \\
\left[\begin{array}{c} (1/2, (\lambda x.\lambda y.\mathsf{if}\ y\ \mathsf{then}\ x\ \mathsf{else}\ \mathsf{not}\ x)0); \\ (1/2, (\lambda x.\lambda y.\mathsf{if}\ y\ \mathsf{then}\ x\ \mathsf{else}\ \mathsf{not}\ x)1) \end{array}\right] \arrow[d] & & [(1, (\lambda y.\mathsf{if}\ y\ \mathsf{then}\ \bigcirc\ \mathsf{else}\ \mathsf{not}\ \bigcirc))] \arrow[d] \\
\left[\begin{array}{l} (^1/_2, \lambda y.\mathsf{if}\ y\ \mathsf{then}\ 0\ \mathsf{else}\ 1); \\ (^1/_2, \lambda y.\mathsf{if}\ y\ \mathsf{then}\ 1\ \mathsf{else}\ 0) \end{array}\right] & & \left[\begin{array}{l} (^1/_4, \lambda y.\mathsf{if}\ y\ \mathsf{then}\ 0\ \mathsf{else}\ 0); \\ (^1/_4, \lambda y.\mathsf{if}\ y\ \mathsf{then}\ 0\ \mathsf{else}\ 1); \\ (^1/_4, \lambda y.\mathsf{if}\ y\ \mathsf{then}\ 1\ \mathsf{else}\ 0); \\ (^1/_4, \lambda y.\mathsf{if}\ y\ \mathsf{then}\ 1\ \mathsf{else}\ 1) \end{array}\right]
\end{tikzcd}$$

Note that terms in both distributions are in normal form. The two paths are syntactically divergent, however the resulting programs share the same behaviour under the same inputs. If we were to apply the resulting abstractions to 0 or to 1, both paths would yield $[(^1/_2, 0); (^1/_2, 1)]$. Therefore, these distributions are semantically confluent, they are not the same terms but they *represent* the same function.

We can formalise this notion for the sub-affine calculus as follows. Let $C := \Diamond \mid Cv$ be an elimination context, where $\Diamond$ is called "placeholder" and v is a normal closed term. We write $C\langle t\rangle$ for $C[t/\Diamond]$. Notice that $C\langle t\rangle$ is a term. We say that C is an elimination context of A, written C^A, if for all $\vdash t : A$, we have $\vdash C\langle t\rangle : \mathbb{B}$. That is, it applies t until it reaches the basic type $\mathbb{B}$.

The computational equivalence is defined as follows.

Definition 1.4.1. Let $D_1 = [(p_i, t_i)]_i$ and $D_2 = [(q_j, r_j)]_j$ be two distributions of terms, all closed of type A. Then, we say that these distributions are computationally equivalent (notation $D_1 \equiv D_2$) if for all C^A we have $C\langle t_i\rangle \to^*_{u_{ik}} b_{ik}$ and $C\langle r_j\rangle \to^*_{s_{jh}} c_{jh}$ such that the b_{ik} and c_{jh} are in normal form, and $[(p_i u_{ik}, b_{ik})]_{ik} \sim [(q_j s_{jh}, c_{jh})]_{jh}$, where $\sim$ denotes the equality on distributions.

The previous definition means that two distributions are computationally equivalent if by applying the resulting terms to all the possible inputs, they produce the same probability distribution of results. Notice that the definition is not assuming confluence.

Then, we can prove that the confluence modulo computational equivalence of the sub-affine calculus (Theorem 1.4.4). We need the following two lemmas, which follow by straightforward induction.

Lemma 1.4.2. *Let $y \notin \mathrm{FV}(r)$, then $t[q/y][r/x] = t[r/x][q[r/x]/y]$.* □

Lemma 1.4.3. *Let t reduce to the distribution D, then $t[r/x]$ reduces to $D[r/x]$.* □

Theorem 1.4.4 (Computational confluence)**.** *Let $\vdash t : A$ in the sub-affine calculus. If t reduces to the distribution D_1 and to the distribution D_2, then $D_1 \equiv D_2$.*

Proof. We only consider the six critical pairs, four derived from the if-then-else construction and two from the classical lambda calculus. Non-critical pairs are trivially probabilistically confluent.

$$\begin{tikzcd}
 & \mathsf{if}\ 1\ \mathsf{then}\ r\ \mathsf{else}\ s \arrow[dl] \arrow[dr] & \\
{[(p_i, \mathsf{if}\ 1\ \mathsf{then}\ r_i\ \mathsf{else}\ s)]_i} \arrow[dr, dashed] & & {[(1,r)]} \arrow[dl, dashed] \\
 & {[(p_i, r_i)]_i} &
\end{tikzcd}$$

$$\begin{tikzcd}
 & \mathsf{if}\ 1\ \mathsf{then}\ s\ \mathsf{else}\ r \arrow[dl] \arrow[dr] & \\
{[(p_i, \mathsf{if}\ 1\ \mathsf{then}\ s\ \mathsf{else}\ r_i)]_i} \arrow[dr, dashed] & & {[(1,s)]} \arrow[dl, dashed] \\
 & {[(p_i, s)]_i \sim [(1,s)]} &
\end{tikzcd}$$

The symmetrical cases with 0 close in a similar way. The fifth critical pair closes by Lemma 1.4.3.

$$\begin{tikzcd}
{[(p_i, (\lambda x.t_i)r)]_i} \arrow[dr, dashed] & (\lambda x.t)r \arrow[l] \arrow[r] & {[(1, t[r/x])]} \arrow[dl, dashed] \\
 & {[(p_i, t_i[r/x])]_i} &
\end{tikzcd}$$

The last critical pair requires a more thorough analysis.

$$\begin{tikzcd}
{[(p_i, (\lambda x.t)r_i)]_i} \arrow[dr, dashed] & (\lambda x.t)r \arrow[l] \arrow[r] & {[(1, t[r/x])]} \\
 & {[(p_i, t[r_i/x])]_i} &
\end{tikzcd}$$

We must prove that for all C^A and for all i, there exists b_{ij} with $C\langle t[r_i/x]\rangle \to^*_{q_{ij}} b_{ij}$ and c_k with $C\langle t[r/x]\rangle \to^*_{s_k} c_k$ such that $[p_i q_{ij}, b_{ij}]_{ij} \sim [s_k, c_k]_k$. It is enough to take only one path to $[p_i q_{ij}, b_{ij}]_{ij}$ and to $[s_k, c_k]_k$ according to the definition of $\equiv$. If $x \notin \mathrm{FV}(t)$, then $C\langle t[r/x]\rangle = C\langle t\rangle = C\langle t[r_i/x]\rangle$. If x appears once in t, $C\langle t[r/x]\rangle \to_{p_i} C\langle t[r_i/x]\rangle$. If x appears more than once in t, it appears at most 2^n times (where n is the number of if-then-else constructions), so we can proceed by induction on n.

- If $n = 1$, then there is one if-then-else, say $\mathsf{if}\ c\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$ and x appears both in s_1 and in s_2. Therefore, there are three cases:
  - c is 0 or 1, then, for $j = 0$ or $j = 1$, we have $C\langle t[r/x]\rangle \to^*_1 C\langle s_j[r/x]\rangle \to^*_{p_i} C\langle s_j[r_i/x]\rangle$. Notice that $C\langle t[r_i/x]\rangle \to_1 C\langle s_j[r_i/x]\rangle$.
  - $c \to^*_{q_j} j$ with $j = 0, 1$, so we have $C\langle t[r/x]\rangle \to^*_{q_j} C\langle s_j[r/x]\rangle \to^*_{p_i} C\langle s_j[r_i/x]\rangle$. That is, we arrived to the distribution $[(q_j p_i, s_j[r_i/x])]_{ij}$. Notice that since $C\langle t[r_i/x]\rangle \to_{q_j} C\langle s_j[r_i/x]\rangle$, we arrive to the same distribution in the right branch of this critical pair.
  - c is an open term, hence not reducing to a constant 0 or 1. In such a case, since $t[r/x]$ is a closed term, it means that the if-then-else construction in t is under a lambda-abstraction, therefore, we must beta-reduce first, either with the argument in t, or with an external argument given by the context C. We can repeat the same process until we get one of the previously treated cases (that is, at some point, the if-then-else becomes a redex).
- If $n > 1$, we proceed as the previous case, reducing the if-then-else first, which reduces n and so the induction hypothesis applies. □

1.5 Conclusion

In this paper we have analyzed the different possibilities to transform a simple probabilistic calculus into a (probabilistically / computationally) confluent calculus. The main contribution on this note is Theorem 1.4.4, which proves that we can relax the affinity restriction on "non-interfering" paths. This technique has been considered in the first author master's thesis [?] to prove the computational confluence of the quantum lambda calculus λ_ρ [?].

Chapter 2

Encoding High-level Quantum Programs as SZX-diagrams

2.1 Introduction

The ZX calculus [25] has been used as intermediary representation language for quantum programs in optimization methods [14, 3, 2] and in the design of error correcting schemes [9].

The highly flexible representation of linear maps as open graphs with a complete formal rewriting system and the multiple extensions adapted to represent different sets of quantum primitives have proven useful in reasoning about the properties of quantum circuits.

Quantum operations are usually represented as quantum circuits composed by primitive gates operating over a fixed number of qubits.The ZX calculus has a close correspondence to this model and is similarly limited to representing operations at a single-qubit level.

In this chapter we will focus on the Scalable ZX extension [5], which generalizes the ZX diagrams to work with arbitrary qubit registers using a compact representation. Previous work [4] has shown that the SZX calculus is capable of encoding nontrivial algorithms via the presentation of multiple hand-written examples. For an efficient usage as an intermediate representation language, we require an automated compilation method from quantum programming languages to SZX diagrams. While ZX diagrams can be directly obtained from a program compiled to a quantum circuit, our focus here is to leverage the parametricity of the SZX calculus.

There are several quantum programming languages capable of encoding high-level parametric programs [17, 11, 24]. Quipper [20] is a language for quantum computing capable or generating families of quantum operations indexed by parameters. These parameters need to be instantiated at compile time to generate concrete quantum circuit representations. Quipper has multiple formal specifications, but we will focus on the linear dependently typed Proto-Quipper-D formalization [19, 18] to express high-level programs with integer parameters.

In order to make use of the parametricity of the SZX calculus, we first introduce a *list initialization* notation to represent multiple elements of a SZX diagram family composed in parallel. We then formally define a fragment of Proto-Quipper-D programs that can be described as families of diagrams. Finally, we present a novel compilation method that encodes quantum programs as families of SZX diagrams and demonstrate the codification and translation of a nontrivial algorithm using this procedure.

The structure of the chapter is as follows: In Section 2.2 we outline both languages and introduce the list initialization notation. In Section 2.3 we define the restricted Proto-Quipper-D fragment containing the relevant operations for the work presented in this chapter. In Section 2.4 we introduce the translation into SZX diagrams. Finally, in Section 2.5 we demonstrate an encoding of the Quantum Fourier Transform algorithm using our method.

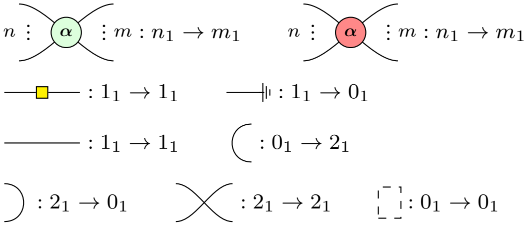
2.2 Background

We describe a quantum state as a system of n qubits corresponding to a vector in the $\mathbb{C}^{2^n}$ Hilbert space. We may partition the set of qubits into multi-qubit *registers* representing logically related subsets. Morally, these registers will be interpreted as vectors in our language. Quantum computations under the QRAM model correspond to compositions of unitary operators between these quantum states. Additionally, the qubits may be initialized on a set state and measured. However, we will not provide a probabilistic reduction to perform the measurements. We will only make the distinction through typing.

High-level programs can be encoded in Quipper [20], a Haskell-like programming language for describing quantum computations. In this work we use a formalization of the language called Proto-Quipper-D[18] with support for linear and dependent types. Concrete quantum operations correspond to linear functions between quantum states, generated as a composition of primitive operations that can be described directly as a quantum circuit. Generic circuits may have additional parameters that must fixed at compilation time to produce the corresponding quantum circuit.

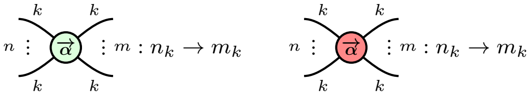
2.2.1 The Scalable ZX-calculus

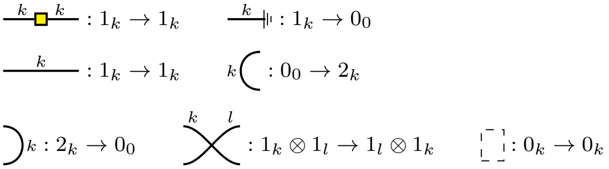
The ZX calculus [25] is a formal graphical language that encodes linear maps between quantum states. Multiple extensions to the calculus have been proposed. We first present the base calculus with the grounded-ZX extension, denoted $\text{ZX}_{⏚}$ [8], to allow us to encode quantum state measurement operations. A $\text{ZX}_{⏚}$ diagram is generated by the following primitives, in addition to parallel and serial composition:



where n_k represents the n-tensor of k-qubit registers, the green and red nodes are called Z and X spiders, $\alpha \in [0, 2\pi)$ is the phase of the spiders, and the yellow square is called the Hadamard node. These primitives allow us to encode any quantum operation, but they can become impractical when working with multiple qubit registers.

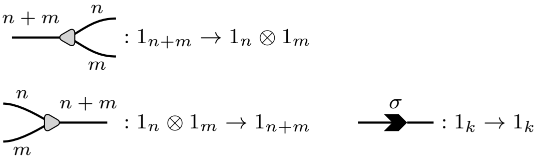
The SZX calculus [5, 4] is a *Scalable* extension to the ZX-calculus that generalizes the primitives to work with arbitrarily sized qubit registers. This facilitates the representation of diagrams with repeated structure in a compact manner. Carette et al. [4] show that the scalable and grounded extensions can be directly composed. We will refer to the resulting $\text{SZX}_{⏚}$-calculus as SZX for simplicity. Bold wires in a SZX diagram are tagged with a non-negative integer representing the size of the qubit register they carry, and other generators are marked in bold to represent a parallel application over each qubit in the register. Bold spiders with multiplicity k are tagged with k-sized vectors of phases $\overline{\alpha} = \alpha_1 :: \cdots :: \alpha_k$. The natural extension of the ZX generators correspond to the following primitives:



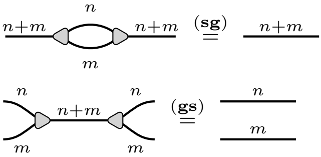


Wires of multiplicity zero are equivalent to the empty mapping. We may omit writing the wire multiplicity if it can be deduced by context.

The extension defines two additional generators; a *split* node to split registers into multiple wires, and a function arrow to apply arbitrary functions over a register. In this work we restrict the arrow functions to permutations $\sigma : [0 \dots k) \to [0 \dots k)$ that rearrange the order of the wires. Using the split node and the wire primitives can derive the rotated version, which we call a *gather*.

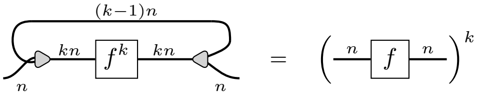

n + m
n
m
: $1_{n+m} \to 1_n \otimes 1_m$
n
n + m
m
: $1_n \otimes 1_m \to 1_{n+m}$
σ
: $1_k \to 1_k$

The rewriting rules of the calculus imply that a SZX diagrams can be considered as an open graph where only the topology of its nodes and edges matters. In the translation process we will make repeated use of the following reductions rules to simplify the diagrams:



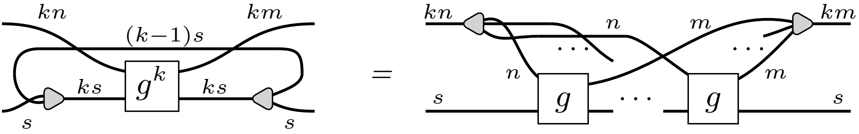
In an analogous manner, we will use a legless gather —◁ to terminate wires with cardinality zero. This could be encoded as the zero-multiplicity spider —([]), which represents the empty mapping. Refer to Appendix .1 for a complete definition of the rewriting rules and the interpretation of the SZX calculus. Cf. [4] for a description of the calculus including the generalized arrow generators.

Carette et al. [4] showed that the SZX calculus can encode the repetition of a function $f : 1_n \to 1_n$ an arbitrary number of times $k \geq 1$ as follows:



where f^k corresponds to k parallel applications of f. With a simple modification this construction can be used to encode an accumulating map operation.

Lemma 1. *Let $g : 1_n \otimes 1_s \to 1_m \otimes 1_s$ and $k \geq 1$, then*

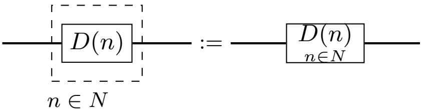


As an example, given a list $N = [n_1, n_2, n_3]$ and a starting accumulator value x_0, this construction would produce the mapping $([n_1, n_2, n_3], x_0) \mapsto ([m_1, m_2, m_3], x_3)$ where $(m_i, x_i) = g(n_i, x_{i-1})$ for $i \in [1, 3]$.

2.2.2 SZX diagram families and list instantiation

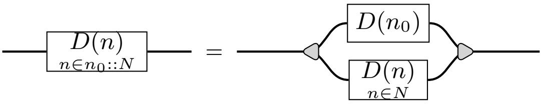
We introduce the definition of a family of SZX diagrams $D : \mathbb{N}^k \to \mathscr{D}$ as a function from k integer *parameters* to SZX diagrams. We require the structure of the diagrams to be the same for all elements in the family, parameters may only alter the wire tags and spider phases. Partial application is allowed, we write $D(n)$ to fix the first parameter of D.

Since instantiations of a family share the same structure, we can compose them in parallel by merging the different values of wire tags and spider phases. We introduce a shorthand for instantiating a family of diagrams on multiple values and combining the resulting diagrams in parallel. This definition is strictly more general than the *thickening endofunctor* presented by Carette et al. [4], which replicates a concrete diagram in parallel. A *list instantiation* of a family of diagrams $D : \mathbb{N}^{k+1} \to \mathscr{D}$ over a list N of integers is written as $(D(n), n \in N)$. This results in a family with one fewer parameter, $(D(n), n \in N) : \mathbb{N}^k \to \mathscr{D}$. We graphically depict a list instantiation as a dashed box in a diagram, as follows.

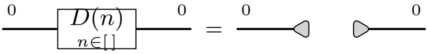


The definition of the list instantiation operator is given recursively on the construction of D in Figure 2.1. On the diagram wires we use $v(N)$ to denote the wire cardinality $\sum_{n\in N} v(n)$, $\overrightarrow{\alpha}(N)$ for the concatenation of phase vectors $\overrightarrow{\alpha}(n_1) :: \cdots :: \overrightarrow{\alpha}(n_m)$, and $\sigma(N)$ for the composition of permutations $\bigotimes_{n\in N} \sigma(n)$. In general, a permutation arrow $\sigma(N, v, w)$ instantiated in concrete values can be replaced by a reordering of wires between two gather gatesusing the rewrite rule (**p**).

Lemma 2. *For any diagram family D, $n_0 : \mathbb{N}$, $N : \mathbb{N}^k$,*



Lemma 3. *A diagram family initialized with the empty list corresponds to the empty map. For any diagram family D,*



Lemma 4. *The list instantiation procedure on an n-node diagram family adds $\mathcal{O}(n)$ nodes to the original diagram.*

2.3 The λ_D calculus

We first define a base language from which to build our translation. In this section we present the calculus λ_D, as a subset of the strongly normalizing Proto-Quipper-D programs. Terms are inductively defined by:

$$\begin{aligned} M, N, L := {}& x \mid C \mid \mathtt{R} \mid \mathtt{U} \mid 0 \mid 1 \mid n \mid \mathtt{meas} \mid \mathtt{new} \mid \lambda x^S.M \mid M\ N \mid \\ & \lambda' x^P.M \mid M\ @\ N \mid \star \mid M \otimes N \mid M;N \mid \\ & \mathtt{let}\ x^{S_1} \otimes y^{S_2} = M\ \mathtt{in}\ N \mid \mathtt{VNil}^A \mid M :: N \mid \\ & \mathtt{let}\ x^S :: y^{\mathtt{Vec}\ n\ S} = M\ \mathtt{in}\ N \mid M \square N \mid \end{aligned}$$

$$
\begin{aligned}
A &:= S \mid P \mid (n : \texttt{Nat}) \to A[n] \\
S &:= \texttt{B} \mid \texttt{Q} \mid \texttt{Unit} \mid S_1 \otimes S_2 \mid S_1 \multimap S_2 \mid \\
&\qquad \texttt{Vec}\ (n : \texttt{Nat})\ S \\
P &:= \texttt{Nat} \mid \texttt{Vec}\ (n : \texttt{Nat})\ \texttt{Nat}
\end{aligned}
$$

Table 2.1: *Type Grammar*

$$\texttt{ifz}\ L\ \texttt{then}\ M\ \texttt{else}\ N \mid \texttt{for}\ k^P\ \texttt{in}\ M\ \texttt{do}\ N$$

Where C is a set of implicit bounded recursive primitives used for operating with vectors and iterating functions. $n \in \mathbb{N}$, $\Box \in \{+, -, \times, /, \wedge\}$ and $\texttt{ifz}\ L\ \texttt{then}\ M\ \texttt{else}\ N$ is the conditional that tests for zero.

Here $\texttt{U}$ denotes a set of unitary operations and $\texttt{R}$ is a phase shift gate with a parametrized angle. In this article we fix the former to the CNOT and Hadamard (H) gates, and the latter to the arbitrary rotation gates $R_{z(\alpha)}$ and $R_{x(\alpha)}$.

For the remaining constants, 0 and 1 represent bits, $\texttt{new}$ is used to create a qubit, and $\texttt{meas}$ to measure it. $\star$ is the inhabitant of the $\texttt{Unit}$ type, and the sequence $M; N$ is used to discard it. Qubits can be combined via the tensor product $M \otimes N$ with $\texttt{let}\ x^{S_1} \otimes y^{S_2} = M\ \texttt{in}\ N$ as its corresponding destructor.

The system supports lists; $\texttt{VNil}^A$ represents the empty list, $M :: N$ the constructor and $\texttt{let}\ x^S :: y^{\texttt{Vec}\ n\ S} = M\ \texttt{in}\ N$ acts as the destructor. Finally, the term $\texttt{for}\ k^P\ \texttt{in}\ M\ \texttt{do}\ N$ allows iterating over parameter lists. The typing system is defined in Figure **??**. We write $|\Phi|$ for the list of variables in a typing context Φ. The type $\texttt{Vec}\ n\ A$ represents a vector of known length n of elements of type A.

We differentiate between *state contexts* (Noted with Γ and Δ) and *parameter contexts* (Noted with Φ). For our case of study, parameter contexts consist only of pairs $x : \texttt{Nat}$ or $x : \texttt{Vec}\ (n : \texttt{Nat})\ \texttt{Nat}$, since they are the only non-linear types of variables that we manage. Every other variable falls under the state context. The terms $\lambda x^S.M$ and MN correspond to the abstraction and application which will be used for state-typed terms. The analogous constructions for parameter-typed terms are $\lambda' x^P.M$ and $M@N$.

In this sense we deviate from the original Proto-Quipper-D type system, which supports a single context decorated with indices. Instead, we use a linear and non-linear approach similar to the work of Cervesato and Pfenning[7].

Types are divided into two kinds; parameter and state types. Both of these can depend on terms of type $\texttt{Nat}$. For the scope of this work, this dependence may only influence the size of vectors types.

Parameter types represent non-linear variable types which are known at the time of generation of the concrete quantum operations. In the translation into SZX diagrams, these variables may dictate the labels of the wires and spiders. Vectors of $\texttt{Nat}$ terms represent their cartesian product. On the other hand, state types correspond to the quantum operations and states to be computed. In the translation, these terms inform the shape and composition of the diagrams. Vectors of state type terms represent their tensor product.

In lieu of unbounded and implicit recursion, we define a series of primitive functions for performing explicit vector manipulation. These primitives can be defined in the original language, with the advantage of them being strongly normalizing. The first four primitives are used to manage state vectors, while the last one is used for generating parameters. For ease of translation some terms are decorated with type annotations, however we will omit these for clarity when the type is apparent.

$$\Phi \vdash \texttt{accuMap}_{A,B,C} : (n : \texttt{Nat}) \to \texttt{Vec}\ n\ A \\ \multimap \texttt{Vec}\ n\ (A \multimap C \multimap B \otimes C) \multimap C \multimap (\texttt{Vec}\ n\ B) \otimes C$$

$$\Phi \vdash \texttt{split}_A : (n : \texttt{Nat}) \to (m : \texttt{Nat}) \to \texttt{Vec}\ (n+m)\ A \\ \multimap \texttt{Vec}\ n\ A \otimes \texttt{Vec}\ m\ A$$

$$\Phi \vdash \texttt{append}_A : (n : \texttt{Nat}) \to (m : \texttt{Nat}) \to \texttt{Vec}\ n\ A \multimap \texttt{Vec}\ m\ A \\ \multimap \texttt{Vec}\ (n+m)\ A$$

$$\Phi \vdash \texttt{drop} : (n : \texttt{Nat}) \to \texttt{Vec}\ n\ \texttt{Unit} \multimap\ \texttt{Unit}$$

$$\Phi \vdash \texttt{range}\ : (n : \texttt{Nat}) \to (m : \texttt{Nat}) \to \texttt{Vec}\ (m-n)\ \texttt{Nat}$$

Since every diagram represents a linear map between qubits there is no representation equivalent to non-terminating terms, even for weakly normalizing programs. This is the main reason behind the design choice of the primitives set.

We include the operational semantics of the calculus and primitives in Appendix .2. The encoding of the primitives as Proto-Quipper-D functions is shown in Appendix .3.

We additionally define the following helpful terms based on the previous primitives to aid in the manipulation of vectors. Cf. Appendix .2 for their definition as λ_D-terms.

$$\Phi \vdash \texttt{map}_{A,B} : (n : \texttt{Nat}) \to \texttt{Vec}\ n\ A \multimap\ \texttt{Vec}\ n\ (A \multimap B) \multimap \texttt{Vec}\ n\ B$$
$$\Phi \vdash \texttt{fold}_{A,C} : (n : \texttt{Nat}) \to \texttt{Vec}\ n\ A \multimap\ \texttt{Vec}\ n\ (A \multimap C \multimap C) \multimap C \multimap C$$
$$\Phi \vdash \texttt{compose}_A : (n : \texttt{Nat}) \to \texttt{Vec}\ n\ (A \multimap A) \multimap A \multimap A$$

The distinction between primitives that deal with state and parameters highlights the inclusion of the `for` as a construction into the language instead of a primitive. Since it acts over both parameter and state types, its function is effectively to bridge the gap between the two of them. This operation closely corresponds to the list instantiation procedure presented in the Section 2.2.1.

For example, if we take ns to be a vector of natural numbers, and xs a vector of abstractions $R@k(\texttt{new}\ 0)$. The term $\texttt{for}\ k\ \texttt{in}\ ns\ \texttt{do}\ xs$ generates a vector of quantum maps by instantiating the abstractions for each individual parameter in ns.

2.4 Encoding programs as diagram families

A key difference between Quipper (and, by extension, Proto-Quipper-D) and λ_D is the approach to defining circuits. In Quipper, circuits are an intrinsic part of the language and can be operated upon. In our case, the translation into SZX diagrams will be mediated with a function defined outside the language.

In this section we introduce an encoding of the lambda calculus presented in Section 2.3 into families of SZX diagrams with context variables as inputs and term values as outputs. We split the lambda-terms into those that represent linear mappings between quantum states and can be encoded as families of SZX diagrams, and parameter terms that can be completely evaluated at compile-time.

2.4.1 Parameter evaluation

We say a type is *evaluable* if it has the form $A = (n_1 : \texttt{Nat}) \to \cdots \to (n_k : \texttt{Nat}) \to P[n_1, \ldots, n_k]$ with P a parameter type. Since A does not encode a quantum operation, we interpret it directly into functions over vectors of natural numbers. The translation of an evaluable type, $\lfloor A \rfloor$, is defined recursively as follows:

$$\lfloor (n : \texttt{Nat}) \to B[n] \rfloor = \mathbb{N} \to \bigcup_{n \in \mathbb{N}} \lfloor B[n] \rfloor \qquad \lfloor \texttt{Nat} \rfloor = \mathbb{N}$$

$$\lfloor \mathtt{Vec}\ (n : \mathtt{Nat})\ \mathtt{Nat} \rfloor = \mathbb{N}^n$$

Given a type judgement $\Phi \vdash L : P$ where P is an evaluable type, we define $\lfloor L \rfloor_\Phi$ as the evaluation of the term into a function from parameters into products of natural numbers. Since the typing is syntax directed, the evaluation is defined directly over the terms as follows:

$$\lfloor x \rfloor_{x:\mathtt{Nat},\Phi} = x, |\Phi| \mapsto x \qquad \lfloor n \rfloor_\Phi = |\Phi| \mapsto n$$

$$\lfloor M \square N \rfloor_\Phi = |\Phi| \mapsto \lfloor M \rfloor_\Phi (|\Phi|) \square \lfloor N \rfloor_\Phi (|\Phi|)$$

$$\lfloor M\ ::\ N \rfloor_\Phi = |\Phi| \mapsto \lfloor M \rfloor_\Phi (|\Phi|) \times \lfloor N \rfloor_\Phi (|\Phi|)$$

$$\left\lfloor \mathtt{VNil}^{\mathtt{Nat}} \right\rfloor_\Phi = |\Phi| \mapsto [\,]$$

$$\left\lfloor \lambda' x^P . M \right\rfloor_\Phi = x, |\Phi| \mapsto \lfloor M \rfloor_\Phi (x, |\Phi|)$$

$$\lfloor M @ N \rfloor_\Phi = \lfloor M \rfloor_\Phi (\lfloor N \rfloor_\Phi (|\Phi|), \Phi)$$

$$\lfloor \mathtt{ifz}\ L\ \mathtt{then}\ M\ \mathtt{else}\ N \rfloor_\Phi = |\Phi| \mapsto \begin{cases} \lfloor M \rfloor_\Phi (|\Phi|) \text{ if } \lfloor L \rfloor_\Phi (|\Phi|) = 0 \\ \lfloor N \rfloor_\Phi (|\Phi|) \text{ otherwise} \end{cases}$$

$$\lfloor \mathtt{range}\ \rfloor_\Phi = n, m, |\Phi| \mapsto \bigtimes_{i=n}^{m-1} i$$

$$\lfloor \mathtt{for}\ k\ \mathtt{in}\ V\ \mathtt{do}\ M \rfloor_\Phi = |\Phi| \mapsto \bigtimes_{k \in \lfloor V \rfloor_\Phi (|\Phi|)} \lfloor M \rfloor_{k:\mathtt{Nat}\Phi} (k, |\Phi|)$$

$$\begin{aligned} &\left\lfloor \mathtt{let}\ x^P :: y^{\mathtt{Vec}\ n\ P} = M\ \mathtt{in}\ N \right\rfloor_\Phi = \\ &\qquad |\Phi| \mapsto \lfloor N \rfloor_{x:P, y:\mathtt{Vec}\ n\ P, \Phi} (y_1, [y_2, \dots, y_n], |\Phi|) \\ &\text{where } [y_1, \dots, y_n] = \lfloor M \rfloor_\Phi (|\Phi|) \end{aligned}$$

Lemma 5. *Given an evaluable type A and a type judgement $\Phi \vdash L : A$, $\lfloor L \rfloor_\Phi \in \bigtimes_{x:P \in \Phi} \lfloor P \rfloor \to \lfloor A \rfloor$.*

Lemma 6. *Given an evaluable type A, a type judgement $\Phi \vdash L : A$, and $M \to N$, then $\lfloor M \rfloor_\Phi = \lfloor N \rfloor_\Phi$.*

2.4.2 Diagram encoding

A non-evaluable type has necessarily the form $A = (n_1 : \mathtt{Nat}) \to \cdots \to (n_k : \mathtt{Nat}) \to S$, with S any state type. We call such types *translatable* since they correspond to terms that encode quantum operations that can be described as families of diagrams.

We first define a translation $[\![\cdot]\!]$ from state types into wire multiplicities as follows. Notice that due to the symmetries of the SZX diagrams the linear functions have the same representation as the products.

$$[\![\mathtt{B}]\!] = 1 \qquad [\![\mathtt{Q}]\!] = 1 \qquad [\![\mathtt{Vec}\ (n : \mathtt{Nat})\ A]\!] = [\![A]\!]^{\otimes n}$$

$$[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!] \qquad [\![A \multimap B]\!] = [\![A]\!] \otimes [\![B]\!]$$

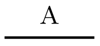
Given a translatable type judgement $\Phi, \Gamma \vdash M : (n_1 : \mathtt{Nat}) \to \cdots \to (n_k : \mathtt{Nat}) \to S$ we can encode it as a family of SZX diagrams

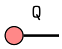
$$n_1, \dots, n_k, |\Phi| \mapsto \overset{[\![\Gamma]\!]}{\text{———}}\boxed{M(|\Phi|)}\overset{[\![S[|\Phi|]]\!]}{\text{———}}$$

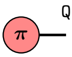
.

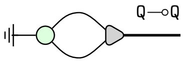
We will omit the brackets in our diagrams for clarity. In a similar manner to the evaluation, we define the translation $[\![M]\!]_{\Phi,\Gamma}$ recursively on the terms as follows:

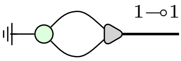
ARREGLAR ESTA PARTE QUE ESTÁ ILEGIBLE

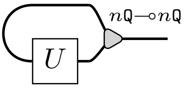
$[\![x]\!]_{\Phi,x:A} = |\Phi| \mapsto$ 

$[\![0]\!]_{\Phi} = |\Phi| \mapsto$ 

$[\![1]\!]_{\Phi} = |\Phi| \mapsto$ 

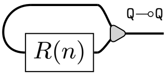
$[\![\texttt{meas}]\!]_{\Phi} = |\Phi| \mapsto$ 

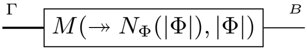
$[\![\texttt{new}]\!]_{\Phi} = |\Phi| \mapsto$ 

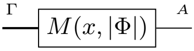
$[\![U]\!]_{\Phi} = |\Phi| \mapsto$ 

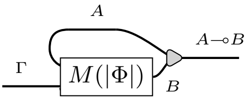
$[\![\star]\!]_{\Phi} = |\Phi| \mapsto$

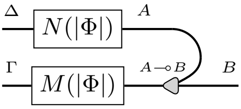
$[\![\texttt{VNil}]\!]_{\Phi} = |\Phi| \mapsto$

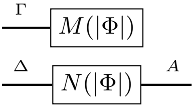
$[\![R]\!]_{\Phi} = n, |\Phi| \mapsto$ 

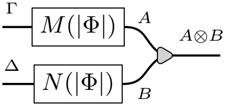
$[\![M \ @N]\!]_{\Phi,\Gamma} = |\Phi| \mapsto$ 

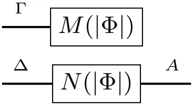
$[\![\lambda' x^A.M]\!]_{\Phi,\Gamma} = x, |\Phi| \mapsto$ 

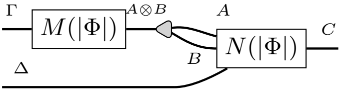
$[\![\lambda x^A.M]\!]_{\Phi,\Gamma} = |\Phi| \mapsto$ 

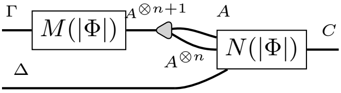
$[\![M \ N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

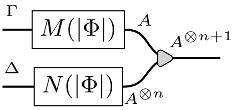
$[\![M; N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

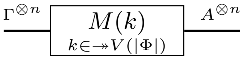
$[\![M \otimes N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

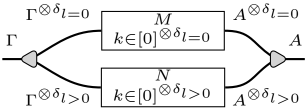
$[\![M;_v N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

$[\![let \ x^A \otimes y^B = M \ in \ N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

$[\![let \ x^A : y^{\texttt{Vec} \ n \ A} = M \ in \ N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

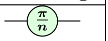
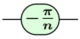
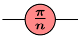
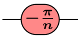
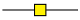
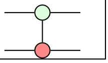
$[\![M :: N]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$ 

$[\![\texttt{for } k \texttt{ in } V \texttt{ do } M]\!]_{\Phi,\Gamma^n} = |\Phi| \mapsto$ 

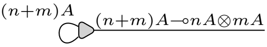
$[\![\texttt{ifz } L \texttt{ then } M \texttt{ else } N]\!]_{\Phi,\Gamma} = |\Phi| \mapsto$ 

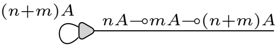
Where δ is the Kronecker delta and $l = \lfloor L \rfloor (|\Phi|)$. Notice that the `new` and `meas` operations share the same translation. Although `new` can be encoded as a simple wire, we keep the additional node to maintain the symmetry with the measurement.

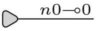
The unitary operators U and rotations R correspond to a predefined set of primitives, and their translation is defined on a by case basis. The following table shows the encoding of the operators used in this paper.

Name	Encoding
Rz(n)	
Rz^{-1}(n)	
Rx(n)	
Rx^{-1}(n)	
H	
CNOT	

The primitives `split`, `append`, `drop` and `accuMap` are translated below. Since vectors are isomorphic to products in the wire encoding, the first three primitives do not perform any operation. We present the translation for these primitives first.

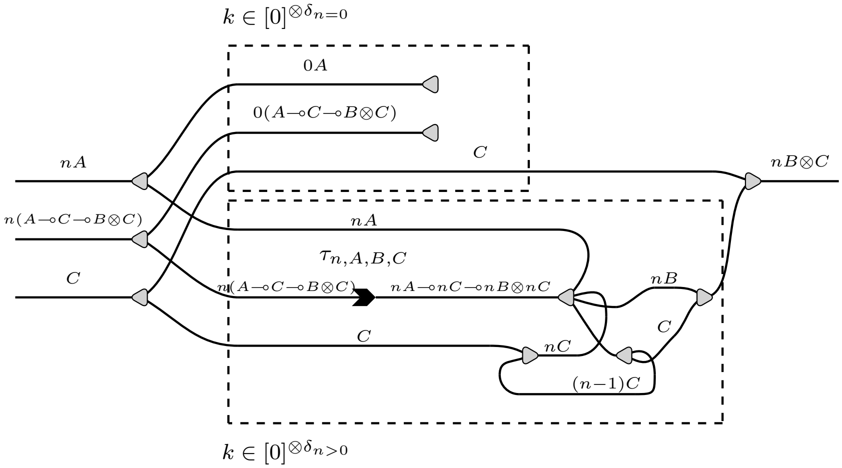
$[\![\texttt{split}_A]\!]_\Phi = n, m, |\Phi| \mapsto$ 

$[\![\texttt{append}_A]\!]_\Phi = n, m, |\Phi| \mapsto$ 

$[\![\texttt{drop}]\!]_\Phi = n, |\Phi| \mapsto$ 

For the accumulating map we utilize the construction presented in Lemma 1, replacing the function box with a function vector input. In the latter we omit the wires and gathers connecting the inputs and outputs of the function to a single wire on the right of the diagram for clarity.

$$\left[\!\left[\texttt{accuMap}_{A,B,C}\right]\!\right]_{\Phi} = n, |\Phi| \mapsto$$


$k \in [0]^{\otimes \delta_{n=0}}$
$0A$
$0(A \multimap C \multimap B \otimes C)$
C
$nB \otimes C$
nA
$n(A \multimap C \multimap B \otimes C)$
C
nA
$\tau_{n,A,B,C}$
$n(A \multimap C \multimap B \otimes C)$
$nA \multimap nC \multimap nB \otimes nC$
nB
C
C
nC
$(n-1)C$
$k \in [0]^{\otimes \delta_{n>0}}$

Where $\tau_{n,A,B,C}$ is a permutation that rearranges the vectors of functions into tensors of vectors for each parameter and return value. That is, $\tau_{n,A,B,C}$ reorders a sequence of registers $(A, C, B, C) \dots (A, C, B, C)$ into the sequence $(A \dots A)(C \dots C)(B \dots B)(C \dots C)$. It is defined as follows:

$$\tau_{n,A,B,C}(i) = \begin{cases} i \bmod k + a * (i \text{ div } k) \\ \qquad\qquad \text{if } i \bmod k < a \\[1em] i \bmod k + c * (i \text{ div } k) + a * (n-1) \\ \qquad\qquad \text{if } a \leq i \bmod k < (a+c) \\[1em] i \bmod k + b * (i \text{ div } k) + (a+c) * (n-1) \\ \qquad\qquad \text{if } (a+c) \leq i \bmod k < (a+c+b) \\[1em] i \bmod k + c * (i \text{ div } k) + (a+c+b) * (n-1) \\ \qquad\qquad \text{if } (a+c+b) \leq i \bmod k \end{cases}$$

For $i \in [0, (a+c+b+c)*n)$, where mod and div are the integer modulo and division operators, $a = [\![A]\!]$, $b = [\![B]\!]$, $c = [\![C]\!]$, and $k = a+c+b+c$.

As a consequence of Lemma 4, the number of nodes in the produced diagrams grows linearly with the size of the input. Notice that the ZX spiders, the ground, and the Hadamard operator are only produced in the translations of the quantum primitives. We may instead have used other variations of the calculus supporting the scalable extension, such as the ZH calculus [1], better suited for other sets of quantum operators.

Lemma 7. *The translation procedure is correct in respect to the operational semantics of λ_D. If A is a translatable type, $\Phi, \Gamma \vdash M : A$, and $M \to N$, then $[\![M]\!]_{\Phi,\Gamma} = [\![N]\!]_{\Phi,\Gamma}$.*

2.5 Application example: QFT

The Quantum Fourier Transform is an algorithm used extensively in quantum computation, notably as part of Shor's algorithm for integer factorization [22]. The QFT function operates generically over n-qubit states and in general a circuit encoding of it requires $\mathcal{O}(n^2)$ gates. In this section we present an encoding of the algorithm as a λ_D term, followed by the translation into a family of constant-sized diagrams. The corresponding Proto-Quipper-D program is listed in Appendix .4.

The following presentation divides the algorithm into three parts. The *crot* term applies a controlled rotation over a qubit with a parametrized angle. *apply_crot* operates over the last $n-k$ qubits of an n-qubit state by applying a Hadamard gate to the first one and then using it as target of successive *crot* applications using the rest of the qubits as controls. Finally, *qft* repeats *apply_crot* for all values of k. In the terms, we use $n \dots m$ as a shorthand for `range` $@n$ $@m$.

crot: $(n : \texttt{Nat}) \to (\texttt{Q} \otimes \texttt{Q}) \multimap (\texttt{Q} \otimes \texttt{Q})$

$$\begin{aligned}
\text{crot} :=\ & \lambda' n^{\texttt{Nat}}.\ \lambda qs^{\texttt{Q}\otimes\texttt{Q}}. \\
& \quad \texttt{let } c^{\texttt{Q}} \otimes q^{\texttt{Q}} = qs \texttt{ in} \\
& \quad \texttt{let } c^{\texttt{Q}} \otimes q^{\texttt{Q}} = \texttt{CNOT}\ c\ (\texttt{Rz}\ @2^n\ q) \texttt{ in} \\
& \qquad \texttt{CNOT}\ c\ (\texttt{Rz}^{-1}\ @2^n\ q)
\end{aligned}$$

apply_crot: $(n : \texttt{Nat}) \to (k : \texttt{Nat}) \to \texttt{Vec}\ n\ \texttt{Q} \multimap \texttt{Vec}\ n\ \texttt{Q}$

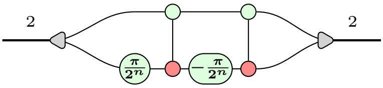
$$\begin{aligned}
\text{apply\_crot} :=\ & \lambda' n^{\texttt{Nat}}.\ \lambda' k^{\texttt{Nat}}.\ \lambda qs^{\texttt{Vec}\ n\ \texttt{Q}}. \\
& \texttt{ifz } (n-k) \texttt{ then } qs \texttt{ else} \\
& \texttt{let } h^{\texttt{Vec}\ k\ \texttt{Q}} \otimes qs'^{\ \texttt{Vec}\ n-k\ \texttt{Q}} = \texttt{split}\ @k\ @(n-k)\ qs \texttt{ in} \\
& \texttt{let } q^{\texttt{Q}} \otimes cs^{\texttt{Vec}\ n-k-1\ \texttt{Q}} = qs' \texttt{ in} \\
& \texttt{let } fs^{\texttt{Vec}\ (n-k-1)\ (\texttt{Q}\otimes\texttt{Q}\multimap\texttt{Q}\otimes\texttt{Q})} = \texttt{for } m^{\texttt{Nat}} \texttt{ in } 2..(n-k+1) \texttt{ do } \text{crot}\ @m \texttt{ in} \\
& \texttt{let } cs'^{\ (\texttt{Vec}\ n-k-1\ \texttt{Q})} \otimes q'^{\ \texttt{Q}} = \texttt{accuMap}\ fs\ (H\ q)\ cs \texttt{ in} \\
& \text{concat}\ h\ (q' : cs')
\end{aligned}$$

qft : $(n : \texttt{Nat}) \to \texttt{Vec}\ n\ \texttt{Q} \multimap \texttt{Vec}\ n\ \texttt{Q}$

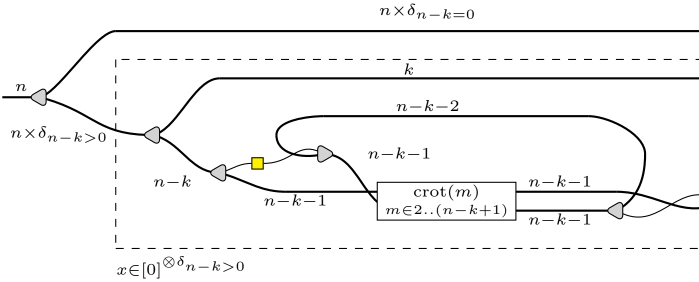
$$\begin{aligned}
\text{qft} :=\ & \lambda' n^{\texttt{Nat}}.\lambda qs^{\texttt{Vec}\ n\ \texttt{Q}}.\texttt{compose} \\
& (\texttt{for } k^{\texttt{Nat}} \texttt{ in } \text{reverse\_vec}\ @(0..n) \texttt{ do } \lambda qs'^{\ \texttt{Vec}\ n\ \texttt{Q}}.\text{apply\_crot}\ @n\ @k\ qs')\ qs
\end{aligned}$$

The translation of each term into a family of diagrams is shown below. We omit the wire connecting the function inputs to the right side of the graphs for clarity and eliminate superfluous gathers and splitters using rules (**sg**) and (**gs**). Notice that, in contrast to a quantum circuit encoding, the resulting diagram's size does not depend on the number of qubits n.

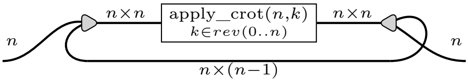
$[\![\text{crot}]\!] = \ n \mapsto$



$[\![\text{apply\_crot}]\!] = \ n, k \mapsto$


$n \times \delta_{n-k=0}$
n
$n \times \delta_{n-k>0}$
k
$n-k-2$
$n-k-1$
$n-k$
$n-k-1$
crot(m)
$m \in 2..(n-k+1)$
$n-k-1$
$n-k-1$
$x \in [0]^{\otimes \delta_{n-k>0}}$

$[\![\text{qft}]\!] = \ n \mapsto$



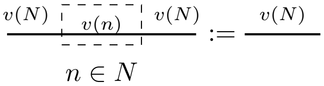
2.6 Conclusion

In this chapter, we presented an efficient method to compile parametric quantum programs written in a fragment of the Proto-Quipper-D language into families of SZX diagrams. We restricted the fragment to strongly normalizing terms that can be represented as diagrams. Additionally, we introduced a notation to easily compose elements of a diagram family in parallel. We proved that our method produces compact diagrams and shown that it can encode non-trivial algorithms.

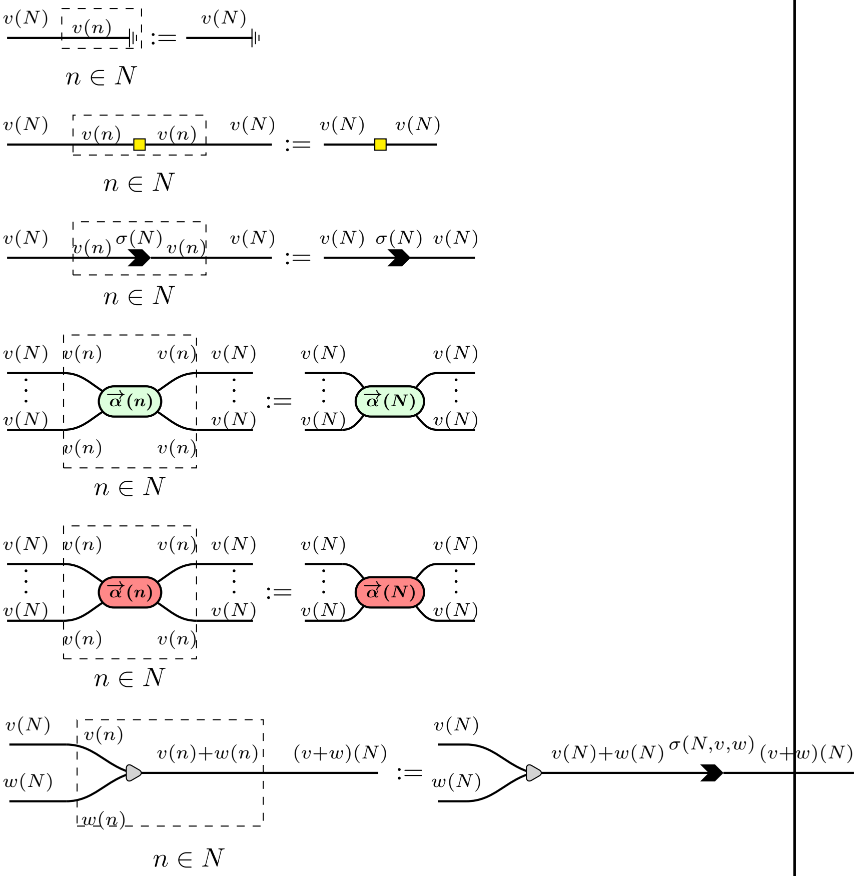
A current line of work is defining categorical semantics for the calculus and families of diagrams, including a subsequent proof of adequacy for the translation. More work needs to be done to expand the fragment of the Quipper language that can be translated.

Given $D : \mathbb{N}^{k+1} \to \mathscr{D}$, $N = [n_1, \ldots, n_m] \in \mathbb{N}^m$,

$$((D_1 \otimes D_2)(n), n \in N) := (D_1(n), n \in N) \otimes (D_2(n), n \in N)$$



$$((D_2 \circ D_1)(n), n \in N) := (D_2(n), n \in N) \circ (D_1(n), n \in N)$$



Where $\sigma(N, v, w) \in \mathbb{F}_2^{v(N)+w(N) \times v(N)+w(N)}$ is the permutation defined as the matrix

$$\sigma(N, v, w) = \left(\sigma_f^N | \sigma_g^N\right), \quad \sigma_f^N \in \mathbb{F}_2^{v(N)+w(N) \times v(N)}, \ \sigma_g^N \in \mathbb{F}_2^{v(N)+w(N) \times w(N)}$$

$$\sigma_f^{[]} = Id_0 \quad \sigma_f^{n::N'} = \begin{pmatrix} Id_{v(n)} & 0 \\ 0 & 0 \\ 0 & \sigma_f^{N'} \end{pmatrix} \quad \sigma_g^{[]} = Id_0 \quad \sigma_g^{n::N'} = \begin{pmatrix} 0 & 0 \\ Id_{w(n)} & 0 \\ 0 & \sigma_g^{N'} \end{pmatrix}$$

Figure 2.1: *Definition of the list instantiation operator.*

Chapter 3

Basis-Sensitive Quantum Typing via Realizability

3.1 Introduction

We previously presented the impossibility theorems which stated that is physically impossible to copy or delete a qubit. There is however, a subtlety in these impossibility theorems. Arbitrary qubits cannot be copied, but it is indeed possible to do so with known qubits. This implies that qubits with known values behave as classical data and can be treated accordingly. Moreover, it suffices to know the basis to which a qubit belongs in order to copy and delete it. This is a known fact in quantum information theory which underlies a number of quantum algorithms.

In most quantum programming languages, qubits are interpreted in a canonical basis (often called the computational basis). In this fashion, classical bits are represented by the basis vectors, and qubits as norm-1 linear combinations of bits. We are allowed to copy and delete classical bits freely, while such operations on arbitrary qubits remain restricted.

In this chapter we will introduce a quantum lambda calculus in the quantum-data / quantum-control paradigm. It uses as starting point the calculus defined in [13], which was introduced using a realizability technique. In the same manner, our aim is to follow this workflow to extract a type system able to track bases throughout the programs. This should allow us to treat qubit in known bases classically, while still handling unknown qubits linearly.

To do this, we will decorate abstractions with the basis it is working in. Morally, the reduction system will consider values in that basis as its classical data. In the same manner, linear combination of these elements will represent quantum data and reduce linearly over the term.

In 1945, Kleene introduced in [21] the notion of realizability as a semantics for Heyting arithmetic. Since then, it has evolved and found applications both in proof theory and functional programming. In our case, we will use it for extracting type systems from the operational semantics of a calculus, resulting in a system in which safety properties hold by construction.

The steps to define a programming language using this technique are as follows. First, define a calculus equipped with a deterministic evaluation strategy. Second, define types as sets of closed values in the language, optionally introducing operations to build more complex types. Third, define the typing judgement $\Gamma \vdash t : A$, where Γ is a context of typed variables, t a term in the calculus, and A a type, as the property that for every substitution θ that map variables in Γ to closed values of their respective type, the term $\theta(t)$ reduces to a value in A, i.e., $\theta(t) \twoheadrightarrow v \in A$.

In this setting, each typing rule corresponds to a provable theorem. For instance, if $\Gamma \vdash t : A$ implies $\Delta \vdash r : B$, then the following rule is valid:

$$\frac{\Gamma \vdash t : A}{\Delta \vdash r : B}$$

The main advantage of using realizability is that it provides us with a framework to define *families of type systems*. We do not build the typing from ad-hoc rules, rather we define them according to the computational content of the calculus. We will present a set of rules which we deem adequate for a basic programming language. But, this set can be extended just as easily by proving the validity of new rules.

The final aim of this chapter is twofold. First, to make use of this extracted type system to give a more accurate description of programs. Second, to take advantage of the syntax of the modified calculus to write algorithms in a more versatile manner, instead of simply translating from a circuit.

The idea of keeping track of non-computational bases has been previously explored; see, for example [23, 16]. In [23], Perdrix introduces an abstract model which keeps track of the basis of qubits which later utilizes to make static analysis of entanglement throughout the program.

In [16], Monzon and Díaz-Caro present a lambda-calculus which integrates some basis information of qubits into the type system. They then continue to prove meta-theoretic and safety properties, showing the calculus to be a strong proof-of-concept for basis analysis in type systems. Indeed, we will expand on these ideas on this chapter.

An important point to note, is that both of these systems are focused on the canonical basis alongside the Hadamard basis. Just taking into account these two bases already proved fruitful, however there are still improvements to be made.

First, the use of single-qubit bases does not take into account bases formed by multiple entangled qubits. That is, bases of vector spaces of higher dimensions which cannot be written as the product of two smaller bases.

Second, the calculus [16] sacrifices higher-order computations as a trade-off for functions that act exclusively on qubits. We wish to recover this feature, that aligns with our aim of writing more flexible algorithms akin to modern programming languages.

The structure of the chapter is as follows: In section 3.2, we define the syntax for the calculus. Then, in section 3.3 we detail the reduction system. We define the type algebra and prove a set of valid typing rules in section 3.4. With the calculus fully defined, we showcase a few examples in section 3.5. We give closing remarks and discuss future work in 3.6.

3.2 The calculus

3.2.1 Syntax

This section presents the calculus upon which our realizability model will be designed. It is a lambda-calculus extended with linear combinations of lambda-terms, which form a vector space.

The syntax of the calculus is described in Table 3.1. It is divided into four distinct syntactic categories: *pure values, pure terms, value distributions and term distributions.* Values are composed by variables, a decorated lambda abstraction and two boolean values representing perpendicular vectors: $|0\rangle$ and $|1\rangle$. A pair of values is also a value itself. Terms include values, applications, pair constructors and destructors and pattern-matching testing for orthogonal vectors represented by the case operator. Both terms and value distributions are built by a $\mathbb{C}$-linear combination of either terms or values respectively. In Table 3.2 we also include notation for linear distributions of pairs. We stress that this notation for pairs does not appear in the syntax, but is rather helpful to describe a particular state.

$$\begin{array}{lcll} v & ::= & x \mid \lambda x_B \,.\, \vec{t} \mid (v,v) \mid |0\rangle \mid |1\rangle & \\ t & ::= & w \mid t\,t \mid \mathsf{let}_{(B,B)}\ (x,y) = \vec{t}\ \mathsf{in}\ \vec{t} \mid & \\ & & \mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v} \mapsto \vec{t} \mid \cdots \mid \vec{v} \mapsto \vec{t}\} & \\ \vec{v} & ::= & v \mid \vec{v}+\vec{v} \mid \alpha\,\vec{v} & (\alpha \in \mathbb{C}) \\ \vec{t} & ::= & t \mid \vec{t}+\vec{t} \mid \alpha\,\vec{t} & (\alpha \in \mathbb{C}) \end{array}$$

Where B is an n-th dimensional orthonormal basis as defined in Def. 2.

Table 3.1: *Syntax of the calculus*

$$\begin{array}{rcl} (\alpha\ v + \vec{v}_1, \vec{v}_2) & := & \alpha(v, \vec{v}_2) + (\vec{v}_1, \vec{v}_2) \\ (w, \alpha\ v + \vec{v}_1) & := & \alpha(w, v) + (w, \vec{v}_1) \end{array}$$

Where v, w are pure values and $\vec{v_1}, \vec{v}_2$ value distributions.

Table 3.2: *Notation for writing pair distributions*

Remark 1. *We do not include a specific term representing the null vector $\vec{0}$ since we do not make use of it. Instead, any distribution $0\,\vec{t}$ will act as it.*

In order to handle the different bases in each abstraction, we need to define the congruence relation between values from Table 3.3. When we define the reduction system, this congruence will allow us to take an argument and interpret it in the corresponding basis of the function. Here, the structure of value distributions starts to take shape. The term and value distributions stop being merely syntactic terms and start acting as proper linear combinations. Since the congruence enables the associativity of the addition, we will use Σ notation to represent sums.

The set of value distributions does not form a vector space, we can easily check this fact from the lack of a neutral element (which also entails a lack of additive inverse for elements). Instead, we work with a distributive-action space as described in [15].

A distributive-action space over a field K is a commutative semi-group $(V, +)$ equipped with a scalar multiplication $(\cdot) : K \times V \to V$ such that for all $\vec{v}, \vec{w} \in V, \alpha, \beta \in K$ it satisfies the following equations:

$$\begin{array}{rr} 1 \cdot \vec{v} = \vec{v} & (\alpha + \beta) \cdot \vec{v} = \alpha \cdot \vec{v} + \beta \cdot \vec{v} \\ \alpha \cdot (\beta \vec{v}) = \alpha\beta \cdot \vec{v} & \alpha \cdot (\vec{v} + \vec{w}) = \alpha \cdot \vec{v} + \alpha \cdot \vec{w} \end{array}$$

In this case we take $\mathbb{C}$ as our field K and, we omit the dot for the scalar product. It is clear from the rules in Table 3.3 that the set of distribution values satisfy the axioms of a distributive-action space. Moreover, the first rule of the congruence simulates the behaviour of the null vector for some $\vec{v}$.

We expand on the rationale for the first rule of Table 3.3 ($\vec{v_1} + 0\,\vec{v_2} \equiv \vec{v_1}$). The main idea of the calculus is to decompose the vectors corresponding to the arguments onto the bases attached to the abstractions. Taking an example from linear algebra, if we were to rewrite the vector $(1, 0)$ as a linear combination of $\{\frac{(1,1)}{\sqrt{2}}, \frac{(1,-1)}{\sqrt{2}}\}$ we would get:

$$\begin{aligned} (1, 0) &= (1, 0) + 0\ (0, 1) \\ &= \frac{1}{2}((1, 0) + (1, 0) + (0, 1) - (0, 1)) \end{aligned}$$

$\vec{v_1} + 0\,\vec{v_2} \;\equiv\; \vec{v_1}$

Where the $\vec{v_i}$ are neither an abstraction, nor a variable

$$1\,\vec{t} \;\equiv\; \vec{t} \qquad\qquad \alpha\,(\beta\,\vec{t}) \;\equiv\; \delta\,\vec{t}$$

Where: $\delta = \alpha\beta$

$$\vec{t_1} + \vec{t_2} \;\equiv\; \vec{t_2} + \vec{t_1} \qquad (\vec{t_1} + \vec{t_2}) + \vec{t_3} \;\equiv\; \vec{t_1} + (\vec{t_2} + \vec{t_3})$$

$$(\alpha + \beta)\,\vec{t} \;\equiv\; \alpha\,\vec{t} + \beta\,\vec{t}$$

$$\alpha\,(\vec{t_1} + \vec{t_2}) \;\equiv\; \alpha\,\vec{t_1} + \alpha\,\vec{t_2}$$

$$\vec{t}(\alpha\vec{s}) \;\equiv\; \alpha(\vec{t}\vec{s}) \qquad\qquad (\alpha\vec{t})\vec{s} \;\equiv\; \alpha(\vec{t}\vec{s})$$

$$(\vec{t} + \vec{s})\vec{r} \;\equiv\; \vec{t}\vec{r} + \vec{s}\vec{r} \qquad\qquad \vec{t}(\vec{s} + \vec{r}) \;\equiv\; \vec{t}\vec{s} + \vec{t}\vec{r}$$

$$\mathsf{let}_{(A_1,B_2)}\ (x_1, x_2) = (\alpha\vec{t})\ \mathsf{in}\ \vec{s} \equiv \alpha(\mathsf{let}_{(A_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{s})$$

$$\begin{aligned}\mathsf{let}_{(A_1,B_2)}\ (x_1, x_2) = \vec{t} + \vec{s}\ \mathsf{in}\ \vec{r} \equiv &\\ &(\mathsf{let}_{(A_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{r})\\ &+ (\mathsf{let}_{(A_1,B_2)}\ (x_1, x_2) = \vec{s}\ \mathsf{in}\ \vec{r})\end{aligned}$$

$$\mathsf{case}\ \alpha\vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\} \equiv \alpha(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})$$

$$\begin{aligned}\mathsf{case}\ (\vec{t} + \vec{s})\ \mathsf{of}\{\vec{v} \mapsto \vec{r_1} \mid \cdots \mid \vec{w} \mapsto \vec{r_2}\} \equiv &\\ &\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v} \mapsto \vec{r_1} \mid \cdots \mid \vec{w} \mapsto \vec{r_2}\}\\ &+ \mathsf{case}\ \vec{s}\ \mathsf{of}\{\vec{v} \mapsto \vec{r_1} \mid \cdots \mid \vec{w} \mapsto \vec{r_2}\}\end{aligned}$$

Table 3.3: *Term congruence*

$$= \frac{1}{\sqrt{2}}\left(\frac{(1,1)}{\sqrt{2}} + \frac{(1,-1)}{\sqrt{2}}\right)$$

If we match the vector $(1,0)$ to $|0\rangle$ and $(0,1)$ to $|1\rangle$, we would need a way to introduce the second coordinate into the equation. That is where the first rule comes into play. We restrict ourselves to vectors, since introducing variables or abstractions could break safety properties of the system.

The core mechanism of the calculus lies in decorating variable bindings with sets of value distributions. Keeping with linear algebra terminology, we will refer to these sets as *(orthonormal) bases*, for reasons which will shortly become clear. These bases will inform the reduction system on how to operate its arguments.

In order to properly characterize the sets that decorate the lambda abstractions, we first have to define which are the values that they must contain.

Definition 1. *A* 1*-dimensional qubit is a value distribution of the form:* $\alpha\,|0\rangle + \beta\,|1\rangle$ *where* $|\alpha|^2 + |\beta|^2 = 1$*. An n-th dimensional qubit is a value distribution of the form* $\alpha(|0\rangle, \vec{w_1}) + \beta(|1\rangle, \vec{w_2})$ *where* $\vec{w_1}$ *and* $\vec{w_2}$ *are* $(n-1)$ *dimensional qubits and the same previous conditions apply to* α *and* β*.*

From this point forward we shall write $\vec{\mathrm{V}}$ to the space of all closed value distributions which we will call *vectors*. This space can be equipped with an inner product $\langle \vec{v} \mid \vec{w} \rangle$ and an $\ell_2 - norm$ $\|\vec{v}\|$ defined as:

$$\begin{aligned} \langle \vec{v} \mid \vec{w} \rangle &:= \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} \overline{\alpha_i}\, \beta_j\, \delta_{v_i, w_j} \\ \|\vec{v}\,\| &:= \sqrt{\langle \vec{v} \mid \vec{v}\, \rangle} = \sqrt{\textstyle\sum_{i=1}^{n} |\alpha_i|^2} \end{aligned}$$

Where $\vec{v} = \sum_{i=1}^{n} \alpha_i\, v_i$ and $\vec{w} = \sum_{j=1}^{m} \beta_j\, w_j$, and where δ_{v_i, w_j} is the Kronecker delta such that it is 1 if $v_i = w_j$ and 0 otherwise.

With the notion of an internal product, we can finalize the details on the calculus syntax. As one might expect, we will say two values are orthogonal when their internal product equals to zero. With the previous definition we can describe the sets decorating the abstractions.

Definition 2. *We will say a set of value distributions B is an n-th dimensional orthonormal basis when it satisfies the following conditions:*

1. *B has cardinality n.*
2. *Each member of B is a qubit of dimension n.*
3. *Each member has norm equal to* 1*.*
4. *Each member of B is pairwise orthogonal to every other member.*

Unlike the usual definition of orthonormal basis, we also need to ensure that the members are qubits. In other words, they are neither variables nor abstractions. Morally, these sets will keep track of the basis the term is working on. A qubit which is a member of this set will be treated on a call-by-value strategy and its data can be treated classically. Any other qubit will first be interpreted as a $\mathbb{C}$-linear combination of elements of the basis and then the function will apply linearly to each component. If the argument cannot be written in the decorating basis, the evaluation gets stuck.

As one would expect from a basis in lineal algebra, there is no non-trivial linear combination of its members that yields a null vector. Otherwise, there would be a basis vector which breaks the pairwise orthogonality condition. This implies that every decomposition onto a base is unique.

Proposition 1. *If B is an n-th dimensional basis, then each n-th dimensional qubit has a unique decomposition in B.*

Proof. Let $\vec{b_i}$, the basis vectors of B. And $\sum_{i}^{n} \alpha_i \vec{b_i}$, $\sum_{i=1}^{n} \beta_i \vec{b_i}$ two decompositions of $\vec{v}$ onto B. Then we have:

$$\vec{v} - \vec{v} = 0\; \vec{v} = \sum_{i=1}^{n} (\alpha_i - \beta_i) \vec{b_i}$$

Since the basis is linearly independent, $\alpha_i = \beta_i$. □

As a corollary, we can show that this result behaves well with the term congruence.

Corollary 1. *If $\vec{v} \equiv \vec{w}$, then they both have the same decomposition over a basis B.*

Proof. Since $\vec{v} - \vec{w} \equiv \vec{v} - \vec{v} \equiv \vec{w} - \vec{w}$, we can use the same reasoning as proposition 1 to conclude that both have the same decomposition. □

3.2.2 Substitutions

The beta reduction will depend on the basis chosen for the abstraction, so we have to define a new substitution which will take this mechanism into account. This operation will substitute the variables for vectors in the chosen basis. The accompanying coefficients correspond to the value distribution which is the object of the substitution.

With this substitution we also define a special kind of basis which we call $\mathcal{P}$ which will act as the canonical basis for lambda abstractions. In this way, we restrict distributions of functions to a single possible basis.

Definition 3. *For a term distribution $\vec{t}$, value distribution $\vec{v}$, variable x and orthogonal basis B, we define the substitution $\vec{t}\langle\vec{v}/x\rangle_A$ as:*

$$\vec{t}\langle\vec{v}/x\rangle_B = \begin{cases} \sum_{i\in I} \alpha_i \vec{t}\ [\vec{b_i}/x] & B = \{\vec{b_i}\}_{i\in I} \wedge \vec{v} \equiv \sum\limits_{i\in I} \alpha_i \vec{b_i} \\ \sum_{i\in I} \alpha_i \vec{t}\ [v_i/x] & B = \mathcal{P} \wedge \vec{v} = \sum\limits_{i\in I} \alpha_i v_i \\ \textit{Undefined} & \textit{Otherwise} \end{cases}$$

The difference between the first two cases is subtle. While the first case substitutes linearly with the decomposition onto the basis B. The second, substitutes linearly over the pure values that conform $\vec{v}$ when $B = \mathcal{P}$. In this manner, this modality recovers the substitution originally described in [13]. This case is important because it is the only way to substitute with a λ-abstraction since they cannot form part of orthonormal bases.

This definition can also be extended to a pair of values in the following way. Let $\vec{v} = \sum_{i\in I} \alpha_i(\vec{v_i}, \vec{w_i})$:

$$\vec{t}\langle\vec{v}/x \otimes y\rangle_{B_1\otimes B_2} = \sum_{i\in I} \alpha_i \vec{t}\langle\vec{v_i}/x\rangle_{B_1}\langle\vec{w_i}/y\rangle_{B_2}$$

Example 3.1. *From here onwards, we define $\mathbb{B} = \{\left|0\right\rangle, \left|1\right\rangle\}$. This basis represents the classical boolean bits. Let:*

$$\vec{v} = \frac{1}{\sqrt{2}}(\left|0\right\rangle, \left|1\right\rangle) - \frac{1}{\sqrt{2}}(\left|1\right\rangle, \left|0\right\rangle)$$

Then the substitution $\vec{t}\langle\vec{v}/x \otimes y\rangle_{\mathbb{B}\otimes\mathbb{B}}$ yields:

$$\vec{t}\langle\vec{v}/x{\otimes}y\rangle_{\mathbb{B}\otimes\mathbb{B}} = \\ \frac{1}{\sqrt{2}}\,\vec{t}\,\langle\left|0\right\rangle/x\rangle_{\mathbb{B}}\langle\left|1\right\rangle/y\rangle_{\mathbb{B}} - \frac{1}{\sqrt{2}}\,\vec{t}\,\langle\left|1\right\rangle/x\rangle_{\mathbb{B}}\langle\left|0\right\rangle/y\rangle_{\mathbb{B}}$$

With this new substitution defined, we set out to prove some lemmas which will be useful later for proving the validity of some typing judgements. First, we want to show that the basis dependent substitution commutes with the linear combination of terms.

Lemma 8. *For term distributions $\vec{t_i}$, value distribution $\vec{v}$, variable x, $\alpha_i \in \mathbb{C}$ and basis B such that $\langle\vec{v}/x\rangle_B$ is defined:*

$$(\sum_i \alpha_i \vec{t_i})\langle\vec{v}/x\rangle_B \equiv \sum_i \alpha_i \vec{t_i}\langle\vec{v}/x\rangle_B$$

Proof. Let $B \neq \mathcal{P}$ and $\vec{v} \equiv \sum_{j=0}^{n} \beta_j \vec{b_j}$ with each $\vec{b_j} \in B$

$$\begin{aligned}(\sum_i \alpha_i \vec{t})\langle \vec{v}/x \rangle_B &= \sum_{j=1}^{m} \beta_j (\sum_{i=1}^{n} \alpha_i t_i)[\vec{b}_j/x] \\ &\equiv \sum_{i=1}^{n} \alpha_i (\sum_{j=1}^{m} \beta_j t_i [\vec{b}_j/x]) \\ &= \sum_{i=1}^{n} \alpha_i \vec{t_i} \langle \vec{v}/x \rangle_B\end{aligned}$$

The case where $B = \mathcal{P}$ is similar. □

The next thing we need to show is that the substitution behaves well with respect to the term congruence previously defined. In essence, the following results states that for each member of the same equivalence class defined by $\equiv$, the result of substitution for those vectors is always syntactically the same.

Lemma 9. *For value distributions $\vec{v}, \vec{w}$, term distribution $\vec{t}$ and a orthonormal basis B such that $\langle \vec{v}/x \rangle_B$ and $\langle \vec{w}/x \rangle_B$ are defined. If $\vec{v} \equiv \vec{w}$, then $\vec{t}\langle \vec{v}/x \rangle_B = \vec{t}\langle \vec{w}/x \rangle_B$.*

Proof. Since $\vec{v} \equiv \vec{w}$, by Corollary 1, we have that both $\vec{v}$ and $\vec{w}$ can be written as:

$$\vec{v} \equiv \vec{w} \equiv \sum_{i=1}^{n} \alpha_i \vec{b_i} \qquad \text{Where } \vec{b_i} \in B$$

Then:

$$\vec{t}\langle \vec{v}/x \rangle_B = \sum_{i=1}^{n} \alpha_i \vec{t}\,[\vec{b_i}/x] = \vec{t}\langle \vec{w}/x \rangle_B$$

□

Remark 2. *The result from lemma 9, does not translate across bases, so $\vec{t}\langle \vec{v}/x \rangle_A \not\equiv \vec{t}\langle \vec{v}/x \rangle_B$. From here onwards we define $|+\rangle := \frac{|0\rangle + |1\rangle}{\sqrt{2}}$ and $|-\rangle := \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. As well, we note $\mathbb{X} = \{|+\rangle, |-\rangle\}$. With this in mind we have:*

$$\begin{aligned}(\lambda x_C \,.\, y)\langle |+\rangle /y\rangle_{\mathbb{X}} = (\lambda x_C \,.\, |+\rangle) \not\equiv \\ \frac{1}{\sqrt{2}}((\lambda x_C \,.\, |0\rangle) + (\lambda x_C \,.\, |1\rangle)) = (\lambda x_C \,.\, y)\langle |+\rangle /y\rangle_{\mathbb{B}}\end{aligned}$$

This boils down to the fact that the $\equiv$-relation does not commute, neither with the lambda abstraction nor the case construct. This is due to the fact that, despite being computationally equivalent, the terms $\lambda x_B \,.\, \sum_{i=1}^{n} \alpha_i \vec{t_i}$ and $\sum_{i=1}^{n} \alpha_i \lambda x_B \,.\, \vec{t_i}$ are not congruent (Similarly for the case construct)

This design choice comes from a physical interpretation. If we think of $(\lambda x_B \,.\, \alpha\, \vec{v_1} + \beta\, \vec{v_2})$ as an experiment that produces the superposition of the states represented by $\vec{v_1}$ and $\vec{v_2}$. We would like to differentiate it from the superposition of experiments $\alpha\ (\lambda x_B \,.\, \vec{v_1}) + \beta\ (\lambda x_B \,.\, \vec{v_2})$.

We now introduce notation for generalized substitutions over a term. A substitution σ can be thought as a set of singular substitutions applied consecutively over a term. More precisely, for a term $\vec{t}$, value distributions $\vec{v_1} \cdots \vec{v_n}$, variables $x_1, \cdots, x_n$ and, bases $B_1, \cdots B_n$:

$$\vec{t}\langle\sigma\rangle := \vec{t}\langle\vec{v_1}/x_1\rangle_{B_1} \cdots \langle\vec{v_n}/x_n\rangle_{B_n}$$

Since every $\vec{v_1}, \cdots, \vec{v_n}$ is closed, the order of the substitutions is irrelevant. We can think of the substitution σ as a partial function from variables to pairs of value distributions and bases. We denote $x_1, \cdots, x_n$ as the domain of σ (Noted $\mathrm{dom}(\sigma)$). In the same way, we can extend the substitution, for a term $\vec{t}$, substitution σ, value distribution $\vec{v}$, variable $x \notin \mathrm{dom}(\sigma)$ and basis B:

$$\vec{t}\langle\sigma\rangle\langle\vec{v}/x\rangle_B = \vec{t}\langle\sigma'\rangle$$

Such that σ' behaves the same as σ and, it maps x to $\vec{v}$ in the basis B. This operation can also extend different generalized substitutions, σ_1, σ_2 with $\mathrm{dom}(\sigma_1) \cap \mathrm{dom}(\sigma_2) = \emptyset$:

$$\vec{v}\langle\sigma'\rangle = \vec{t}\langle\sigma_1\rangle\langle\sigma_2\rangle$$

Such that behaves as either σ_1 or σ_2 for variables in their respective domains.

3.3 Reduction system

The reduction system implements a mechanism where every vector in the space is read in the corresponding basis attached to the abstraction. It does this by allowing an evaluation step only when the argument can be decomposed onto that basis. The system works modulo the congruence defined in Table 3.3. We describe it in detail in Table 3.4.

The three main rules are the β-reduction, let-destructor and case pattern matching. The λ abstraction and let construct both attach an orthonormal basis to the variables they are binding. These bases keep track of which vectors it considers as classical data. Any $\mathbb{C}$-combination of them will be treated as quantum data, meaning, linearly.

The only exception is in the case of higher order reductions. Since we do not have defined orthogonal bases for programs, we introduce a special basis $\mathcal{P}$ which acts as the traditional computational basis. We can think of it as being composed of every single pure value. For example:

$$\sum_{i=1}^{n} \alpha_i(\lambda x_{\mathcal{P}} \,.\, \vec{t_i}) \sum_{j=1}^{m} \beta_j(\lambda y_{B_X} \,.\, \vec{s_j}) \to$$
$$\sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i\beta_j \vec{t_i}[(\lambda y_{B_X} \,.\, \vec{s_j})/x]$$

The case pattern matching controls the flow of programs. It generalizes the $\mathsf{if} - \mathsf{then} - \mathsf{else}$ branching. However, we do not consider fixed true or false values. Each operator will keep track of a set of orthogonal values. Then it will test the argument for equality against each vector and choose the matching branch. If the argument is a linear combination of several vectors, the result will be the corresponding linear combination of branches. For example:

$$\mathsf{case}\ \ |-\rangle\ \ \mathsf{of}\{|0\rangle \mapsto \vec{t_1} \mid |1\rangle \mapsto \vec{t_2}\} \to \frac{1}{\sqrt{2}} \cdot \vec{t_1} - \frac{1}{\sqrt{2}} \cdot \vec{t_2}$$

The advantage of this general approach over a binary conditional is the possibility to match against several vectors simultaneously. For boolean tuples, it makes no difference

If $\vec{t_i}\langle\vec{v}/x\rangle_A$ is defined:

$$\sum_{i=1}^{n}\alpha_i(\lambda x_A\,.\,\vec{t_i})\ \vec{v}\to\sum_{i=1}^{n}\alpha_i\vec{t_i}\langle\vec{v}/x\rangle_A$$

$$\mathsf{let}_{(B,B')}\ (x,y)=\vec{v}\ \mathsf{in}\ \vec{t}\to\vec{t}\langle\vec{v}/x\otimes y\rangle_{B\otimes B'}$$

If $\vec{v}\equiv\sum_{i=1}^{n}\alpha_i\vec{v_i}$:

$$\mathsf{case}\ \vec{v}\ \mathsf{of}\{\vec{v_1}\mapsto\vec{t_1}\mid\cdots\mid\vec{v_n}\mapsto\vec{t_2}\}\to\sum_{i=1}^{n}\alpha_i\vec{t_i}$$

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

$$\frac{t\longrightarrow\vec{r}}{\mathsf{let}_{(A,B)}\ (x,y)=t\ \mathsf{in}\ \vec{s}\longrightarrow\mathsf{let}_{(A,B)}\ (x,y)=\vec{r}\ \mathsf{in}\ \vec{s}}$$

$$\frac{t\longrightarrow\vec{r}}{\begin{array}{l}\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v}\mapsto\vec{s}_1\mid\cdots\mid\vec{w}\mapsto\vec{s}_2\}\longrightarrow\\ \qquad\mathsf{case}\ \vec{r}\ \mathsf{of}\{\vec{v}\mapsto\vec{s}_1\mid\cdots\mid\vec{w}\mapsto\vec{s}_2\}\end{array}}$$

Table 3.4: *Reduction system*

since we can treat each component independently. However, there are orthogonal bases which cannot be written as the product of two smaller bases themselves. In this case, the general case allows us match against these vectors. For example:

$$\begin{aligned}\mathsf{case}\ \vec{v}\ \mathsf{of}\ \{&\frac{|00\rangle+|11\rangle}{2}\mapsto\vec{t_1}\mid\\ &\frac{|00\rangle-|11\rangle}{2}\mapsto\vec{t_2}\mid\\ &\frac{|01\rangle+|10\rangle}{2}\mapsto\vec{t_3}\mid\\ &\frac{|01\rangle-|10\rangle}{2}\mapsto\vec{t_4}\}\end{aligned}$$

This particular set of four vectors is called the *Bell basis.* It is useful in the field of quantum communication. In a later section, we will explore the quantum teleportation algorithm which heavily relies on these states.

Defining the system in this way determines a strategy in the *call-by-value* family, which we dub *call-by-arbitrary-basis.* Note that evaluation is weak, meaning that no reduction occurs under lambda, pairs, let or conditional constructors. This prevents unnecessary work, reducing sub-terms that may or may not be utilized.

The congruence relation on terms gives rise to different redexes. However, we can show that the relation $\equiv$ commutes with the reflexive-transitive closure of the reduction $\to$ (We shall note $\twoheadrightarrow$ as this reflexive-transitive closure). In other words, equivalence is preserved by the reduction $\twoheadrightarrow$.

Theorem 1 (Reduction preserves equivalence)**.** *Let $\vec{t}$ and $\vec{s}$ be closed term distributions with $\vec{t}\equiv\vec{s}$. If $\vec{t}\to\vec{t'}$ and $\vec{s}\to\vec{s'}$, then there exist term distributions $\vec{r_1}$ and $\vec{r_2}$ such that*

$\vec{t'} \twoheadrightarrow \vec{r_1}$, $\vec{s'} \twoheadrightarrow \vec{r_2}$, and $\vec{r_1} \equiv \vec{r_2}$. Diagrammatically:

$$\begin{tikzcd}
 & \vec{t} \arrow[dl] & \equiv & \vec{s} \arrow[dr] & \\
\vec{t'} \arrow[dr, "*"] & & & & \vec{s'} \arrow[dl, "*"'] \\
 & \vec{r_1} & \equiv & \vec{r_2} &
\end{tikzcd}$$

Proof. We do a case-by-case analysis over the relation $\vec{t} \equiv \vec{s}$.

$\vec{t_1} + 0\vec{t_2} \equiv \vec{t_1}$: This case follows from Lemma **??** since the reductions can only be performed in $\vec{t_1}$.

$0\vec{t} \equiv \vec{0}$: The term distributions cannot reduce on either side of the equivalence.

$1\vec{t} \equiv \vec{t}$: This case follows from Lemma **??**.

$\alpha(\beta\vec{t}) \equiv \delta\vec{t}$: This case follows from Lemma **??**.

$\vec{t_1} + \vec{t_2} \equiv \vec{t_2} + \vec{t_1}$: This case follows from Lemma **??**. We just have to match the reductions on both sides of the equivalence.

$\vec{t_1} + (\vec{t_2} + \vec{t_3}) \equiv (\vec{t_1} + \vec{t_2}) + \vec{t_3}$: This case follows from Lemma **??**. We just have to match the reductions on both sides of the equivalence.

$(\alpha + \beta)\vec{t} \equiv \vec{t}$: We start analyzing the coefficients. If $\alpha + \beta = 0$, then there cannot be a reduction on the left hand-side. If $(\alpha + \beta) \neq 0$ and either $\alpha = 0$ or $\beta = 0$, then we are on a particular case of $\vec{t_1} + 0\vec{t_2} \equiv \vec{t_1}$ with $\vec{t_1} = \vec{t_2}$. Otherwise, we match the reductions on both sides of the equivalence with Lemma **??**.

$\alpha(\vec{t_1} + \vec{t_2}) \equiv \alpha\vec{t_1} + \alpha\vec{t_2}$: If $\alpha = 0$, then the term distributions cannot reduce on either side of the equivalence. Otherwise, we match the reductions on both sides of the equivalence with Lemma **??**.

$\vec{t}(\alpha\vec{s}) \equiv \alpha(\vec{t}\vec{s})$: If $\alpha = 0$, then there is no reduction possible on the right-hand side. If there is an internal reduction on either $\vec{s}$ or $\vec{t}$, then we match the reductions on both sides of the equivalence with Lemma **??**.

If $\vec{t} = (\lambda x_B \,.\, \vec{t_1})$, $\vec{s} \equiv \sum_{i=1}^{n} \beta_i \vec{b_i}$ with $\vec{b_i} \in B$ and $\vec{t_1}\langle\vec{v}/x\rangle_B$, is defined then (we consider the case $B \neq \mathcal{P}$):

$$\begin{aligned}
(\lambda x_B \,.\, \vec{t_i})(\alpha\vec{v}) \to{}& \vec{t_1}\langle\alpha\vec{v}/x\rangle_B \\
={}& \sum_{i=1}^{n} \alpha\beta_i \vec{t_1}[\vec{b_i}/x]
\end{aligned}$$

On the other side:

$$\begin{aligned}
\alpha((\lambda x_B \,.\, \vec{t_1})\vec{v}) \to{}& \alpha(\vec{t_1}\langle\vec{v}\rangle_B) \\
={}& \alpha(\sum_{i=1}^{n} \beta_i \vec{t_1}[\vec{b_i}/x])
\end{aligned}$$

And we have that both terms are equivalent. The case for $B = \mathcal{P}$ is similar.

$(\alpha\vec{t})\vec{s} \equiv \alpha(\vec{t}\vec{s})$: If $\alpha = 0$, then there is no reduction possible on the right-hand side. If there is an internal reduction on either $\vec{s}$ or $\vec{t}$, then we match the reductions on both sides of the equivalence with Lemma **??**. There are no other possible redexes since the abstraction must be a pure value to reduce on the left hand-side.

$(\vec{t}+\vec{s})\vec{r} \equiv \vec{t}\vec{s}+\vec{t}\vec{r}$: If there is an internal reduction on either $\vec{t}$,$\vec{s}$ or $\vec{r}$, then we match the reductions on both sides of the equivalence with Lemma **??**. There are no other possible redexes since the abstraction must be a pure value to reduce on the left hand-side.

$\vec{t}(\vec{s}+\vec{r}) \equiv \vec{t}\vec{s}+\vec{t}\vec{r}$: If there is an internal reduction on either $\vec{t}$, $\vec{s}$ or $\vec{r}$, then we match the reductions on both sides of the equivalence with Lemma **??**.

If $\vec{t} = (\lambda x_B \,.\, \vec{t_1})$, $\vec{s} \equiv \sum_{i=1}^{n} \alpha_i \vec{b_i}$, and $\vec{r} \equiv \sum_{i=1}^{n} \beta_i \vec{b_i}$ with $\vec{b_i} \in B$ where $\vec{t_1}\langle \vec{v}/x\rangle_B$ and $\vec{t_1}\langle \vec{w}/x\rangle_B$ are defined then (we consider the case where $B \neq \mathcal{P}$):

$$\begin{aligned}(\lambda x_B \,.\, \vec{t_1})(\vec{v}+\vec{w}) &\to \vec{t_1}\langle \vec{v}+\vec{w}/x\rangle_B\\ &= \sum_{i=1}^{n} (\alpha_i+\beta_i)\vec{t_1}[\vec{b_i}/x]\end{aligned}$$

On the other side:

$$\begin{aligned}(\lambda x_B \,.\, \vec{t_1})\vec{v} + (\lambda x_B \,.\, \vec{t_1})\vec{w} &\to \vec{t_1}\langle \vec{v}/x\rangle_B + (\lambda x_B \,.\, \vec{t_1})\vec{w}\\ &\to \vec{t_1}\langle \vec{v}/x\rangle_B + \vec{t_1}\langle \vec{w}/x\rangle_B\\ &= \sum_{i=1}^{n} \alpha_i \vec{t_1}[\vec{b_i}/x] + \sum_{i=1}^{n} \beta_i \vec{t_1}[\vec{b_i}/x]\end{aligned}$$

And we have that both terms are equivalent. The case for $B = \mathcal{P}$ is similar.

$\mathsf{let}_{(B_1,B_2)}\ (x_1,x_2) = (\alpha\vec{t})\ \mathsf{in}\ \vec{s} \equiv \alpha(\mathsf{let}_{(A,B)}\ (x_1,x_2) = \vec{t}\ \mathsf{in}\ \vec{s})$: If $\alpha = 0$, then there is no reduction possible on the right-hand side. If there is an internal reduction on either $\vec{s}$ or $\vec{t}$, then we match the reductions on both sides of the equivalence with Lemma **??**.

If $\vec{t} \equiv \sum_{i=1}^{n} \beta_i(\vec{v_i},\vec{w_i})$ with $\vec{v_i} \in B_1$, $\vec{w_i} \in B_2$ and $\vec{s}\langle \vec{v}/x_1 \otimes x_2\rangle_{B_1 \otimes B_2}$ is defined, then (we consider $B_1, B_2 \neq \mathcal{P}$):

$$\begin{aligned}\mathsf{let}_{(B_1,B_2)}\ (x_1,x_2) = (\alpha\vec{v})\ \mathsf{in}\ \vec{s} &\to \vec{s}\langle \alpha\vec{v}/x_1 \otimes x_2\rangle_{B_1\otimes B_2}\\ &= \sum_{i=1}^{n} \alpha\beta_i \vec{s}[\vec{v_i}/x_1][\vec{w_i}/x_2]\end{aligned}$$

On the other side;

$$\begin{aligned}\alpha(\mathsf{let}_{(B_1,B_2)}\ (x_1,x_2) = \vec{v}\ \mathsf{in}\ \vec{s}) &\to \alpha(\vec{s}\langle \vec{v}/x_1 \otimes x_2\rangle_{B_1\otimes B_2})\\ &= \alpha\sum_{i=1}^{n} \beta_i \vec{s}[\vec{v_i}/x_1][\vec{w_i}/x_2]\end{aligned}$$

And we have that both terms are equivalent. The case for $B_1, B_2 = \mathcal{P}$ are similar.

$\mathsf{let}_{(B_1,B_2)}\ (x_1,x_2) = \vec{t}+\vec{s}\ \mathsf{in}\ \vec{r} \equiv$
$(\mathsf{let}_{(B_1,B_2)}\ (x_1,x_2) = \vec{t}\ \mathsf{in}\ \vec{r}) + (\mathsf{let}_{(B_2,B_2)}\ (x_1,x_2) = \vec{s}\ \mathsf{in}\ \vec{r})$:

If there is an internal reduction on either $\vec{t}$, $\vec{s}$ or $\vec{r}$, then we match the reductions on both sides of the equivalence with Lemma **??**.

If $\vec{t} \equiv \sum_{i=1}^{n} \alpha_i(\vec{v_i}, \vec{w_i})$, $\vec{s} \equiv \sum_{i=1}^{n} \beta_i(\vec{v_i}, \vec{w_i})$ with $\vec{v_i} \in B_1$, $\vec{w_i} \in B_2$ whera $\vec{r}\langle \vec{v}/x_1 \otimes x_2 \rangle_{B_1 \otimes B_2}$ and $\vec{r}\langle \vec{v}/x_1 \otimes x_2 \rangle_{B_1 \otimes B_2}$ are defined, then (we consider $B_1, B_2 \neq \mathcal{P}$):

$$\begin{aligned} &\mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{v} + \vec{w}\ \mathsf{in}\ \vec{r} \\ &\qquad\qquad \to \vec{r}\langle \vec{v} + \vec{w}/x_1 \otimes x_2 \rangle_{B_1 \otimes B_2} \\ &\qquad\qquad = \sum_{i=1}^{n} (\alpha_i + \beta_i)\vec{r}[\vec{v_i}/x_1][\vec{w_i}/x_2] \end{aligned}$$

On the other side:

$$\begin{aligned} &(\mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{r}) + (\mathsf{let}_{(B_2,B_2)}\ (x_1, x_2) = \vec{s}\ \mathsf{in}\ \vec{r}) \\ &\to \vec{r}\langle \vec{v}/x_1 \otimes x_2 \rangle_{B_1 \otimes B_2} + (\mathsf{let}_{(B_2,B_2)}\ (x_1, x_2) = \vec{s}\ \mathsf{in}\ \vec{r}) \\ &\to \vec{r}\langle \vec{v}/x_1 \otimes x_2 \rangle_{B_1 \otimes B_2} + \vec{r}\langle \vec{w}/x_1 \otimes x_2 \rangle_{B_1 \otimes B_2} \\ &= \sum_{i=1}^{n} \alpha_i \vec{r}[\vec{v_i}/x_1][\vec{w_i}/x_2] + \sum_{i=1}^{n} \beta_i \vec{r}[\vec{v_i}/x_1][\vec{w_i}/x_2] \end{aligned}$$

And we have that both terms are equivalent. The case for $B_1, B_2 = \mathcal{P}$ are similar.

$$\begin{array}{l} \mathsf{case}\ \alpha\vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\} \equiv \\ \alpha(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\}) \end{array} :$$

If $\alpha = 0$, then there is no reduction possible on the right hand-side. If there are internal reductions on $\vec{t}$, then we match on both sides of the equivalence with Lemma **??**.

If $\vec{t} \equiv \sum_{i=1}^{n} \beta_i \vec{v_i}$. Then:

$$\mathsf{case}\ \alpha\vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\} \to \sum_{i=1}^{n} \alpha\beta_i\vec{s_i}$$

On the other side:

$$\alpha(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\}) \to \alpha\sum_{i=1}^{n} \beta_i\vec{s_i}$$

And we have that both terms are equivalent.

$$\begin{array}{l} \mathsf{case}\ (\vec{t} + \vec{s})\ \mathsf{of}\{\vec{v_1} \mapsto \vec{r_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{r_n}\} \equiv \\ \mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{r_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{r_n}\} + \\ \mathsf{case}\ \vec{s}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{r_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{r_n}\} \end{array} :$$

If there is an internal reduction on either $\vec{t}$ or $\vec{s}$, then we match the reductions on both sides of the equivalence with Lemma **??**.

If $\vec{t} \equiv \sum_{i=1}^{n} \alpha_i\vec{v_i}$, and $\vec{s} \equiv \sum_{i=1}^{n} \beta_i\vec{v_i}$. Then:

$$\mathsf{case}\ (\vec{t} + \vec{s})\ \mathsf{of}\{\vec{v_1} \mapsto \vec{r_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{r_n}\} \to \sum_{i=1}^{n} (\alpha_i + \beta_i)\vec{r_i}$$

On the other side:

$$\begin{aligned}
&\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{r_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{r_n}\} + \mathsf{case}\ \vec{s}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{r_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{r_n}\} \\
&\to \sum_{i=1}^{n} \alpha_i + \vec{r_i} + \mathsf{case}\ \vec{s}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{r_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{r_n}\} \\
&\to \sum_{i=1}^{n} \alpha_i \vec{r_i} + \sum_{i=1}^{n} \beta_i \vec{r_i}
\end{aligned}$$

And we have that both terms are equivalent.

□

Convention 1. *With the previous result in mind, we will consider term distributions modulo the $\equiv$ congruence. This will not affect distributions under λ-abstractions or case conditionals which we only consider up to α-conversion. Notice that reduction modulo $\equiv$ is deterministic.*

3.4 Realizability model

In this section, we present the type system corresponding to the untyped language introduced in the previous section, along with its realizability semantics.

3.4.1 Unitary Type Semantics

Given the deterministic machine presented in the previous section, the next step to extract a typing system is to define the sets of values which will characterize its types. In order to achieve this we first need to identify the notion of what exactly constitutes a type.

Our aim is to define types that are exclusively inhabited by values of norm equal to 1. The vectors that we wish to study all fall in the *unit sphere*. We will write $\mathcal{S}_1$ for the set $\mathcal{S}_1 := \{\vec{v} \in \vec{\mathrm{V}} \mid \|\vec{v}\| = 1\}$. This corresponds with the mathematical notion of representing quantum data as unit vectors in a Hilbert space.

Definition 4 (Unitary value distribution)**.** *We say a value distribution $\vec{v}$ is unitary when it has norm equal to* 1*. In other words, when $\vec{v} \in \mathcal{S}_1$.*

Definition 5 (Unitary type)**.** *We define a unitary type (or just type) as a notation A together with a set of unitary value distributions noted $[\![A]\!]$ called the unitary semantics of A.*

We next move onto the type realizers. Since our aim is to extract a quantum lambda calculus, we wish to filter global phases of qubits at this level. Since the global phase of a quantum state has no physical significance, we wish to assign the same types to a term $\vec{t}$ and $e^{i\theta} \cdot \vec{t}$. This idea will guide the definition of type realizers.

Definition 6 (Type realizer)**.** *Given a type A and a term distribution $\vec{t}$, we say that $\vec{t}$ realizes A (noted $\vec{t} \Vdash A$), when there is a value distribution $\vec{v}$ such that:*

- $\vec{t} \twoheadrightarrow e^{i\theta} \cdot \vec{v}$
- $\vec{v} \in [\![A]\!]$

For each type A, we note the set of its realizers as $\{\Vdash A\}$.

$$T := B_X \mid T \to T \mid T \times T \mid \sharp T$$

$$\begin{aligned}
\llbracket B_X \rrbracket &:= X \qquad \text{Where: } X \text{ is an orthonormal basis} \\
\llbracket A \times B \rrbracket &:= \left\{ (\vec{v}, \vec{w}) : \vec{v} \in \llbracket A \rrbracket,\ \vec{w} \in \llbracket B \rrbracket \right\} \\
\llbracket A \Rightarrow B \rrbracket &:= \left\{ \sum_{i=1}^{n} \alpha_i (\lambda x_B \,.\, \vec{t_i}) \in \mathcal{S}_1 : \forall \vec{w} \in \llbracket A \rrbracket, (\sum_{i=1}^{n} \alpha_i \vec{t_i}) \langle \vec{w}/x \rangle_A \Vdash B \right\} \\
\llbracket \sharp A \rrbracket &:= (\llbracket A \rrbracket^{\perp})^{\perp}
\end{aligned}$$

Table 3.5: *Type notations and semantics*

With the notions of unitary types and its realizers we can start defining the specific approach for our previously defined language. We begin with the type grammar defined on Table 3.5 and build a simple algebra from the sets of values we aim to represent. From this point onwards denote by $\mathbb{T}$ the set of all types and by $\mathbb{T}_B$ the set of all bases.

The types B_X act as atomic types. They represent a finite set X of orthogonal vectors forming an orthonormal basis. We can represent boolean values with a basis of size 2, but we are not limited to only one kind since there are infinite bases to choose from.

The type $A \times B$ represents the cartesian product of A and B. However, the syntax grammar only allows for pairs of pure values. So there is a small subtlety on the type depicted in the table. For every $\vec{v} = \sum_{i=1}^{n} \alpha_i v_i \in \llbracket A \rrbracket$ and $\vec{w} = \sum_{j=1}^{m} \beta_j w_j \in \llbracket B \rrbracket$ (With v_i and w_j pure values) when we filter out the notation for pairs, we get:

$$\llbracket A \times B \rrbracket := \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j (v_i, w_j) : \vec{v} \in \llbracket A \rrbracket,\ \vec{w} \in \llbracket B \rrbracket \right\}$$

We stress this fact for rigorousness, but for ease of reading from this point onwards we will make use of the previously defined notation.

The arrow type $A \Rightarrow B$ is composed by the distributions of lambda abstractions that take values from the interpretation of A to realizers of B. The last type $\sharp A$ takes the double orthogonal complement and intersects it with the unit sphere.

The type grammar is standard except for type $\sharp A$. We use it to represent quantum data, i.e. linear resources, so terms of this type will not be able to be erased or duplicated. This can be thought as the opposite of the *bang* (!) modality in linear logic. For a more in-depth analysis, refer to [12].

Intuitively, applying the sharp ($\sharp$) operator to a type A yields the span of the original type (intersected with the unitary sphere). This describes the possible linear combinations of values in the unitary semantics of A. The following proposition proves that characterization:

Proposition 2. *The type interpretation $\llbracket \sharp A \rrbracket$ contains the norm-1 linear combination of values in $\llbracket A \rrbracket$.*

$$\llbracket \sharp A \rrbracket = (\llbracket A \rrbracket^{\perp})^{\perp} = \mathsf{Span}(\llbracket A \rrbracket) \cap \mathcal{S}_1$$

Proof. Proof by double inclusion.

$\mathsf{Span}([\![A]\!]) \cap \mathcal{S}_1 \subseteq ([\![A]\!]^\perp)^\perp$: Let $\vec{v} \in \mathsf{Span}([\![A]\!]) \cap \mathcal{S}_1$. Then $\vec{v}$ is of the form $\sum_{i=1}^n \alpha_i \vec{v_i}$ with $\vec{v_i} \in [\![A]\!]$. Taking $\vec{w} \in [\![A]\!]^\perp$, we examine the inner product:

$$\begin{aligned}\langle \vec{v} \mid \vec{w} \rangle &= \langle \sum_{i=1}^n \alpha_i \vec{v_i} \mid \vec{w} \rangle \\ &= \sum_{i=1}^n \overline{\alpha_i} \langle \vec{v_i} \mid \vec{w} \rangle = 0\end{aligned}$$

Then $\vec{v} \in ([\![A]\!]^\perp)^\perp$.

$([\![A]\!]^\perp)^\perp \subseteq \mathsf{Span}([\![A]\!]) \cap \mathcal{S}_1$: Reasoning by contradiction, we assume that there is a $\vec{v} \in ([\![A]\!]^\perp)^\perp$ such that $v \notin \mathsf{Span}([\![A]\!]) \cap \mathcal{S}_1$. Since $\vec{v} \notin \mathsf{Span}([\![A]\!])$, $\vec{v} = \vec{w_1} + \vec{w_2}$ such that $\vec{w_1} \in \mathsf{Span}[\![A]\!]$ and $\vec{w_2}$ is a non-null vector which cannot be written as a linear combination of elements of $[\![A]\!]$. In other words, $\vec{w_2} \in [\![A]\!]^\perp$. Taking the inner product:

$$\langle \vec{v} \mid \vec{w_2} \rangle = \langle \vec{w_1} + \vec{w_2} \mid \vec{w_2} \rangle = \|\vec{w_2}\| \neq 0$$

Then $\vec{v} \notin ([\![A]\!]^\perp)^\perp$. The contradiction stems from assuming $\vec{v} \notin \mathsf{Span}[\![A]\!] \cap \mathcal{S}_1$. □

The following proposition shows that, as one would expect from the span, multiple applications of the sharp operator does not produce a different result beyond the first one.

Proposition 3. *The $\sharp$ operator is idempotent, that is $[\![\sharp A]\!] = [\![\sharp(\sharp A)]\!]$*

Proof. We want to prove that $((([\![A]\!]^\perp)^\perp)^\perp)^\perp = ([\![A]\!]^\perp)^\perp$. For ease of reading, we will write $A^{\perp^n}$ for n successive applications of the operation $\perp$.

$A \subseteq A^{\perp^2}$: Let $\vec{v} \in A$. Then, for all $\vec{w} \in A^\perp$, $\langle \vec{v} \mid \vec{w} \rangle = 0$. Then $\vec{v} \in A^{\perp^2}$. With this we have $A \subseteq A^{\perp^2}$.

$A^{\perp^3} \subseteq A^\perp$: Let $\vec{u} \in A^{\perp^3}$. Then, for all $\vec{v} \in A^{\perp^2}$, $\langle \vec{u} \mid \vec{v} \rangle = 0$. Since we have shown that $A \subseteq A^{\perp^2}$, we have that for all $\vec{w} \in A$, $\langle \vec{u} \mid \vec{w} \rangle = 0$. Then $\vec{u} \in A^\perp$. With this we have $A^{\perp^3} \subseteq A^\perp$.

With these two inclusions we have that $A^\perp = A^{\perp^3}$. So we conclude that: $[\![\sharp(\sharp A)]\!] = A^{\perp^4} = A^{\perp^2} = [\![\sharp A]\!]$ □

Remark 3. *A basis type B_X may be formed by value distributions of pairs and so might be written as the product type of smaller bases. For example, let $X = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, then $B_X = \mathbb{B} \times \mathbb{B}$. However, for the case of entangled bases this cannot be done. A clear example is the Bell basis:* $\mathsf{Bell} = \{\frac{|00\rangle+|11\rangle}{\sqrt{2}}, \frac{|00\rangle-|11\rangle}{\sqrt{2}}, \frac{|01\rangle+|10\rangle}{\sqrt{2}}, \frac{|01\rangle-|10\rangle}{\sqrt{2}}\}$.

The only thing left would be to check that our type algebra captures sets of value distributions we wish to study. Proposition 4 states that every member of a type interpretation has norm 1.

Proposition 4. *For every type A, $[\![A]\!] \subseteq \mathcal{S}_1$.*

Proof. Proof by induction on the shape of A. Since by definition, $[\![B_X]\!]$, $[\![A \Rightarrow B]\!]$ and $[\![\sharp A]\!]$ are built from values in $\mathcal{S}_1$ the only case we need to examine is $[\![A \times B]\!]$.

Let $\vec{v} = \sum_{i=0}^n \alpha_i v_i \in [\![A]\!]$ and $\vec{w} = \sum_{j=0}^m \beta_j w_j$ where every v_i are pairwise orthogonal, same for w_j. Then:

$$(\vec{v}, \vec{w}) = \sum_{i=0}^n \sum_{j=0}^m \alpha_i \beta_j (v_i, w_j)$$

So we have:

$$\|(\vec{v}, \vec{w})\| = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |\alpha_i \beta_j|^2} = \sqrt{\sum_{i=1}^{n} |\alpha_i|^2 \sum_{j=1}^{m} |\beta_j|^2}$$

Since both $\vec{v} \in [\![A]\!]$ and $\vec{w} \in [\![B]\!]$, by inductive hypothesis, we have that $\|\vec{v}\| = \|\vec{w}\| = 1$. Which is to say $\sum_{i=1}^{n} |\alpha_i|^2 = \sum_{j=1}^{m} |\beta_j| = 1$. So we conclude $\|(\vec{v}, \vec{w})\| = 1$.

□

Defining types as sets of values also induces an intuitive way to define a subtyping relationship. We say a type A is subtype of a type B (noted $A \leq B$) if the set of realizers of A is included in the set of realizers of B ($\{\Vdash A\} \subseteq \{\Vdash B\}$). If the sets coincide, we say that A is isomorphic to B (noted $A \cong B$).

Example 3.2. *For example, for every type A, $A \leq \sharp A$. For bases, $B_{\mathbb{B}}$ and $B_{\mathbb{X}}$ we have that: neither $B_{\mathbb{B}} \not\leq B_{\mathbb{X}}$, nor $B_{\mathbb{B}} \not\leq B_{\mathbb{X}}$. However, $\sharp B_{\mathbb{B}} \cong \sharp B_{\mathbb{X}}$.*

Although every type is defined by norm 1 value distributions, not every norm 1 distribution belongs to the interpretation of a type. Take for example the distribution $\frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} (|0\rangle, |0\rangle)$. Another case is a linear combination of abstractions with different bases. For example, the term:

$$\frac{1}{\sqrt{2}} (\lambda x_{\mathbb{B}} . \mathsf{NOT}\ x) + \frac{1}{\sqrt{2}} (\lambda x_{\mathbb{X}} . x)$$

Is not a member of an arrow type, since the bases decorating each abstraction do not match. However, it is computationally equivalent to the abstraction $(\lambda x_{\mathbb{B}} . |+\rangle)$ which belongs to the set $[\![B_{\mathbb{B}} \Rightarrow B_{\mathbb{X}}]\!]$.

3.4.2 Characterization of unitary operators

One of the main results of [13], is the characterization of $\mathbb{C}^2 \to \mathbb{C}^2$ unitary operators using values in $[\![\sharp\mathbb{B} \Rightarrow \sharp\mathbb{B}]\!]$ [13, Theorem IV.12]. In this subsection we expand on this result. Our goal is to prove that abstractions of type $\sharp B_X \Rightarrow \sharp B_Y$ (both bases of size n) represent $\mathbb{C}^n \to \mathbb{C}^n$ unitary operators.

Unitary operators are the isomorphisms of Hilbert spaces since they preserve the basic structure of the space. With this in mind, the first step is to show that the members in $\sharp B_X \Rightarrow \sharp B_Y$ send basis vectors from B_X onto orthogonal vectors in $[\![\sharp B_Y]\!]$. In other words, these abstractions preserve both norm and orthogonality.

Lemma 10. *Given types B_X, B_Y of size n and a closed λ-abstraction $\lambda x_X . \vec{t}$ we have that $\lambda x_A . \vec{t} \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$ if and only if there are value distributions $\vec{w_i} \in [\![\sharp B_Y]\!]$ such that $\forall \vec{v_i} \in [\![B_X]\!]$:*

$$\vec{t}[\vec{v_i}/x] \twoheadrightarrow \vec{w_i} \perp \vec{w_j} \twoheadleftarrow \vec{t}[\vec{v_j}/x] \qquad \text{if } i \neq j$$

Proof. The condition is necessary: Suppose that $\lambda x_X . \vec{t_k} \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$, thus $\forall \vec{v_i} \in [\![\sharp B_X]\!]$, $\vec{t}\langle \vec{v_i}/x \rangle_X \twoheadrightarrow \vec{w_i} \in [\![\sharp B_Y]\!]$. It remains to be seen that $\vec{w_i} \perp \vec{w_j}$ if $i \neq j$. For that, we consider $\alpha_i \in \mathbb{C}$ such that $\sum_{i=1}^{n} |\alpha_i|^2 = 1$. By linear application on the basis X we observe that:

$$(\lambda x_X . \vec{t})(\sum_{i=1}^{n} \alpha_i \vec{v_i}) \to \vec{t}\langle \sum_{i=1}^{n} \alpha_i \vec{v_i}/x \rangle_X$$

$$\begin{aligned}
&= \sum_{i=1}^{n} \alpha_i \vec{t}[\vec{v_i}/x] \\
&\twoheadrightarrow \sum_{i=1}^{n} \alpha_i \vec{w_i}
\end{aligned}$$

But since $\sum_{i=1}^{n} \alpha_i \vec{v_i} \in [\![\sharp A]\!]$, then $\sum_{i=1}^{n} \alpha_i \vec{w_i} \in [\![\sharp B]\!]$ too. Which implies $\| \sum_{i=1}^{n} \alpha_i \vec{w_i} \| = 1$. Therefore:

$$\begin{aligned}
1 = \| \sum_{i=1}^{n} \alpha_i \vec{w_i} \| &= \langle \sum_{i=1}^{n} \alpha_i \vec{w_i} \mid \sum_{j=1}^{n} \alpha_j \vec{w_j} \rangle \\
&= \sum_{i=1}^{n} |\alpha_i|^2 \langle \vec{w_i} \mid \vec{w_i} \rangle + \sum_{i,j=1; i \neq j}^{n} \bar{\alpha}_i \alpha_j \langle \vec{w_i} \mid \vec{w_j} \rangle \\
&= \sum_{i=1}^{n} |\alpha_i|^2 \langle \vec{w_i} \mid \vec{w_i} \rangle + \sum_{i,j=1; i<j}^{n} 2\ \mathsf{Re}(\bar{\alpha}_\mathsf{i} \alpha_\mathsf{j} \langle \tilde{\mathsf{w}}_\mathsf{i} \mid \tilde{\mathsf{w}}_\mathsf{j} \rangle) \\
&= \sum_{i=1}^{n} |\alpha_i|^2 \| \vec{w_i} \|^2 + 2 \sum_{i,j=1; i<j}^{n} \mathsf{Re}(\bar{\alpha}_\mathsf{i} \alpha_\mathsf{j} \langle \tilde{\mathsf{w}}_\mathsf{i} \mid \tilde{\mathsf{w}}_\mathsf{j} \rangle) \\
&= \sum_{i=1}^{n} |\alpha_i|^2 + 2 \sum_{i,j=1; i<j}^{n} \mathsf{Re}(\bar{\alpha}_\mathsf{i} \alpha_\mathsf{j} \langle \tilde{\mathsf{w}}_\mathsf{i} \mid \tilde{\mathsf{w}}_\mathsf{j} \rangle) \\
&= 1 + 2 \sum_{i,j=1; i<j}^{n} \mathsf{Re}(\bar{\alpha}_\mathsf{i} \alpha_\mathsf{j} \langle \tilde{\mathsf{w}}_\mathsf{i} \mid \tilde{\mathsf{w}}_\mathsf{j} \rangle)
\end{aligned}$$

And thus we are left with $\sum_{i,j=1;i<j}^{n} \mathsf{Re}(\bar{\alpha}_\mathsf{i} \alpha_\mathsf{j} \langle \tilde{\mathsf{w}}_\mathsf{i} \mid \tilde{\mathsf{w}}_\mathsf{j} \rangle) = 0$. Taking $\alpha_{i'} = \alpha_{j'} = \frac{1}{\sqrt{2}}$ with 0 for the rest of coefficients, we have $\mathsf{Re}(\langle \tilde{\mathsf{w}}_{\mathsf{i}'} \mid \tilde{\mathsf{w}}_{\mathsf{j}'} \rangle) = 0$ for any two arbitrary i' and j'. In the same way, taking $\alpha_{i'} = \frac{1}{\sqrt{2}}$ and $\alpha_{j'} = \frac{i}{\sqrt{2}}$ with 0 for the rest of the coefficients, we have $\mathsf{Im}(\langle \tilde{\mathsf{w}}_{\mathsf{i}'} \mid \tilde{\mathsf{w}}_{\mathsf{j}'} \rangle) = 0$ for any two arbitrary i' and j'. Finally, we can conclude that $\langle \vec{w_i} \mid \vec{w_j} \rangle = 0$ if $i \neq j$.

The condition is sufficient: Suppose that there are $\vec{w_i} \in [\![\sharp B_Y]\!]$ such that for every $\vec{v_i} \in [\![B_X]\!]$:

$$\vec{t}[\vec{v_i}/x] \twoheadrightarrow \vec{w_i} \perp \vec{w_j} \twoheadleftarrow \vec{t}[\vec{v_j}/x] \qquad \text{If } i \neq j$$

Given any $\vec{u} \in [\![\sharp B_X]\!]$ we have that $\vec{u} = \sum_{i=1}^{n} \alpha_i \vec{v_i}$ with $\sum_{i=1}^{n} |\alpha_i|^2 = 1$ and $\vec{v_i} \in [\![B_X]\!]$. Then

$$(\lambda x_X \,.\, \vec{t})\vec{u} \to \vec{t_k} \langle \vec{u}/x \rangle_X = \sum_{i=1}^{n} \alpha_i \vec{t}[\vec{v_i}/x] \twoheadrightarrow \sum_{i=1}^{n} \alpha_i \vec{w_i}$$

We have that for each i, $\vec{w_i} \in [\![\sharp B_Y]\!]$. In order to show that $(\lambda x_A \,.\, \vec{t})\vec{u} \Vdash \sharp B_Y$ we still have to prove that $\| \sum_{i=1}^{n} \alpha_i \vec{w_i} \| = 1$

$$\| \sum_{i=1}^{n} \alpha_i \vec{w_i} \|^2 = \langle \sum_{i=1}^{n} \alpha_i \vec{w_i} \mid \sum_{j=1}^{n} \alpha_j \vec{w_j} \rangle$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} |\alpha_i|^2 \langle \vec{w_i} \mid \vec{w_i} \rangle + \sum_{i,j=1; i\neq j}^{n} \bar{\alpha_i}\alpha_j \langle \vec{w_i} \mid \vec{w_j} \rangle \\
&= \sum_{i=1}^{n} |\alpha_i|^2 + 0 \\
&= 1
\end{aligned}
$$

Then $\sum_{i=1}^{n} \alpha_i \vec{w_i} \in [\![\sharp(\sharp B_Y)]\!] = [\![\sharp B_Y]\!]$ by Lemma 3. Since for every $\vec{u} \in [\![\sharp A]\!]$, $(\lambda x_A \,.\, \vec{t})\vec{u} \Vdash \sharp B$, we can conclude that $\lambda x_A \,.\, \vec{t} \in [\![\sharp A \to \sharp B]\!]$. □

Next, we need to bridge the gap between the values in the calculus with vectors in the space $\mathbb{C}^n$. In order to do this, we introduce a meta-language operation π_n which translates value distributions into vectors in $\mathbb{C}^n$. The operation simply writes the value in the canonical basis and takes the corresponding coefficients.

Definition 7. *Let B_X be an orthonormal basis of size n, then for every $\vec{v} \in [\![B_X]\!]$:*

$$\vec{v} \equiv \sum_{i=1}^{n} \alpha_i \,|i\rangle$$

Where $|i\rangle$ is the n-th dimensional product of $|0\rangle$ and $|1\rangle$ with i written in binary and $\sum_{i=1}^{n} |\alpha_i|^2 = 1$. (For example, $|3\rangle$ with $n = 4$ is $|0011\rangle$). We define $\pi_n : [\![B_X]\!] \to \mathbb{C}^n$ as::

$$\pi_n(\vec{v}) = (\alpha_1, \cdots, \alpha_n)$$

We will omit the subscript when it can be deduced from the context.

Definition 8. *We say a λ-abstraction $(\lambda x_X \,.\, \vec{t})$ represents an operator $F : \mathbb{C}^n \to \mathbb{C}^n$ when:*

$$(\lambda x_X \,.\, \vec{t})\vec{v} \twoheadrightarrow \vec{w} \iff F(\pi_n(\vec{v})) = \pi_n(\vec{w})$$

This means, a lambda term represents a function $F : \mathbb{C}^n \to \mathbb{C}^n$ if it encodes the action of F on vectors. This definition, in conjunction with the previous lemma, allow us build a characterization of unitary operators as values in $\sharp B_X \Rightarrow \sharp B_X$.

Theorem 2. *Let B_X, B_Y be orthonormal bases of size n. A closed λ-abstraction $(\lambda x_X \,.\, \vec{t}) \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$ if and only if it represents a unitary operator $F : \mathbb{C}^n \to \mathbb{C}^n$.*

Proof. The condition is necessary: Suppose that $(\lambda x_X \,.\, \vec{t}) \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$, then by Lemma 10 we have that, for every $\vec{v_i} \in [\![B_X]\!]$ there exist $\vec{w_i} \in [\![\sharp B_Y]\!]$ such that $\vec{t}[\vec{v_i}/x] \twoheadrightarrow \vec{w_i}$ and $\vec{w_i} \perp \vec{w_j}$ if $i \neq j$. Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be the operator defined as $F(\pi(\vec{v_i})) = \pi(\vec{w_i})$. From the linear application on X, it is clear that $(\lambda x_X \,.\, \vec{t})$ represents the operator F. Moreover, the operator F is unitary since $\|\pi(\vec{w_i})\|_{\mathbb{C}^n} = \|\pi(\vec{w_j})\|_{\mathbb{C}^n} = 1$ and $\langle \pi(\vec{w_i}) \mid \pi(\vec{w_j}) \rangle_{\mathbb{C}^n} = 0$.

The condition is sufficient: Suppose that $(\lambda x_X \,.\, \vec{t})$ represents a unitary operator $F : \mathbb{C}^n \to \mathbb{C}^n$. From this we deduce that:

$$(\lambda x_X \,.\, \vec{t})\vec{v_i} \twoheadrightarrow \vec{w_i}$$

For some $\vec{v_i} \in [\![B_X]\!]$, $\vec{w_i} \in [\![B_Y]\!]$ such that $F(\pi(\vec{v_i})) = \pi(\vec{w_i})$. Then we have:

$$(\lambda x_X \,.\, \vec{t})\vec{v_i} \to \vec{t}\langle \vec{v_i}/x \rangle_X = \vec{t}[\vec{v_i}/x] \twoheadrightarrow \vec{w_i} \in [\![\sharp B_Y]\!],$$

since $\|\vec{w_i}\| = \|F(\pi(\vec{v_i}))\|_{\mathbb{C}^n} = 1$, we can deduce from Lemma 10, that $(\lambda x_X \,.\, \vec{t}) \in [\![\sharp B_X \Rightarrow \sharp B_Y]\!]$. Then:

$$\langle \vec{w_i} \mid \vec{w_j} \rangle = \langle F(\pi(\vec{v_i})) \mid F(\pi(\vec{v_j})) \rangle_{\mathbb{C}^n} = 0$$

□

These results can be extended to unitary distributions of lambda abstractions, since $(\lambda x_X \,.\, \sum_{i=1}^n \alpha_i \vec{t_i})$ is syntactically different but computationally equivalent to $\sum_{i=1}^n \alpha_i(\lambda x_X \,.\, \vec{t_i})$. Ultimately, we generalized one of the main theorems in [13]. The inclusion of the basis type in our system allow us to reason more easily about the action of the operators and translate the proof onto a more general case.

3.4.3 Typing rules

Our focus in this section is to enumerate and prove the validity of various typing rules. The objective being to extract a reasonable set of rules to constitute a type system. We first need to lay the groundwork to properly define what does it mean for a typing rule to be valid.

Definition 9. *A context (Denoted by capital Greek letters* Γ*,* Δ*) is a mapping* $\Gamma : \mathsf{Var} \to \mathbb{T} \times \mathbb{T}_B$ *assigning a type and basis to each variable in its domain. We note the mapping* $\Gamma(x_i) \mapsto (A_i, B_{X_i})$ *as:*

$$\Gamma = x_{1B_{X_1}} : A_1, \cdots, x_{nB_{X_n}} : A_n$$

As usual with typing judgements, the context will keep track of the type of free variables of a term. However, since the substitution operation depends on a basis we also wish to include that information. This is not strictly necessary, since the basis a variable is interpreted should not impact on the type. For example the result of the substitution:

$$(\lambda x_{\mathbb{B}} \,.\, (x, y))\langle |0\rangle /y\rangle_{B_{\mathbb{B}}} = (\lambda x_{\mathbb{B}} \,.\, (x, |0\rangle))$$

And the substitution:

$$(\lambda x_{\mathbb{B}} \,.\, (x, y))\langle |0\rangle /y\rangle_{B_{\mathbb{X}}} = \frac{1}{\sqrt{2}}((\lambda x_{\mathbb{B}} \,.\, (x, |+\rangle)) + (\lambda x_{\mathbb{B}} \,.\, (x, |-\rangle)))$$

Are not syntactically equivalent, but they are equivalent under elimination contexts. Therefore, since typing via realizability captures computational behaviour, the types will match. We will however keep basis information on the contexts to later simplify our proofs. With this, we can define which substitutions validate a context.

Definition 10. *Given a context* Γ *we call the unitary semantics of* Γ*, noted* $\llbracket \Gamma \rrbracket$*, to the set of substitutions such that:*

$$\begin{aligned} \llbracket \Gamma \rrbracket := \{ \sigma \text{ substitution } \mid \operatorname{dom}(\sigma) = \operatorname{dom}(\Gamma) \text{ and } \forall x_i \in \operatorname{dom}(\Gamma), \\ \Gamma(x_i) = (A_i, B_{X_i}) \Rightarrow \sigma(x_i) = \langle \vec{v_i}/x_i \rangle_{B_{X_i}} \wedge \vec{v_i} \in \llbracket A_i \rrbracket \} \end{aligned}$$

In order for the calculus to be correct we need to ensure that qubits are treated linearly. The first step is to identify which variables in the context represent quantum data, those will be the ones associated with a type of the form $\sharp A$. We call the subset of Γ composed by these variables, its *strict domain.*

Definition 11. *We define the strict domain of a context* Γ*, noted* $\operatorname{dom}^\sharp(\Gamma)$*, as:*

$$\operatorname{dom}^\sharp(\Gamma) := \{ x \in \operatorname{dom}(\Gamma) \mid \llbracket \Gamma(x) \rrbracket = \llbracket \sharp(\Gamma(x)) \rrbracket \}$$

Here we make use of the idempotence of $\sharp$ (Proposition 3) to define the strict domain.

In order for a typing judgement $\Gamma \vdash \vec{t} : A$ to be valid, it needs to comply with two conditions. First, every free variable in the term $\vec{t}$ must be in the domain of the context Γ and every variable in the strict context $\operatorname{dom}^\sharp(\Gamma)$ must appear in the term $\vec{t}$. This ensures there is no erasure of information and every variable is accounted. Linear treatment of quantum data is enforced by the substitution.

Second, every substitution in the unitary semantics of Γ, when applied to the term $\vec{t}$, must yield a term which reduces to a realizer of type A. This condition matches the computational behaviour of the term and context to the type. To put it more precisely:

Definition 12. *We say that a typing judgement $\Gamma \vdash \vec{t} : A$ is valid when:*

- $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma)$
- *For all $\sigma \in [\![\Gamma]\!]$, $\vec{t}\langle\sigma\rangle \Vdash A$*

With this definition in mind, we consider a typing rule to be valid, when starting from valid judgements we reach a valid conclusion. In Table 3.6 we enumerate several of these rules. One important thing to note is that there are infinite valid rules, we limit ourselves to listing a subset which could constitute a reasonable typing system for a typed calculus.

We are also interested on *orthogonal terms*, that is, terms which reduce to orthogonal values. Naturally, unless these terms are closed, we need to take the context into consideration. We define orthogonality judgements in the following manner:

Definition 13. *We say that an orthogonality judgement $\Gamma \vdash (\Delta_1 \vdash \vec{t}) \perp (\Delta_2 \vdash \vec{s}) : A$ is valid when:*

- *The judgement $\Gamma, \Delta_1 \vdash \vec{t} : A$ is valid.*
- *The judgement $\Gamma, \Delta_2 \vdash \vec{s} : A$ is valid.*
- *For every $\sigma \in [\![\Gamma, \Delta_1]\!], \tau \in [\![\Gamma, \Delta_2]\!]$ there are value distributions $\vec{v}, \vec{w}$ such that $\vec{t}\langle\sigma\rangle \twoheadrightarrow \vec{v}, \vec{s}\langle\tau\rangle \twoheadrightarrow \vec{w}$ and $\vec{v} \perp \vec{w}$.*

If both Δ_1 and Δ_2 are empty, we will note the judgement as $\Gamma \vdash \vec{t} \perp \vec{s} : A$. We will be mostly interested in these cases.

The main result of this section, is the proof of validity of each of the rules presented in Table 3.6.

Theorem 3. *The rules in Table 3.6 are valid.*

Proof. For each typing rule in Table 3.6 we have to show the typing judgement is valid starting from the premises:

Axiom It is clear that $\mathrm{dom}^\sharp(x : A) \subseteq \{x\} = \mathrm{dom}(x : A)$. Moreover, given $\sigma \in [\![x_B : A]\!]$, we have $\sigma = \langle\vec{v}/x\rangle_B$ for some $\vec{v} \in [\![A]\!]$. Therefore, $x\langle\sigma\rangle = x\langle\vec{v}\rangle_B = \vec{v} \Vdash A$.

Sub Trivial since $\{\Vdash A\} \subseteq \{\Vdash A'\}$.

UnitLam If the hypothesis is valid, $\mathrm{dom}^\sharp(\Gamma, x_A : A) \subseteq \mathrm{FV}(\sum_{i=1}^n \alpha_i \vec{t_i}) \subseteq \mathrm{dom}(\Gamma, x_A : A)$. It follows that $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\sum_{i=1}^n \alpha_i(\lambda x_A \,.\, \vec{t_i})) \subseteq \mathrm{dom}(\Gamma)$. Given $\sigma \in [\![\Gamma]\!]$, we want to show that $(\sum_{i=1}^n \alpha_i(\lambda x_A \,.\, \vec{t_i}))\langle\sigma\rangle \Vdash A \Rightarrow B$. Let $\vec{v} \in [\![A]\!]$, then:

$$\begin{aligned}
(\sum_{i=1}^n \alpha_i(\lambda x_A \,.\, \vec{t_i}))\langle\sigma\rangle\vec{v} &= (\sum_{j=1}^m \beta_j(\sum_{i=1}^n \alpha_i(\lambda x_A \,.\, \vec{t_i})[\sigma_i]))\vec{v} \\
&= (\sum_{i=1}^n \sum_{j=1}^m \alpha_i\beta_j(\lambda x_A \,.\, \vec{t_i}[\sigma_j]))\vec{v} \\
&\to \sum_{i=1}^n \sum_{j=1}^m \alpha_i\beta_j\vec{t_i}[\sigma_j]\langle\vec{v}/x\rangle_A \\
&= \sum_{i=1}^n \alpha_i\vec{t_i}\langle\sigma\rangle\langle\vec{v}/x\rangle_A \\
&= (\sum_{i=1}^n \alpha_i\vec{t_i})\langle\sigma\rangle\langle\vec{v}/x\rangle_A \qquad \text{By Lemma 8}
\end{aligned}$$

$$\frac{B_X \leq A \vee X = \mathcal{P}}{x_X : A \vdash x : A}\ \text{(Axiom)} \qquad \frac{\Gamma \vdash \vec{t} : A \quad A \leq A'}{\Gamma \vdash \vec{t} : A'}\ \text{(Sub)}$$

$$\frac{\Gamma, x_A : A \vdash \sum_{i=1}^{n} \alpha_i \vec{t_i} : B}{\Gamma \vdash \sum_{i=1}^{n} \alpha_i (\lambda x_A \,.\, \vec{t_i}) : A \Rightarrow B}\ \text{(UnitLam)}$$

$$\frac{\Gamma \vdash \vec{s} : A \Rightarrow B \quad \Delta \vdash \vec{t} : A}{\Gamma, \Delta \vdash \vec{s}\vec{t} : B}\ \text{(App)} \qquad \frac{\Gamma \vdash \vec{t} : A}{\Gamma \vdash e^{i\theta} \cdot \vec{t} : A}\ \text{(GlobalPhase)}$$

$$\frac{\Gamma \vdash \vec{t} : A \quad \Delta \vdash \vec{s} : B}{\Gamma, \Delta \vdash (\vec{t}, \vec{s}) : A \times B}\ \text{(Pair)} \qquad \frac{\Gamma \vdash \vec{t} : B \quad \flat A \quad A \leq B}{\Gamma, x_A : B \vdash \vec{t} : C}\ \text{(Weak)}$$

$$\frac{\Gamma \vdash \vec{t} : A_1 \times A_2 \quad \Delta, x_{B_1} : A_1, y_{B_2} : A_2 \vdash \vec{s} : C}{\Gamma, \Delta \vdash \mathsf{let}_{(B_1, B_2)}\ (x, y) = \vec{t}\ \mathsf{in}\ \vec{s} : C}\ \text{(LetPair)}$$

$$\frac{\Gamma \vdash \vec{t} : \sharp(A_1 \times A_2) \quad \Delta, x_{B_1} : \sharp A_1, y_{B_2} : \sharp A_2 \vdash \vec{s} : C}{\Gamma, \Delta \vdash \mathsf{let}_{(B_1, B_2)}\ (x, y) = \vec{t}\ \mathsf{in}\ \vec{s} : \sharp C}\ \text{(LetTens)}$$

$$\frac{\Gamma \vdash \vec{t} : B_{\{\vec{v_i}\}_{i=1}^{n}} \quad \forall i,\ \Delta \vdash \vec{s_i} : A}{\Gamma, \Delta \vdash \mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\} : A}\ \text{(Case)}$$

$$\frac{\Gamma \vdash \vec{t} : \sharp B_{\{\vec{v_i}\}_{i=1}^{n}} \quad \forall i \neq j,\ \Delta \vdash \vec{s_i} \perp \vec{s_j} : A}{\Gamma, \Delta \vdash \mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\} : \sharp A}\ \text{(UnitCase)}$$

$$\frac{\forall i \neq j,\, \Gamma \vdash \vec{t_i} \perp \vec{t_j} : A \quad \sum_{i=1}^{n} |\alpha_i|^2 = 1}{\Gamma \vdash \sum_{i=1}^{n} \vec{t_i} : \sharp A}\ \text{(Sum)}$$

$$\frac{\Gamma, x_A : A, y_A : A \vdash \vec{t} : B \quad \flat A}{\Gamma, x_A : A \vdash \vec{t}[y := x] : B}\ \text{(Contr)} \qquad \frac{\Gamma \vdash \vec{t} : A \quad \vec{t} \equiv \vec{s}}{\Gamma \vdash \vec{s} : A}\ \text{(Equiv)}$$

Where the property $\flat$ is defined as:

$$\flat X \iff \forall \vec{v}, \vec{w} \in \llbracket X \rrbracket,\ \vec{v} \neq \vec{w} \Rightarrow \langle \vec{v} \mid \vec{w} \rangle = 0$$

Table 3.6: *Some valid typing rules*

Considering that $\langle\sigma\rangle \in [\![\Gamma]\!]$, then we have that $\langle\sigma\rangle\langle\vec{v}/x\rangle_A \in [\![\Gamma, x_A : A]\!]$. Since we assume $\Gamma, x_A : A \vdash \sum_{i=1}^{n} \alpha_i \vec{t_i} : B$, then $\vec{t_i}\langle\sigma\rangle\langle\vec{v}/x\rangle_A \Vdash B$. Finally, we can conclude that the distribution: $\sum_{i=1}^{n} \alpha_i(\lambda x_A \,.\, \vec{t_i}) \in [\![A \Rightarrow B]\!]$.

App If the hypotheses are valid, then:

- $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{s}) \subseteq \mathrm{dom}(\Gamma)$ and $\vec{s}\langle\sigma_\Gamma\rangle \Vdash A \Rightarrow B \;\forall \sigma_\Gamma \in [\![\Gamma]\!]$.
- $\mathrm{dom}^\sharp(\Delta) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Delta)$ and $\vec{t}\langle\sigma_\Delta\rangle \Vdash A \;\forall \sigma_\Delta \in [\![\Delta]\!]$.

From this, we can conclude that $\mathrm{dom}^\sharp(\Gamma, \Delta) \subseteq \mathrm{FV}(\vec{st}) \subseteq \mathrm{dom}(\Gamma, \Delta)$. Given $\sigma \in [\![\Gamma, \Delta]\!]$, we can observe that $\sigma = \sigma_\Gamma, \sigma_\Delta$ for some $\sigma_\Gamma \in [\![\Gamma]\!]$ and $\sigma_\Delta \in [\![\Delta]\!]$. Then we have:

$$\begin{aligned}
(\vec{ts})\langle\sigma\rangle &= (\vec{ts})\langle\sigma_\Gamma\rangle\langle\sigma_\Delta\rangle \\
&= (\sum_{i=i}^{n} \alpha_i(\vec{ts})[\sigma_{\Gamma i}])\langle\sigma_\Delta\rangle \\
&= \sum_{j=1}^{m} \beta_j(\sum_{i=1}^{n} \alpha_i(\vec{ts})[\sigma_{\Gamma i}])[\sigma_{\Delta j}] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i\beta_j \vec{t}[\sigma_{\Gamma i}][\sigma_{\Delta j}]\vec{s}[\sigma_{\Gamma i}][\sigma_{\Delta j}] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i\beta_j \vec{t}[\sigma_{\Gamma i}]\vec{s}[\sigma_{\Delta j}] \\
&\equiv (\sum_{i=1}^{n} \alpha_i \vec{t}[\sigma_{\Gamma i}])(\sum_{j=1}^{m} \beta_j \vec{s}[\sigma_{\Delta j}]) \\
&= \vec{t}\langle\sigma_\Gamma\rangle\vec{s}\langle\sigma_\Delta\rangle \\
&\twoheadrightarrow (e^{i\theta_1}\vec{v})(e^{i\theta_2}\vec{w}) \qquad \text{Where: } \vec{v} \in [\![A \Rightarrow B]\!], \vec{w} \in [\![A]\!] \\
&\equiv e^{i\theta}(\vec{v}\vec{w}) \qquad \text{with: } \theta = \theta_1 + \theta_2 \\
&\to e^{i\theta}\vec{r} \qquad \text{where: } \vec{r} \Vdash B
\end{aligned}$$

Then we can conclude that $(\vec{ts})\langle\sigma\rangle \Vdash B$.

Pair If the hypotheses are valid, then:

- $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{s}) \subseteq \mathrm{dom}(\Gamma)$ and $\vec{s}\langle\sigma_\Gamma\rangle \Vdash A \;\forall \sigma_\Gamma \in [\![\Gamma]\!]$.
- $\mathrm{dom}^\sharp(\Delta) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Delta)$ and $\vec{t}\langle\sigma_\Delta\rangle \Vdash B \;\forall \sigma_\Delta \in [\![\Delta]\!]$.

From this, we can conclude that $\mathrm{dom}^\sharp(\Gamma, \Delta) \subseteq \mathrm{FV}((\vec{s}, \vec{t})) \subseteq \mathrm{dom}(\Gamma, \Delta)$. Given $\sigma \in [\![\Gamma, \Delta]\!]$, we can observe that $\sigma = \sigma_\Gamma, \sigma_\Delta$ for some $\sigma_\Gamma \in [\![\Gamma]\!]$ and $\sigma_\Delta \in [\![\Delta]\!]$. Then we have:

$$\begin{aligned}
(\vec{t}, \vec{s})\langle\sigma\rangle &= (\vec{t}, \vec{s})\langle\sigma_\Gamma\rangle\langle\sigma_\Delta\rangle \\
&= \sum_{j=1}^{m} \beta_j(\sum_{i=1}^{n} \alpha_i(\vec{t}, \vec{s})[\sigma_{\Gamma i}])[\sigma_{\Delta j}]
\end{aligned}$$

$$
\begin{aligned}
&\equiv \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i\beta_j(\vec{t}[\sigma_{\Gamma i}][\sigma_{\Delta j}], \vec{s}[\sigma_{\Gamma i}][\sigma_{\Delta j}]) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i\beta_j(\vec{t}[\sigma_{\Gamma i}], \vec{s}[\sigma_{\Delta j}]) \\
&= (\sum_{i=1}^{n} \alpha_i\vec{t}[\sigma_{\Gamma i}], \sum_{j=1}^{m} \beta_j\vec{s}[\sigma_{\Delta j}]) \\
&= (\vec{t}\langle\sigma_\Gamma\rangle, \vec{s}\langle\sigma_\Delta\rangle) \\
&\twoheadrightarrow (e^{i\theta_1}\cdot\vec{v}, e^{i\theta_2}\cdot\vec{w}) \qquad \text{where: } \vec{v}\in[\![A]\!], \vec{w}\in[\![B]\!] \\
&= e^{i\theta}(\vec{v},\vec{w}) \qquad \text{where: } \vec{v}\in[\![A]\!], \vec{w}\in[\![B]\!]
\end{aligned}
$$

From this we can conclude that $(\vec{t},\vec{s})\langle\sigma\rangle \Vdash A\times B$. Finally, $\Gamma,\Delta\vdash(\vec{t},\vec{s}) : A\times B$

LetPair If the hypotheses are valid, then:

- $\mathrm{dom}^\sharp(\Gamma)\subseteq \mathrm{FV}(\vec{t})\subseteq \mathrm{dom}(\Gamma)$ and $\vec{t}\langle\sigma_\Gamma\rangle \Vdash A\times B\ \forall\sigma_\Gamma\in[\![\Gamma]\!]$
- $\mathrm{dom}^\sharp(\Delta, x_{1B_1} : A_1, x_{2B_2} : A_2)\subseteq \mathrm{FV}(\vec{s})$
- $\mathrm{FV}(\vec{s})\subseteq \mathrm{dom}(\Delta, x_{1B_1} : A_1, x_{2B_2} : A_2)$
- $\vec{s}\langle\sigma_\Delta\rangle \Vdash C\ \forall\sigma_\Delta\in[\![\Delta, x_{1B_1} : A_1, x_{2B_2} : A_2]\!]$

From this, we can conclude that:

- $\mathrm{dom}^\sharp(\Gamma,\Delta)\subseteq \mathrm{FV}(\mathsf{let}_{(B_1,B_2)}\ (x,y)=\vec{s}\ \mathsf{in}\ \vec{t})$
- $\mathrm{FV}(\mathsf{let}_{(B_1,B_2)}\ (x,y)=\vec{s}\ \mathsf{in}\ \vec{t})\subseteq \mathrm{dom}(\Gamma,\Delta)$

Given $\sigma\in[\![\Gamma,\Delta]\!]$, we have that $\langle\sigma\rangle=\langle\sigma_\Gamma\rangle,\langle\sigma_\Delta\rangle$ for some $\sigma_\Gamma\in[\![\Gamma]\!]$ and $\sigma_\Delta\in[\![\Delta]\!]$. Then we have:

$$
\begin{aligned}
&(\mathsf{let}_{(B_1,B_2)}\ (x,y)=\vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma\rangle = \\
&(\mathsf{let}_{(B_1,B_2)}\ (x,y)=\vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma_\Gamma\rangle\langle\sigma_\Delta\rangle \\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} \alpha_i\beta_j(\mathsf{let}_{(B_1,B_2)}\ (x,y)=\vec{t}\ \mathsf{in}\ \vec{s})[\sigma_{\Gamma i}][\sigma_{\Delta j}] \\
&\equiv \mathsf{let}_{(B_1,B_2)}\ (x,y)=\sum_{i=1}^{n}\alpha_i[\sigma_{\Gamma i}]\vec{t}\ \mathsf{in}\ \sum_{j=1}^{m}\beta_j\vec{s}[\sigma_{\Delta j}] \\
&= \mathsf{let}_{(B_1,B_2)}\ (x,y)=\vec{t}\langle\sigma_\Gamma\rangle\ \mathsf{in}\ \vec{s}\langle\sigma_\Delta\rangle \\
&\twoheadrightarrow \mathsf{let}_{(B_1,B_2)}\ (x,y)=e^{i\theta}\cdot(\vec{v},\vec{w})\ \mathsf{in}\ \vec{s}\langle\sigma_\Delta\rangle \\
&\qquad\qquad\qquad \text{Where: } \vec{v}\in[\![A]\!], \vec{w}\in[\![B]\!] \\
&\to e^{i\theta_1}\cdot(\vec{s}\langle\sigma_\Delta\rangle\langle(\vec{v},\vec{w})/x_1\otimes x_2\rangle_{B_1\otimes B_2}) \\
&= e^{i\theta_1}\cdot(\vec{s}\langle\sigma_\Delta\rangle\langle\vec{v}/x_1\rangle_{B_1}\langle\vec{w}/x_2\rangle_{B_2}) \\
&\twoheadrightarrow e^{i\theta_1}\cdot(e^{i\theta_2}\cdot\vec{u}) \qquad \text{where: } \vec{u}\in[\![C]\!] \\
&\equiv e^{i\theta}\cdot\vec{u} \qquad \text{where: } \theta=\theta_1+\theta_2
\end{aligned}
$$

Since $\langle\sigma_\Delta\rangle\langle\vec{v}/x_1\rangle_{B_1}\langle\vec{w}/x_2\rangle_{B_2}\in[\![\Delta, x_{1B_1} : A_1, x_{2B_2} : A_2]\!]$, then we can conclude that $(\mathsf{let}_{(B_1,B_2)}\ (x,y)=\vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma\rangle \Vdash C$.

LetTens If the hypotheses are valid then:

- $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma)$ and $\vec{t}\langle\sigma\rangle \Vdash A \otimes B \; \forall \sigma \in [\![\Gamma]\!]$
- $\mathrm{dom}^\sharp(\Delta, {x_1}_{B_1} : \sharp A_1, {x_2}_{B_2} : \sharp A_2) \subseteq \mathrm{FV}(\vec{s})$
- $\subseteq \mathrm{dom}(\Delta, x_1 : B_1, {x_2}_{B_2} : A_2)$
- $\vec{s}\langle\sigma\rangle \Vdash \sharp C \; \forall \sigma \in [\![\Delta, {x_1}_{B_1} : \sharp A_1, {x_2}_{B_2} : \sharp A_2]\!]$

From this we can conclude that:

- $\mathrm{dom}^\sharp(\Gamma, \Delta) \subseteq \mathrm{FV}(\mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{s})$
- $\mathrm{FV}(\mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{s}) \subseteq \mathrm{dom}(\Gamma, \Delta)$

Given $\sigma \in [\![\Gamma, \Delta]\!]$, we have that $\langle\sigma\rangle = \langle\sigma_\Gamma\rangle, \langle\sigma_\Delta\rangle$ for some $\sigma_\Gamma \in [\![\Gamma]\!]$ and $\sigma_\Delta \in [\![\Delta]\!]$. Using the first hypothesis we have that, $\vec{t}\langle\sigma_\Gamma\rangle \Vdash \sharp(A_1 \times A_2)$, from Proposition 2 we have that:

$$\vec{t}\langle\sigma_\Gamma\rangle \twoheadrightarrow e^{i\theta_1} \cdot \vec{u} = e^{i\theta_1} \cdot (\sum_{k=1}^{l} \gamma_k(\vec{v}_k, \vec{u}_k))$$

With:

- $\sum_{k=1}^{l} |\gamma_k|^2 = 1$
- $\forall k,\ \vec{v}_k \in [\![A_1]\!],\ \vec{u}_k \in [\![A_2]\!]$
- $\forall k \neq l, \langle(\vec{v_k}, \vec{u_k}) \mid (\vec{v_l}, \vec{u_l})\rangle = 0$

Then:

$$\begin{aligned}
&(\mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{s})\langle\sigma\rangle \\
&\quad = \mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{s}\langle\sigma_\Gamma\rangle\langle\sigma_\Delta\rangle \\
&\quad = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{s}\ [\sigma_{\Gamma i}][\sigma_{\Delta j}] \\
&\quad \equiv \mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \sum_{i=1}^{n} \alpha_i \vec{t}\ [\sigma_{\Gamma i}]\ \mathsf{in}\ \sum_{j=1}^{m} \beta_j \vec{s}\ [\sigma_{\Delta j}] \\
&\quad = \mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{t}\langle\sigma_\Gamma\rangle\ \mathsf{in}\ \vec{s}\langle\sigma_\Delta\rangle \\
&\quad \twoheadrightarrow \mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = e^{i\theta_1} \cdot \vec{u}\ \mathsf{in}\ \vec{s}\langle\sigma_\Delta\rangle \\
&\quad \to e^{i\theta_1} \cdot (\vec{s}\langle\sigma_\Delta\rangle\langle\vec{u}/x_1 \otimes x_2\rangle_{B_1 \otimes B_2}) \\
&\quad = e^{i\theta_1} \cdot (\sum_{k=1}^{l} \gamma_k \vec{s}\langle\sigma_\Delta\rangle\langle\vec{v_k}/x\rangle_{B_1}\langle\vec{u_k}/y\rangle_{B_2}) \\
&\quad \twoheadrightarrow e^{i\theta_1} \cdot (\sum_{k=1}^{l} \gamma_k e^{i\rho_k} \vec{w_k})
\end{aligned}$$

Since $\vec{s}\langle\sigma_\Delta\rangle\langle\vec{v_k}/x\rangle_{B_1}\langle\vec{u_k}/y\rangle_{B_2} \in [\![\Delta, x_{B_1} : \sharp A_1, y_{B2} : \sharp A_2]\!]$, for every k, then $\vec{w_k} \in [\![C]\!]$. It remains to be seen that the term has norm-1, $\| \sum_{k=1}^{l} \gamma_k e^{i\rho_k} \vec{w_k} \| = 1$. For that, we observe:

$$\| \sum_{k=1}^{l} \gamma_k e^{i\rho_k} \vec{w_k} \|$$

$$
\begin{aligned}
&= \langle \sum_{k=1}^{l} \alpha_i e^{i\rho_k} \vec{w_k} \mid \sum_{k'=1}^{l} \gamma_{k'} e^{i\rho_{k'}} \vec{w_{k'}} \rangle \\
&= \sum_{k=1}^{l} \sum_{k'}^{l} \overline{\gamma_k e^{i\rho_k}}\ \gamma_{k'} e^{i\rho_{k'}} \langle \vec{w_k} \mid \vec{w_{k'}} \rangle \\
&= \sum_{k=1}^{l} \sum_{k'=1}^{l} \overline{\gamma_k e^{i\rho_k}}\ \gamma_{k'} e^{i\rho_{k'}} \langle \vec{v_k} \mid \vec{v_{k'}} \rangle \langle \vec{u_k} \mid \vec{u_{k'}} \rangle \quad \text{(from Lemma 22)} \\
&= \sum_{k=1}^{k} \sum_{k'=1}^{l} \overline{\gamma_k e^{i\rho_k}}\ \gamma_{k'} e^{i\rho_{k'}} \langle (\vec{u_k}, \vec{v_k}) \mid (\vec{u_{k'}}, \vec{v_{k'}}) \rangle \quad \text{(from Prop 5)} \\
&= \sum_{k=1}^{n} \overline{\gamma_k e^{i\rho_k}}\ \gamma_k e^{i\rho_k} \langle (\vec{v_k}, \vec{u_k}) \mid (\vec{v_k}, \vec{u_k}) \rangle \\
&\quad + \sum_{k,k'=1;k\neq k'}^{n} \overline{\gamma_k e^{i\rho_k}}\ \gamma_{k'} e^{i\rho_{k'}} \langle (\vec{v_k}, \vec{u_k}) \mid (\vec{v_{k'}}, \vec{u_{k'}}) \rangle \\
&= \sum_{k=1}^{n} \overline{\gamma_k e^{i\rho_k}}\ \gamma_k e^{i\rho_k} + 0 \\
&= \sum_{k=1}^{l} |\gamma_k|^2 |e^{i\rho_k}|^2 = 1
\end{aligned}
$$

Then $\sum_{i=1}^{n} \alpha_i \vec{w_i} \in \llbracket \sharp C \rrbracket$. Finally, we can conclude that:

$$(\mathsf{let}_{(B_1,B_2)}\ (x_1, x_2) = \vec{t}\ \mathsf{in}\ \vec{s})\langle \sigma \rangle \Vdash \sharp C$$

Case If the hypotheses are valid then:

- $\mathrm{dom}^{\sharp}(\Gamma) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma)$
- For every $\sigma_\Gamma \in \llbracket \Gamma \rrbracket$, $\vec{t}\langle \sigma_\Gamma \rangle \Vdash B_{\{\vec{v_i}\}_{i=1}^{n}}$
- For every $i \in \{0, \cdots, n\}, \mathrm{dom}^{\sharp}(\Delta) \subseteq \mathrm{FV}(\vec{s_i}) \subseteq \mathrm{dom}(\Delta)$
- For every $i \in \{0, \cdots, n\}, \sigma_\Delta \in \llbracket \Delta \rrbracket$, $\vec{s_i}\langle \sigma_\Delta \rangle \Vdash A$

From this we can conclude that:

- $\mathrm{dom}^{\sharp}(\Gamma, \Delta) \subseteq \mathrm{FV}(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})$
- $\mathrm{FV}(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\}) \subseteq \mathrm{dom}(\Gamma, \Delta)$

Then, given $\sigma \in \llbracket \Gamma, \Delta \rrbracket$, we have that $\langle \sigma \rangle = \langle \sigma_\Gamma \rangle \langle \sigma_\Delta \rangle$ for some $\sigma_\Gamma \in \llbracket \Gamma \rrbracket$ and $\sigma_\Delta \in \llbracket \Delta \rrbracket$. Using the first hypothesis we have that, $\vec{t}\langle \sigma_\Gamma \rangle \twoheadrightarrow e^{i\theta_1} \cdot \vec{v_k}$ for some $k \in \{1, \cdots, n\}$. From the second hypothesis we have that $\vec{s_i}\langle \sigma_\Delta \rangle \twoheadrightarrow e^{i\rho_i} \cdot \vec{u_i} \in \llbracket A \rrbracket$ for $i \in \{1, \cdots, n\}$. Therefore:

$$
\begin{aligned}
&(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v}_1 \mapsto \vec{s_1} \mid \cdots \mid \vec{v}_n \mapsto \vec{s_n}\})\langle \sigma \rangle \\
&= (\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v}_1 \mapsto \vec{s_1} \mid \cdots \mid \vec{v}_n \mapsto \vec{s_n}\})\langle \sigma_\Gamma \rangle \langle \sigma_\Delta \rangle \\
&= (\sum_{i=1}^{n} \alpha_i \mathsf{case}\ \vec{t}[\sigma_{\Gamma i}]\ \mathsf{of}\{\vec{v}_1 \mapsto \vec{s_1} \mid \cdots \mid \vec{v}_n \mapsto \vec{s_n}\})\langle \sigma_\Delta \rangle
\end{aligned}
$$

$$
\begin{aligned}
&\equiv (\mathsf{case}\ \sum_{i=1}^{n} \alpha_i \vec{t}[\sigma_{\Gamma i}]\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v}_n \mapsto \vec{s_n}\})\langle\sigma_\Delta\rangle \\
&= (\mathsf{case}\ \vec{t}\langle\sigma_\Gamma\rangle\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle\sigma_\Delta\rangle \\
&\twoheadrightarrow (\mathsf{case}\ e^{i\theta_1} \cdot \vec{v_k}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle\sigma_\Delta\rangle \\
&\to e^{i\theta_1} \cdot (\vec{s_k}\langle\sigma_\Delta\rangle) \\
&\twoheadrightarrow e^{i\theta_1} \cdot (e^{i\rho_k} \cdot \vec{u_k}) \qquad \text{Where: } \vec{u_k} \in \llbracket A \rrbracket \\
&\equiv e^{i\theta} \cdot \vec{u_k} \qquad \text{With: } \theta = \theta_1 + \theta_2
\end{aligned}
$$

Since we pose no restriction on k, we can conclude that:

$$(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle\sigma\rangle \Vdash A$$

UnitCase If the hypotheses are valid, then:

- $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma)$
- For every $\sigma_\Gamma \in \llbracket\Gamma\rrbracket$, $\vec{t}\langle\sigma_\Gamma\rangle \Vdash \sharp B_{\{\vec{v_i}\}_{i=1}^n}$
- For every i, $\mathrm{dom}^\sharp(\Delta) \subseteq \mathrm{FV}(\vec{s_i}) \subseteq \mathrm{dom}(\Delta)$
- For every $i \in \{0, \cdots, n\}, \sigma_\Delta \in \llbracket\Delta\rrbracket$, $\vec{s_i}\langle\sigma_\Delta\rangle \Vdash A$

From this we can conclude that:

- $\mathrm{dom}^\sharp(\Gamma, \Delta) \subseteq \mathrm{FV}(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v}_1 \mapsto \vec{s_1} \mid \cdots \mid \vec{v}_n \mapsto \vec{s_n}\})$
- $\mathrm{FV}(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v}_1 \mapsto \vec{s_1} \mid \cdots \mid \vec{v}_n \mapsto \vec{s_n}\}) \subseteq \mathrm{dom}(\Gamma, \Delta)$

Then, given $\sigma \in \llbracket\Gamma, \Delta\rrbracket$, we have that $\langle\sigma\rangle = \langle\sigma_\Gamma\rangle\langle\sigma_\Delta\rangle$ for some $\sigma_\Gamma \in \llbracket\Gamma\rrbracket$ and $\sigma_\Delta \in \llbracket\Delta\rrbracket$. Using the first hypothesis we have that, $\vec{t}\langle\sigma_\Gamma\rangle \Vdash \sharp B_{\{\vec{v_i}\}_{i=1}^n}$, then $\vec{t}\langle\sigma_\Gamma\rangle \twoheadrightarrow e^{i\theta_1} \cdot \vec{u} = e^{i\theta_1} \cdot (\sum_{i=1}^n \beta_i \vec{v_i})$ where $\sum_{i=1}^n |\beta_i|^2$. From the second hypothesis we have that $\vec{s_i}\langle\sigma_\Delta\rangle \twoheadrightarrow e^{i\rho_i} \cdot \vec{u_i} \in \llbracket A\rrbracket$ for $i \in \{1, \cdots, n\}$ and $u_i \perp u_j$ if $i \neq j$. Therefore:

$$
\begin{aligned}
&(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle\sigma\rangle \\
&\quad = (\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle\sigma_\Gamma\rangle\langle\sigma_\Delta\rangle \\
&\quad = (\sum_{i=1}^{n} \alpha_i \mathsf{case}\ \vec{t}[\sigma_{\Gamma i}]\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle\sigma_\Delta\rangle \\
&\quad \equiv (\mathsf{case}\ \sum_{i=1}^{n} \alpha_i \vec{t}[\sigma_{\Gamma i}]\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle\sigma_\Delta\rangle \\
&\quad = (\mathsf{case}\ \vec{t}\langle\sigma_\Gamma\rangle\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle\sigma_\Delta\rangle \\
&\quad \twoheadrightarrow (\mathsf{case}\ e^{i\theta_1} \cdot \vec{u}\ \mathsf{of}\{\vec{v} \mapsto \vec{s_1} \mid \cdots \mid \vec{w} \mapsto \vec{s_2}\})\langle\sigma_\Delta\rangle \\
&\quad \to e^{i\theta_1} \cdot (\sum_{i=1}^{n} \beta_i s_i)\langle\sigma_\Delta\rangle \\
&\quad = e^{i\theta_1} \cdot (\sum_{j=1}^{n} \delta_j (\sum_{i=1}^{n} \beta_i \vec{s_i})[\sigma_{\Delta j}]) \\
&\quad = e^{i\theta_1} \cdot (\sum_{j=1}^{n} \delta_j (\sum_{i=1}^{n} \beta_i \vec{s_i}[[\sigma_{\Delta j}]]))
\end{aligned}
$$

$$\equiv e^{i\theta_1} \cdot (\sum_{i,j=1}^{n} \beta_i \delta_j \vec{s_i}[\sigma_{\Delta j}])$$

$$= e^{i\theta_1} \cdot (\sum_{i=1}^{n} \beta_i \vec{s_i}\langle \sigma_\Delta \rangle)$$

$$\twoheadrightarrow e^{i\theta_1} \cdot (\sum_{i=1}^{n} \beta_i e^{i\rho_i} \cdot \vec{u_i})$$

It remains to be seen that: $\| \sum_{i=1}^{n} \beta_i e^{i\rho_i} \cdot \vec{u_i} \| = 1$:

$$\begin{aligned}
\| \sum_{i=1}^{n} \beta_i e^{i\rho_i} \cdot \vec{u_i} \| &= \langle \sum_{i=1}^{n} \beta_i e^{i\rho_i} \cdot \vec{u_i} \mid \sum_{i=1}^{n} \beta_i e^{i\rho_i} \cdot \vec{u_i} \rangle \\
&= \sum_{i,j=1}^{n} \overline{\beta_i e^{i\rho_i}} \beta_j e^{i\rho_j} \langle \vec{u_i} \mid \vec{u_j} \rangle \\
&= \sum_{i=1}^{n} \overline{\beta_i e^{i\rho_i}} \beta_i e^{i\rho_i} \langle \vec{u_i} \mid \vec{u_i} \rangle \\
&\quad + \sum_{i,j=1; i \neq j}^{n} \overline{\beta_i e^{i\rho_i}} \beta_j e^{i\rho_j} \langle \vec{u_i} \mid \vec{u_j} \rangle \\
&= \sum_{i=1}^{n} |\beta_i|^2 |e^{i\rho_i}|^2 + 0 \\
&= \sum_{i=1}^{n} |\beta_i|^2 = 1
\end{aligned}$$

Then we can conclude that $\sum_{i=1}^{n} \beta_i e^{i\rho_i} \vec{u_i} \in [\![\sharp A]\!]$ and finally:

$$(\mathsf{case}\ \vec{t}\ \mathsf{of}\{\vec{v_1} \mapsto \vec{s_1} \mid \cdots \mid \vec{v_n} \mapsto \vec{s_n}\})\langle \sigma \rangle \Vdash \sharp A$$

Sum If the hypothesis is valid then for every i, $\mathrm{dom}^\sharp(\Gamma) \subseteq \mathrm{FV}(\vec{t_i}) \subseteq \mathrm{dom}(\Gamma)$. From this we can conclude that $\mathrm{dom}^\sharp(\Gamma) \subseteq \sum_{i=1}^{n} \alpha_i \vec{t_i} \subseteq \mathrm{dom}(\Gamma)$. Given $\sigma \in [\![\Gamma]\!]$, we have for every i, $\vec{t_i}\langle \sigma \rangle \twoheadrightarrow e^{i\rho_i} \cdot \vec{v_i}$ where $\vec{v_i} \in [\![A]\!]$. Moreover, for every $i \neq j$, $\vec{v_i} \perp \vec{v_j}$ and $\sum_{i=1}^{n} |\alpha_i|^2 = 1$. Then:

$$\begin{aligned}
(\sum_{i=1}^{n} \alpha_i \vec{t_i})\langle \sigma \rangle &= \sum_{j=1}^{m} \beta_j (\sum_{i=1}^{n} \alpha_i \vec{t_i})[\sigma_j] \\
&\equiv \sum_{i=1}^{n} \alpha_i \sum_{j=1}^{m} \beta_j \vec{t_i}[\sigma_j] \\
&= \sum_{i=1}^{n} \alpha_i \vec{t_i}\langle \sigma \rangle \\
&\twoheadrightarrow \sum_{i=1}^{n} \alpha_i e^{i\rho_i} \vec{v_i}
\end{aligned}$$

It remains to be seen that $\| \sum_{i=1}^n \alpha_i e^{i\rho_i} \vec{v_i} \| = 1$. But:

$$\begin{aligned}
&\| \sum_{i=1}^n \alpha_i e^{i\rho_i} \vec{v_i} \| \\
&= \langle \sum_{i=1}^n \alpha_i e^{i\rho_i} \vec{v_i} \mid \sum_{i=1}^n \alpha_i e^{i\rho_i} \vec{v_i} \rangle \\
&= \sum_{i=i}^n \sum_{j=1}^n \overline{\alpha_i e^{i\rho_i}} \alpha_j e^{i\rho_j} \langle \vec{v_i} \mid \vec{v_j} \rangle \\
&= \sum_{i=1}^n \overline{\alpha_i e^{i\rho_i}} \alpha_i e^{i\rho_i} \langle \vec{v_i} \mid \vec{v_i} \rangle + \sum_{\substack{i,j=1 \\ i \neq j}}^n \overline{\alpha_i e^{i\rho_i}} \alpha_j e^{i\rho_j} \langle \vec{v_i} \mid \vec{v_j} \rangle \\
&= \sum_{i=1}^n |\alpha_i|^2 |e^{i\rho_i}|^2 + 0 \\
&= \sum_{i=1}^n |\alpha_i|^2 = 1
\end{aligned}$$

Then we can conclude that $\sum_{i=1}^n \alpha_i e^{i\rho_i} \vec{v_i} \in [\![\sharp A]\!]$ and finally $(\sum_{i=1}^n \alpha_i \vec{t_i})\langle\sigma\rangle \Vdash \sharp A$.

Weak Given $\sigma \in [\![\Gamma, x_A : B]\!]$, we observe that $\langle\sigma\rangle{=}\langle\sigma_\Gamma\rangle \langle\vec{v}/x\rangle_A$ for some $\sigma_\Gamma \in [\![\Gamma]\!]$ and $\vec{v} \in [\![B]\!]$. Using the first hypothesis, we know that $\vec{t}\langle\sigma_\Gamma\rangle \twoheadrightarrow e^{i\theta}\vec{w}$ where $\vec{w} \in [\![B]\!]$. Then we have:

$$\vec{t}\langle\sigma\rangle = \vec{t}\langle\sigma_\Gamma\rangle \langle\vec{v}/x\rangle_A \twoheadrightarrow e^{i\theta}\vec{w}\langle\vec{v}/x\rangle_A$$

Since $\vec{v} \in [\![A]\!]$, $\vec{w}\langle\vec{v}/x\rangle_A = \vec{w}[\vec{v}/x] = \vec{w}$ and $\vec{w} \in [\![B]\!]$, we can finally conclude that $\vec{t}\langle\sigma\rangle \Vdash B$.

Contr If the hypothesis is valid, we have that $\mathrm{dom}^\sharp(\Gamma, x_A : A, y_A : A) \subseteq \mathrm{FV}(\vec{t}) \subseteq \mathrm{dom}(\Gamma, x_A : A, y_A : A)$ and given $\sigma \in [\![\Gamma, x_A : A, y_A : A]\!]$, then $\vec{t}\langle\sigma\rangle \in [\![B]\!]$. Since we assume $\flat A$, we have that $\mathrm{dom}^\sharp(\Gamma, x_A : A, y_A : A) = \mathrm{dom}^\sharp(\Gamma, x_A : A)$. Therefore:

$$\mathrm{dom}^\sharp(\Gamma, x_A : A) \subseteq \mathrm{FV}(\vec{t})[x/y] \subseteq \mathrm{dom}(\Gamma, x_A : A)$$

Given $\sigma \in [\![\Gamma, x_A : A]\!]$, we observe that $\langle\sigma\rangle = \langle\vec{v}/x\rangle_A\langle\sigma_\Gamma\rangle$ with $\sigma_\Gamma \in [\![\Gamma]\!]$ and $\vec{v} \in [\![A]\!]$. Since $\vec{v} \in [\![A]\!]$, we know that $\vec{t}[\vec{v}/z] = \vec{t}\langle\vec{v}/z\rangle_A$ for any variable z. Then we have:

$$\begin{aligned}
\vec{t}[x/y]\langle\sigma\rangle &= \vec{t}[x/y]\langle\vec{v}/x\rangle_A\langle\sigma_\Gamma\rangle \\
&= \vec{t}[x/y][\vec{v}/x]\langle\sigma_\Gamma\rangle \\
&= \vec{t}[\vec{v}/y][\vec{v}/x]\langle\sigma_\Gamma\rangle \\
&= \vec{t}\langle\vec{v}/y\rangle_A\langle\vec{v}/x\rangle_A\langle\sigma_\Gamma\rangle
\end{aligned}$$

Since $\langle\vec{v}/y\rangle_A\langle\vec{v}/x\rangle_A\langle\sigma\rangle \in [\![\Gamma, x_A : A, y_A : A]\!]$, we get:

$$\vec{t}\langle\vec{v}/y\rangle \langle\vec{v}/x\rangle_A\langle\sigma_\Gamma\rangle \twoheadrightarrow e^{i\theta}\vec{w} \in [\![B]\!]$$

Then we can finally conclude that $\vec{t}[x/y]\langle\sigma\rangle \Vdash B$.

$$\mathsf{Deutsch} := (\lambda f_{\mathcal{P}} \,.\, \mathsf{let}_{(\mathbb{B},\mathbb{B})}\ (x, y) = (f(\mathbb{H}\,|0\rangle)(\mathbb{H}\,|1\rangle))\ \mathsf{in}\ ((\mathbb{H}x), y))$$
$$\mathbb{H} := \lambda x_{\mathbb{B}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|0\rangle \mapsto |+\rangle \mid |1\rangle \mapsto |-\rangle\}$$

Table 3.7: *Deutsch algorithm term*

Equiv It follows from definition and the fact that the reduction commutes with the congruence relation.

GlobalPhase It follows from the definition of type realizers.

□

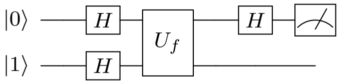
3.5 Examples

In this chapter we examine two use cases for the λ_B calculus. First, taking advantage of the basis types defined in the type algebra, we are able to give a more expressive type to the term encoding Deutsch's algorithm. Second, we make use of the deferred measurement principle and pattern matching from the case constructor to write a descriptive term encoding the quantum teleportation protocol.

3.5.1 Deutsch's algorithm

The Deutsch-Josza algorithm is a small example designed to showcase a problem which is solved exponentially faster by a quantum computer over a classical one. In it, we take as input a black box oracle which encodes a function $f : \{0,1\}^n \to \{0,1\}$. This function can be either *constant* or *balanced* (It outputs 0 for exactly half the inputs and 1 for the other half). The task to solve is to determine under which of the two classes the oracle falls.

In this section we will focus on the case where $n = 1$, the original formulation of Deutsch's algorithm. However, this results can be generalized to any arbitrary n. The quantum circuit implementing the algorithm is the following:



For a detailed discussion on the logic and operation of the algorithm, see [10]. We will do a comparison between Deutsch's algorithm written in different bases and see what information we can glean from the typing of the terms.

We first define the terms for the algorithm. The top level Deutsch abstraction, takes an oracle U_f which inputs two qubits $|xy\rangle$ and outputs $|x(y \oplus f(x))\rangle$ where $\oplus$ denotes addition modulo 2. The circuit will output $|0\rangle$ on the first qubit if the function f is balanced

On Table 3.8 we note the four possible oracles. D_1 and D_4 correspond to the oracles encoding the 0 and 1 constant functions and D_2, D_3 to the identity and bit-flip respectively.

Each of these oracles can be typed as $\mathbb{B} \to \mathbb{B} \to (\mathbb{B} \times \mathbb{B})$. But since we are passing $|+\rangle$ and $|-\rangle$ as arguments, the typing we would be able to assign is: $\sharp\mathbb{B} \to \sharp\mathbb{B} \to \sharp(\mathbb{B} \times \mathbb{B})$. Which means that the final typing for Deutsch would be:

$$\vdash \mathsf{Deutsch} : (\sharp\mathbb{B} \Rightarrow \sharp\mathbb{B} \Rightarrow (\sharp\mathbb{B} \Rightarrow \sharp\mathbb{B})) \Rightarrow \sharp(\mathbb{B} \times \mathbb{B})$$

This would seem to suggest that the result of the computation is a superposition of pairs of booleans.

$$
\begin{aligned}
D_1 &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, (x, y) \\
D_2 &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, \mathsf{CNOT}\ x\ y \\
D_3 &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, \mathsf{CNOT}\ x\ (\mathsf{NOT}\ y) \\
D_4 &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, (x, (\mathsf{NOT}\ y)) \\
&\text{Where:} \\
\mathsf{CNOT} &:= \lambda x_{\mathbb{B}} \,.\, \lambda y_{\mathbb{B}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|0\rangle \mapsto (|0\rangle, y) \mid |1\rangle \mapsto (|1\rangle, \mathsf{NOT}\ y)\} \\
\mathsf{NOT} &:= \lambda x_{\mathbb{B}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|0\rangle \mapsto |1\rangle \mid |1\rangle \mapsto |0\rangle\}
\end{aligned}
$$

Table 3.8: *Oracles implementing the four possible functions $f : \{0,1\} \mapsto \{0,1\}$*

$$
\begin{aligned}
\mathsf{Deutsch} &:= (\lambda U_{f_{\mathcal{P}}} \,.\, \mathsf{let}_{(\mathbb{X},\mathbb{X})}\ (x, y) = (U_f\,|+\rangle\,|-\rangle)\ \mathsf{in} \\
&\qquad \mathsf{case}\ x\ \mathsf{of}\{|+\rangle \mapsto |0\rangle \mid |-\rangle \mapsto |1\rangle\}) \\
D_1 &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, (x, y) \\
D_2 &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, \mathsf{CNOT}_{\mathbb{X}}\ x\ y \\
D_3 &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, \mathsf{CNOT}_{\mathbb{X}}\ x\ (\mathsf{NOT}_{\mathbb{X}}\ y) \\
D_4 &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, (x, (\mathsf{NOT}_{\mathbb{X}}\ y)) \\
&\text{Where:} \\
\mathsf{CNOT}_{\mathbb{X}} &:= \lambda x_{\mathbb{X}} \,.\, \lambda y_{\mathbb{X}} \,.\, \mathsf{case}\ y\ \mathsf{of}\{ \\
&\qquad |+\rangle \mapsto (x, |+\rangle) \mid \\
&\qquad |-\rangle \mapsto (\mathsf{Z}_{\mathbb{X}} x, |-\rangle) \\
&\qquad \} \\
\mathsf{Z}_{\mathbb{X}} &:= \lambda x_{\mathbb{X}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|+\rangle \mapsto |-\rangle \mid |-\rangle \mapsto |+\rangle\} \\
\mathsf{NOT}_{\mathbb{X}} &:= \lambda x_{\mathbb{X}} \,.\, \mathsf{case}\ x\ \mathsf{of}\{|+\rangle \mapsto |+\rangle \mid |-\rangle \mapsto (-1) \cdot |-\rangle\}
\end{aligned}
$$

Table 3.9: *Deutsch term and oracles in the shifted Hadamard basis.*

However, this approach underutilizes the amount of information we have available. We know that the oracle will receive specifically the state $|+-\rangle$, and so we can rewrite the intervening terms taking this information into account. In table 3.9 we restate the terms, but this time the abstractions and conditional cases are written in the basis $\mathbb{X} = \{|+\rangle, |-\rangle\}$.

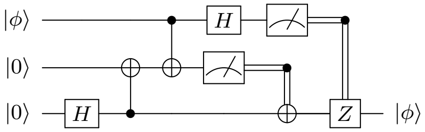
In this case, for each of the oracles we can assign the type $\mathbb{X} \to \mathbb{X} \to \mathbb{X} \times \mathbb{X}$ and type the term Deutsch as $(\mathbb{X} \to \mathbb{X} \to \mathbb{X} \times \mathbb{X}) \to \mathbb{B}$. There is a key difference here, the type of the oracles ensure that the result will be in the basis state $\mathbb{X} \times \mathbb{X}$. In other words, the result will be a pair with either $|+\rangle$ or $|-\rangle$ in its components (up to a global phase). Since we know this fact, we can manipulate the result of f as we would with classical bits, and discard the second component.

Both functions are equivalent on an operational point of view. But reframing it onto a different basis, allows us to give a more tight typing to the terms and more insight on how the algorithm works. If we analize the typing judgements, we observe that none of the variables has a $\sharp$ type. This has two consequences, first we can safely discard the second qubit and second, the Hadamard transform guarantees that the first qubit will yield a boolean. This correlates with the fact that Deutsch's algorithm is deterministic and we can statically ensure the result will be a basis vector.

3.5.2 Quantum teleportation

The *principle of deferred measurement* is a result which states that any quantum circuit can delay the measurements performed without modifying its outcome. More precisely, any gate controlled by the outcome of a measurement is equivalent to another gate whose control has not yet been measured. The calculus λ_B does not implement a mechanism to measure states, but using the case constructor is possible to simulate these quantum controlled gates.

A notable example of an algorithm which makes use of classical controlled gates is the *quantum teleportation*. In it, two agents (usually called Alice and Bob) share two parts of a Bell state and make use of the entanglement to move a quantum state from a qubit owned by Alice to a qubit owned by Bob. The quantum circuit representation of the algorithm is the following:



The algorithm first encodes the Bell state Φ^+ onto the second and third qubit and then performs a Bell basis measurement on the first and second qubit. In order to do this, it first decomposes applying a CNOT gate followed by a Hadamard gate on the first qubit (the adjoint of the Bell state generation). Then the first and second qubit are measured, which informs the correction needed for the third qubit to recover the state ϕ.

We can simulate the operation of the algorithm, via a λ-term which instead of outright measuring, describes the steps to take in each of the possible outcomes. A possible implementation is:

$$
\begin{aligned}
(\lambda x_{\mathbb{B}} \,.\, \mathsf{let}_{(\mathbb{B},\mathbb{B})}\ (y_1, y_2) = \Phi^+\ \mathsf{in}\ \ \mathsf{case}(x, y_1)\ \mathsf{of}\ \{ & \Phi^+ \mapsto (\Phi^+, y_2)\\
& \Phi^- \mapsto (\Phi^-, Z y_2)\\
& \Psi^+ \mapsto (\Psi^+, X y_2)\\
& \Psi^- \mapsto (\Psi^-, Z X y_2)\\
& \})
\end{aligned}
$$

The λ-term takes the state $|\phi\rangle$ as an argument, then matches the first qubit of the Bell pair and the $|\phi\rangle$ qubit, with the vectors of the Bell basis. In each branch, corrects the third qubit to recover the original $|\phi\rangle$ state. This is akin to controlling the correction with each of the Bell basis vectors.

The λ_B calculus provides syntax which allows the abstraction of the steps encoding and decoding on the Bell basis. This technique makes full use of the deferred measurement principle and can be applied to measurements on arbitrary bases. The final type of the term is $\sharp B_{\mathbb{B}} \Rightarrow \sharp B_{\mathsf{Bell}} \times \sharp B_{\mathbb{B}}$.

TODO: HABLAR DEL PAPER DE SIMON ACÁ EN UNA NUEVA SUBSECTION

3.6 Conclusion

In this chapter we explore a quantum-data/quantum-control λ-calculus, with the additional feature of framing the abstraction in different bases besides the canonical one.

The mechanism needed to implement this idea is the decoration in λ-terms and let constructors. Along with a new substitution which dictates the decomposition of the value distribution onto different bases. These changes do not add expressive power to the original calculus it is based from, however they provide a different point of view when writing programs.

The reduction system itself orchestrates the computation and makes use of the syntax and substitution previously defined. The main point to note is that the evaluation commutes with the congruence relationship, ensuring that interpreting a vector in a different basis does not alter the result of the computation. And in turn, allows us to consider value distributions modulo this congruence.

The previous work pays its dividends when considering the realizability model. The inclusion of the atomic types B_X, is used to characterize the abstractions that represent unitary functions. This is the main result of the section and is a generalization of the characterization found in [13]. Here, the use of basis types gives way to a simpler proof.

The other main result of the chapter is the validity of the several typing rules described in Table 3.6. Extracting them via the realizability technique, ensures their correctness and can later form the foundation of the type system for a programming language.

Finally, we present two examples that showcase the advantage of the typing system and syntax. First Deutsch's algorithm, which exhibits a more expressive type and in turn, allows to treat the result classically. Second, the case for quantum teleportation, where we are able to simulate gates controlled by a Bell basis measurement as branches on a pattern matching case.

There are a few remaining lines of research that stem from this work. A natural progression would be to provide a categorical model to study the calculus through a different lens and relate it to other well studied systems.

As well, we could try to give a translation into an intermediate language like ZX alongside the lines of the second chapter. Proving that, despite the programs being detached from the circuitry, they can still be implemented concretely.

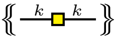
Appendices

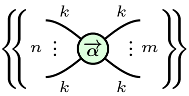
.1 Semantics of the SZX calculus

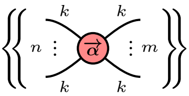
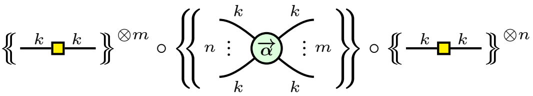
We reproduce below the standard interpretation of SZX$_{\overline{\mp}}$ diagrams as density matrices and completely positive maps [6, 4], modulo scalars.

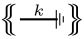
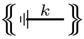
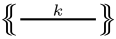
Let $D_n \subseteq \mathbb{C}^{2^n \times 2^n}$ be the set of n-qubit density matrices. We define the functor $\{\cdot\} : \mathrm{ZX}_{\overline{\mp}} \to \mathbf{CPM}(\mathbf{Qubit})$ which associates to any diagram $\mathscr{D} : n \to m$ a completely positive map $\{\mathscr{D}\} : D_n \to D_m$, inductively as follows.

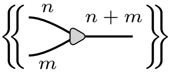
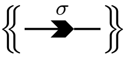
$$\{\mathscr{D}_1 \otimes \mathscr{D}_2\} := \{\mathscr{D}_1\} \otimes \{D_2\} \qquad \{\mathscr{D}_2 \circ \mathscr{D}_1\} := \{\mathscr{D}_2\} \circ \{\mathscr{D}_1\}$$

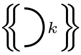
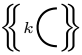
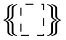
 $:= \rho \mapsto V\ \rho\ V^\dagger$ where $V = \sum_{x,y \in \mathbb{F}_2^k} (-1)^{x \bullet y} \left|y\right\rangle\!\left\langle x\right|$

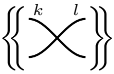
 $:= \rho \mapsto V\ \rho\ V^\dagger$ where $V = \sum_{x \in \mathbb{F}_2^k} e^{ix\bullet\vec{\alpha}} \left|x\right\rangle^{\otimes m} \left\langle x\right|^{\otimes n}$

 := 

 $:= \rho \mapsto \sum_{x \in \mathbb{F}_2^k} \left\langle x\right| \rho \left|x\right\rangle$  $:= \sum_{x \in \mathbb{F}_2^k} \left|x\right\rangle\!\left\langle x\right|$  $:= \rho \mapsto \rho$

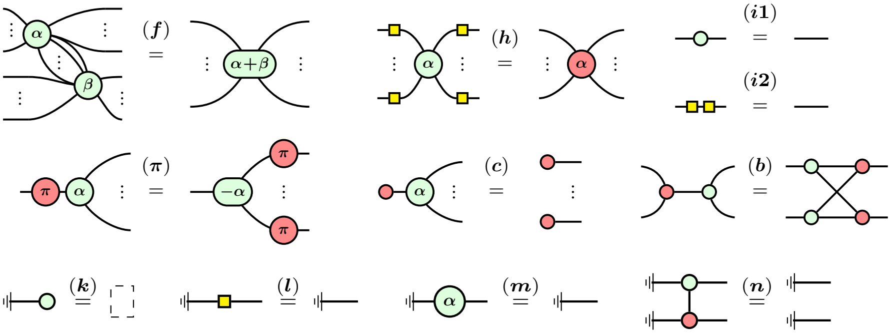
 $:= \rho \mapsto \rho$  $:= \rho \mapsto V\ \rho\ V^\dagger$ where $V = \sum_{x \in \mathbb{F}_2^k} \left|\sigma(x)\right\rangle\!\left\langle x\right|$

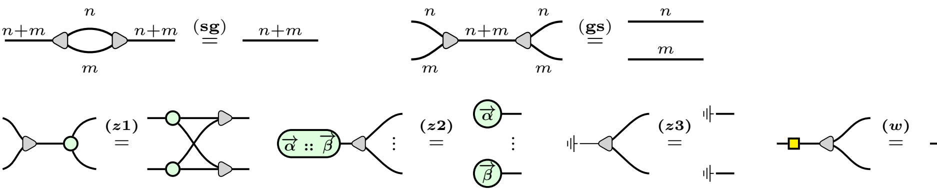
 $:= \rho \mapsto \sum_{x \in \mathbb{F}_2^k} \left\langle xx\right| \rho \left|xx\right\rangle$  $:= \sum_{x \in \mathbb{F}_2^k} \left|xx\right\rangle\!\left\langle xx\right|$  $:= Id_0$

 $:= \rho \mapsto V\ \rho\ V^\dagger$ where $V = \sum_{x \in \mathbb{F}_2^k, y \in \mathbb{F}_2^l} \left|yx\right\rangle\!\left\langle xy\right|$

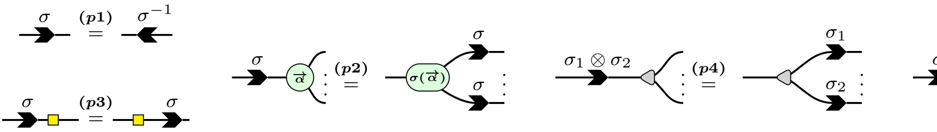
where $\forall u, v \in \mathbb{R}^n, u \bullet v = \sum_{i=1}^{m} u_i v_i$.

The SZX$_{\overline{\mp}}$ calculus defines a set of rewrite rules, shown below.

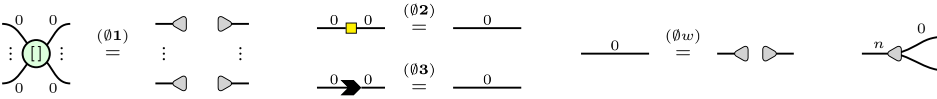




Additionally, for the arrows restricted to permutations of wires we have the following rules [4]:



Finally, since wires with cardinality zero correspond to empty mappings they can be discarded from the diagrams.



.2 Operational Semantics of the λ_D calculus

We define a weak call-by-value small step operational semantics on Table 10.

A key point to note here is that every rewriting rule preserves the state. There are no measurements or unitary operations applied, the rewriting is merely syntactical. Since our goal is translation into an SZX-diagram, this system is powerful enough. We include the rewrite rules for the primitives on Table 11.

Additionally, we define useful macros based on these functions on Table 12. They provide syntactic sugar to deal with state vectors.

.3 Implementation of primitives in Proto-Quipper-D

The implicitly recursive primitives defined in Section 2.3 can be implemented in proto-quipper-D as follows. The implementation has been checked with the `dpq` tool implemented by Frank Fu (see `https://gitlab.com/frank-peng-fu/dpq-remake`).

```
module Primitives where
import "/dpq/Prelude.dpq"

foreach : ! forall a b (n : Nat)
  -> (Parameter a) => !(a -> b) -> Vec a n -> Vec b n
foreach f l = map f l

split : ! forall a (n : Nat) (m: Nat) -> Vec a (n+m) -> Vec a n * Vec a m
split n m v =
  case v of
    VNil -> (VNil, VNil)
    VCons x v' ->
      case n of
        Z -> (VNil, v)
```

$$\begin{aligned}
V := {} & x \mid C \mid 0 \mid 1 \mid \texttt{meas} \mid \texttt{new} \mid \texttt{U} \mid \\
& \lambda x^S.M \mid \lambda' x^P.M \mid \star \mid M \otimes N \mid \\
& \texttt{VNil} \mid M :: N
\end{aligned}$$

$$\begin{aligned}
(\lambda x.M)V &\to M[V/x] \\
(\lambda' x.M)@V &\to M[V/x] \\
\texttt{let } x \otimes y = M_1 \otimes M_2 \texttt{ in } N &\to N[x/M_1][y/M_2] \\
\texttt{let } x :: y = M_1 :: M_2 \texttt{ in } N &\to N[x/M_1][y/M_2] \\
\texttt{ifz } V \texttt{ then } M \texttt{ else } N &\to \begin{cases} M & \text{If } V = 0 \\ N & \text{Otherwise} \end{cases} \\
\star\ ;\ M &\to M \\
\texttt{VNil}\ ;_v\ M &\to M \\
V_1 \square V_2 &\to V \qquad \text{Where } V_i = n_i \text{ and } V = n_1 \square n_2 \\
\texttt{for } k \texttt{ in } M_1\ :: M_2 \texttt{ do } N &\to N[k/M_1]\ :: \texttt{for } k \texttt{ in } M_2 \texttt{ do } N \\
\texttt{for } k \texttt{ in VNil do } N &\to \texttt{VNil}
\end{aligned}$$

$$\frac{M \to N}{MV \to NV} \qquad \frac{M \to N}{LM \to LN} \qquad \frac{M \to N}{M@V \to N@V} \qquad \frac{M \to N}{L@M \to L@N}$$

$$\frac{M \to N}{\texttt{let } x \otimes y = M \texttt{ in } L \to \texttt{let } x \otimes y = N \texttt{ in } L} \qquad \frac{M \to N}{\texttt{let } x :: y = M \texttt{ in } L \to \texttt{let}}$$

$$\frac{M \to N}{L \square M \to L \square N} \qquad \frac{M \to N}{M \square V \to M \square V}$$

$$\frac{M \to N}{M\ ;\ L \to N\ ;\ L} \qquad \frac{M \to N}{\texttt{let } x : y = M \texttt{ in } L \to \texttt{let } x : y = N \texttt{ in } L}$$

$$\frac{M \to N}{\texttt{ifz } M \texttt{ then } L_1 \texttt{ else } L_2 \to \texttt{ifz } N \texttt{ then } L_1 \texttt{ else } L_2} \qquad \frac{M -}{\texttt{for } k \texttt{ in } M \texttt{ do } L -}$$

Table 10: *Rewrite system for λ_D.*

$$\begin{aligned}
\texttt{accuMap}\ @n\ xs\ fs\ z \rightarrow & \texttt{ifz}\ n\ \texttt{then}\ xs\ ;_v\ \ fs\ ;_v\ \ \texttt{VNil} \otimes z\ \texttt{else} \\
& \texttt{let}\ x :: xs' = xs\ \texttt{in}\ \texttt{let}\ f :: fs' = fs\ \texttt{in} \\
& \texttt{let}\ y \otimes z' = f\ x\ z\ \texttt{in}\ \texttt{let}\ ys \otimes z'' = \texttt{accuMap}\ @(n - \\
& (y :: ys) \otimes z'' \\
\texttt{split}\ @n\ @m\ xs \rightarrow & \texttt{ifz}\ n\ \texttt{then}\ \texttt{VNil} \otimes xs\ \texttt{else}\ \texttt{let}\ y :: xs' = xs\ \texttt{in} \\
& \texttt{let}\ ys_1 \otimes ys_2 = \texttt{split}@(n-1)\ @m\ xs'\ \texttt{in}\ (y :: ys_1) \\
\texttt{append}\ @n\ @m\ xs\ ys \rightarrow & \texttt{ifz}\ n\ \texttt{then}\ xs\ ;_v\ \ ys\ \texttt{else} \\
& \texttt{let}\ x :: xs' = xs\ \texttt{in}\ x :: (\texttt{append}\ @(n-1)\ @m\ xs'\ ys \\
\texttt{drop}\ @n\ xs \rightarrow & \texttt{ifz}\ n\ \texttt{then}\ xs\ ;_v\ \star\ \texttt{else}\ \texttt{let}\ x :: xs' = xs\ \texttt{in}\ x\ ;\ \texttt{dro} \\
\texttt{range}\ \ @n\ @m \rightarrow & \texttt{ifz}\ m - n\ \texttt{then}\ \texttt{VNil}\ \texttt{else}\ n\ \ :: \texttt{range}\ \ @(n+1)\ @m
\end{aligned}$$

Table 11: *Reductions pertaining to the primitives.*

$$\begin{aligned}
\texttt{map}\ @n\ xs\ fs := & \ \texttt{let}\ fs' \otimes u_1 = \texttt{accuMap}\ @n\ fs \\
& \qquad (\texttt{for}\ k\ \texttt{in}\ (0..n)\ \texttt{do}\ \lambda f.\lambda u.(\lambda x.\lambda u.fx \otimes u) \otimes u)\ \star \\
& \ \texttt{in}\ \texttt{let}\ xs' \otimes u_2 = \texttt{accuMap}\ @n\ xs\ fs'\ \star \\
& \ \texttt{in}\ u_1\ ;\ u_2\ ;\ xs' \\
\texttt{fold}\ @n\ xs\ fs\ z := & \ \texttt{let}\ fs' \otimes u = \texttt{accuMap}\ @n\ fs \\
& \qquad (\texttt{for}\ k\ \texttt{in}\ (0..n)\ \texttt{do}\ \lambda f.\lambda u.(\lambda x.\lambda y. \star \otimes f\ x\ y) \otimes u)\ \star \\
& \ \texttt{in}\ \texttt{let}\ us \otimes r = \texttt{accuMap}\ @n\ xs\ fs'\ z \\
& \ \texttt{in}\ u\ ;\ \texttt{drop}\ @n\ us\ ;\ r \\
\texttt{compose}\ @n\ xs = & \ \texttt{fold}\ @n\ xs\ (\texttt{for}\ k\ \texttt{in}\ 0..n\ \texttt{do}\ (\lambda f.\lambda g.\lambda x.f\ (g\ x)))\ (\lambda x.x)
\end{aligned}$$

Table 12: *Function macros.*

```
        S n' ->
          let (v1, v2) = split n' m v'
          in (VCons x v1, v2)

cons : ! forall a (n : Nat) (m: Nat) -> Vec a n -> Vec a m -> Vec a (n+m)
cons n m vn vm =
  case vn of
    VNil -> VNil
    VCons x vn' -> x : cons n m vn' vm

accuMap : ! forall a b c (n : Nat)
  -> Vec a n -> Vec (a -> c -> (b,c)) n -> c -> (Vec b n, c)
accuMap n xs fs z =
  case xs of
    VNil -> (VNil, z)
    VCons x xs' ->
      case fs of
        VCons f fs' ->
          case n of
            S n' ->
              let (y, z') = f x z in
              let (ys, z'') = accuMap n' xs' fs' z' in
              (VCons y ys, z'')

mapp : ! forall a b (n : Nat) -> Vec a n -> Vec (a -> b) n -> Vec b n
mapp n v f =
  let (v', _) = accuMap n v (\x z -> (f x, z)) VNil
  in v'

fold : ! forall a b (n : Nat) -> Vec a n -> Vec (a -> b -> b) n -> b -> b
fold n v f z =
  let (_, z') = accuMap n v (\x z -> (VNil, f x z)) z
  in z'

compose : ! (n : Nat) -> Vec (a -> a) n -> a -> a
compose n fs x = fold fs (replicate n (\f x -> f x)) x

range_aux : ! (n : Nat) -> (m : Nat) -> Nat -> Vec Nat (minus m n)
range_aux n m x =
  case m of
    Z -> VNil
    S m' -> case n of
              Z -> let r' = range_aux Z m' (S x)
                   in subst (\x -> Vec Nat x) (minusSZ' m') (VCons x r')
              S n' -> range_aux n' m' (S x)

range : ! (n : Nat) -> (m : Nat) -> Vec Nat (minus m n)
range n m = range_aux n m Z

drop : ! (n : Nat) -> Vec Unit n -> Unit
drop n v = case n of
            Z -> ()
            S n' -> case v of
                      VCons _ v' -> drop n' v'
```

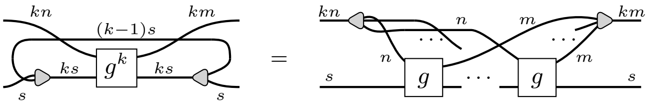
.4 QFT algorithm in Quipper code

The following Proto-Quipper-D code corresponds to the algorithm presented in Section 2.5. This implementation has been checked with the `dpq` tool implemented by Frank Fu (see `https://gitlab.com/frank-peng-fu/dpq-remake`). Notice that, in contrast to the presented lambda terms, the type checker implementation requires explicit encodings of the Leibniz equalities between parameter types.

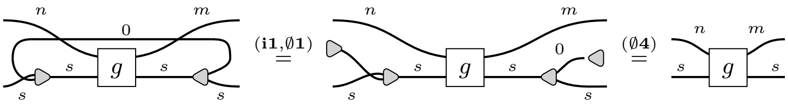
```
module Qft where
import "/dpq/Prelude.dpq"

crot : ! (n : Nat) -> Qubit * Qubit -> Qubit * Qubit
crot n q = let (q',c) = q in flip $ R n q' c

-- Specify types to help the typechecker
applyCrot_aux : ! (n : Nat) -> Qubit -> Qubit -> Qubit * Qubit
applyCrot_aux n ctrl q = crot n (q, ctrl)

-- Apply a CROT sequence to a qubit register, ignoring the first k qubits.
applyCrot : ! (n k : Nat) -> Vec Qubit n -> Vec Qubit n
applyCrot n k qs =
  let WithEq r e = inspect (minus n k)
  in case r of
      Z -> qs
      S n' ->
        let
          -- e : Eq Nat (S n') (minus n k)
          -- e' : Eq Nat (add k (S n')) n
          e' = trans (symAdd k (S n')) $ minusPlus n n' k $ sym (minus n k) e
          -- qs' : Vec Qubit (minus n k)
          qs' = subst (\m -> Vec Qubit m) (sym (add k (S n')) e') qs
          (head, qs') = split k (S n') $ qs'
          (q,ctrls) = chop qs'
          -- fs : Vec (Qubit -> Qubit -> Qubit * Qubit) (minus n' Z)
          fs = foreach (\k -> applyCrot_aux (S(S k))) $ 0..n'
          -- fs : Vec (Qubit -> Qubit -> Qubit * Qubit) Z
          eq = sym n' $ minusZ n'
          fs = subst (\m -> Vec (Qubit -> Qubit -> Qubit * Qubit) m) eq fs
          (ctrls', q') = accumap fs (H q) ctrls
        in subst (\m -> Vec Qubit m) e' $ append head (VCons q' ctrls')

-- Required for the type checker to derive the second !
qft_aux : ! (n : Nat) -> ! (k : Nat) -> Vec Qubit n -> Vec Qubit n
qft_aux n head_size qs = applyCrot n head_size qs

qft : ! (n : Nat) -> Vec Qubit n -> Vec Qubit n
qft n qs = let f = qft_aux n in compose' (foreach f $ reverse_vec (0..n)) qs
```

.5 Proofs

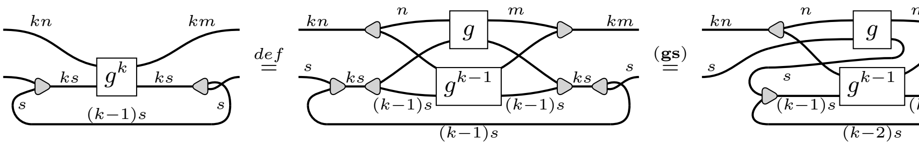
Lemma 11. *([1]) Let $g : 1_n \otimes 1_s \to 1_m \otimes 1_s$ and $k \geq 1$, then*

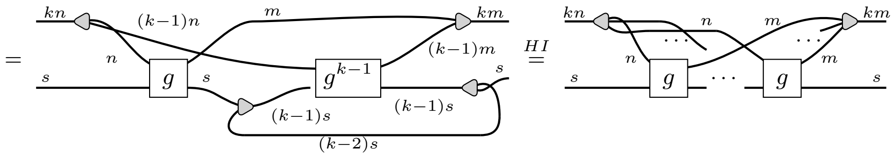


Proof. By induction on k. If $k = 1$,



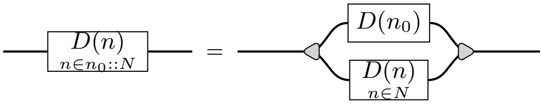
If $k > 1$,





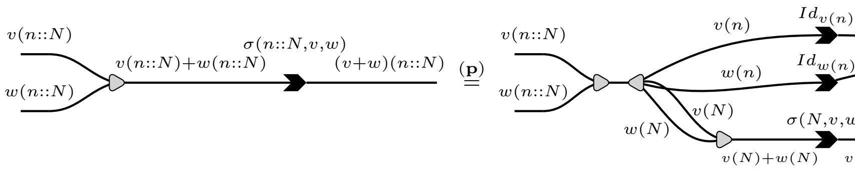
□

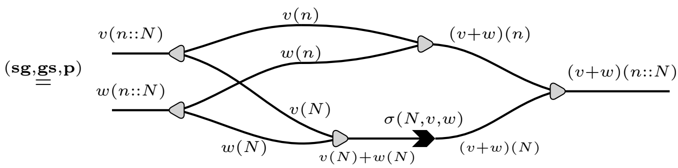
Lemma 12. *([2]) For any diagram family D, $n_0 : \mathbb{N}$, $N : \mathbb{N}^k$,*



Proof. By induction on the term construction

- If $\mathscr{D}$ is a gather,

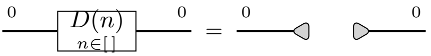




- The other cases can be directly derived from the commutation properties of the gather generator via rules $(z1), (z2), (z3), (w)$, and $(p4)$.

□

Lemma 13. *(3) A diagram family initialized with the empty list corresponds to the empty map. For any diagram family D,*



Proof. Notice that any wire in the initialized diagrams has cardinality zero. By rules $(\emptyset\mathbf{1})$, $(\emptyset\mathbf{2})$, $(\emptyset\mathbf{3})$, $(\emptyset\mathbf{4})$, and $(\emptyset\mathbf{w})$ every internal node can be eliminated from the diagram. □

Lemma 14. *(4) The list instantiation procedure on an n-node diagram family adds $\mathcal{O}(n)$ nodes to the original diagram.*

Proof. By induction on the term construction. Notice that the instantiation of any term except the gather does not introduce any new nodes, and the gather introduction creates exactly one extra node. Therefore, the list instantiation adds a number of nodes equal to the number of gather generators in the diagram. □

Lemma 15. *Given type judgements $\Phi, x : A \vdash M : B$, and $\Phi \vdash N : A$. $\lfloor M \rfloor_{x:A,\Phi} (\lfloor N \rfloor_{\Phi}, |\Phi|) = \lfloor M[N/x] \rfloor_{\Phi} (\ |\Phi|\ )$.*

Proof. Proof by straightforward induction on M. □

Lemma 16. *(5) Given an evaluable type A and a type judgement $\Phi \vdash M : A$, $\lfloor M \rfloor_{\Phi} \in \bigtimes_{x:P\in\Phi} \lfloor P \rfloor \to \lfloor A \rfloor$.*

Proof. By induction on the typing judgement $\Phi \vdash M : A$:

- If $\Phi \vdash n : \texttt{Nat}$, then $\lfloor n \rfloor_{\Phi} = |\Phi| \mapsto n \in \bigtimes_{x:P\in\Phi} \lfloor P \rfloor \to \mathbb{N}$.
- If $\Phi, x : A \vdash x : A$, then $\lfloor x \rfloor_{x:A,\Phi} = x, |\Phi| \mapsto x \in \bigtimes_{y:P\in x:A,\Phi} \lfloor P \rfloor \to \lfloor A \rfloor$.
- If $\Phi \vdash M \Box N\ :\ \texttt{Nat}\Phi, \Gamma, \Delta \vdash M :: N\ \ : \texttt{Vec}\ (n+1)\ A$, then $\lfloor M \Box N \rfloor_{\Phi} = |\Phi| \mapsto \lfloor M \rfloor_{\Phi} (|\Phi|) \Box\ \lfloor N \rfloor_{\Phi} (\ |\Phi|\ )$. By inductive hypothesis, $\lfloor M \rfloor_{\Phi} (|\Phi|), \lfloor M \rfloor_{\Phi} (|\Phi|) \in \mathbb{N}$. Then, $|\Phi| \mapsto \lfloor M \rfloor_{\Phi} (|\Phi|) \Box \lfloor N \rfloor_{\Phi} (|\Phi|) \in \bigtimes_{x:P\in\Phi} \lfloor P \rfloor \to \mathbb{N}$.
- If $\Phi \vdash \lambda' x.M : (x : P) \to B$ then $\left\lfloor \lambda' x^P.M \right\rfloor_{\Phi} = x, |\Phi| \mapsto \lfloor M \rfloor_{\Phi} (x, |\Phi|)$. By inductive hypothesis, $\lfloor M \rfloor_{\Phi} (x, |\Phi|) \in \lfloor B \rfloor$. Then, $|\Phi| \mapsto \lfloor M \rfloor_{\Phi} (x, |\Phi|) \in \bigtimes_{y:P\in x:P\Phi} \lfloor P \rfloor \to \lfloor B \rfloor$.
- If $\Phi, \Gamma \vdash M@N : B[x/r]$, then $\lfloor M@N \rfloor_{\Phi} = |\Phi| \mapsto \lfloor M \rfloor_{\Phi} (\lfloor N \rfloor_{\Phi} (|\Phi|), \Phi)$. By inductive hypothesis, $\lfloor N \rfloor_{\Phi} (|\Phi|) \in \mathbb{N}$ and $x \mapsto \lfloor M \rfloor_{\Phi} (x, \Phi) \in \lfloor x : \texttt{Nat} \to B[x] \rfloor$. Then, $|\Phi| \mapsto \lfloor M \rfloor_{\Phi} (\lfloor N \rfloor_{\Phi} (|\Phi|), \Phi) \in \bigtimes_{y:P\in\Phi} \lfloor P \rfloor \to \lfloor B[A/x] \rfloor$.
- If $\Phi \vdash \texttt{VNil}^A : \texttt{Vec}\ 0\ A$, then $\left\lfloor \texttt{VNil}^P \right\rfloor_{\Phi} = |\Phi| \mapsto [] \in \bigtimes_{x:P\in\Phi} \lfloor P \rfloor \to \mathbb{N}^0$.
- If $\Phi, \Gamma, \Delta \vdash M :: N\ \ : \texttt{Vec}\ (n+1)\ A$, then $\lfloor M\ ::\ N \rfloor_{\Phi} = |\Phi| \mapsto \lfloor M \rfloor_{\Phi} (|\Phi|) \times \lfloor N \rfloor_{\Phi} (|\Phi|)$. By inductive hypothesis $M_{\Phi}(|\Phi|) \in \lfloor A \rfloor$ and $\lfloor N \rfloor_{\Phi} (|\Phi|) \in \lfloor A \rfloor^n$. Then, $|\Phi| \mapsto \lfloor M \rfloor_{\Phi} (|\Phi|) \times \lfloor N \rfloor_{\Phi} (|\Phi|) \in \bigtimes_{x:P\in\Phi} \lfloor P \rfloor \to \lfloor A \rfloor^{n+1}$.
- If $\Phi, \Gamma \vdash \texttt{ifz}\ L\ \texttt{then}\ M\ \texttt{else}\ N : A$, then
$$\lfloor \texttt{ifz}\ L\ \texttt{then}\ M\ \texttt{else}\ N \rfloor_{\Phi} = |\Phi| \mapsto \begin{cases} \lfloor M \rfloor_{\Phi} (|\Phi|) \text{ if } \lfloor L \rfloor_{\Phi} (|\Phi|) = 0 \\ \lfloor N \rfloor_{\Phi} (|\Phi|) \text{ otherwise} \end{cases}.$$
By inductive hypothesis $\lfloor m \rfloor_{\Phi} (|\Phi|), \lfloor N \rfloor_{\Phi} (|\Phi|) \in \lfloor A \rfloor$ and $\lfloor L \rfloor_{\Phi} (|\Phi|) \in \mathbb{N}$. Then $\lfloor \texttt{ifz}\ L\ \texttt{then}\ M\ \texttt{else}\ N \rfloor_{\Phi} \in \bigtimes_{x:P\in\Phi} \lfloor P \rfloor \to \lfloor A \rfloor$.

- If $\Phi \vdash \mathtt{for}\ k\ \mathtt{in}\ N\ \mathtt{do}\ M : \mathtt{Vec}\, n\, A$, then $\lfloor \mathtt{for}\ k\ \mathtt{in}\ V\ \mathtt{do}\ M \rfloor_{\Phi} = |\Phi| \mapsto \bigtimes_{k \in \lfloor N \rfloor_{\Phi}(|\Phi|)} \lfloor M \rfloor_{k:\mathtt{Nat}}$ $|\Phi|)$. By induction hypothesis, $\lfloor N \rfloor_{\Phi}(|\Phi|) \in \mathtt{Nat}^n$ and $x \mapsto \lfloor M \rfloor_{\Phi}(x, \Phi) \in \lfloor k : \mathtt{Nat} \to A[k] \rfloor$. Then $|\Phi| \mapsto \bigtimes_{k \in \lfloor N \rfloor_{\Phi}(|\Phi|)} \lfloor M \rfloor_{k:\mathtt{Nat}\Phi}(k, |\Phi|) \in \bigtimes_{x:P \in \Phi} \lfloor P \rfloor \to \lfloor A[k/N] \rfloor^n$.

- If $\Phi \vdash \mathtt{let}\ x^B : y^{\mathtt{Vec}\ n\ B} = M\ \mathtt{in}\ N : A$, then $\left\lfloor \mathtt{let}\ x^P :: y^{\mathtt{Vec}\ n\ P} = M\ \mathtt{in}\ N \right\rfloor_{\Phi} =$ $|\Phi| \mapsto$
  $\lfloor N \rfloor_{x:P, y:\mathtt{Vec}\ n\ P\Phi}(y_1, [y_2, \ldots, y_n], |\Phi|)$ where $[y_1, \ldots, y_n] = \lfloor M \rfloor_{\Phi}(|\Phi|)$.
  By inductive hypothesis $\lfloor M \rfloor_{\Phi}(|\Phi|) \in \lfloor B \rfloor^n$ and $\lfloor N \rfloor_{x:P, y:\mathtt{Vec}\ n\ P\Phi}(y_1, [y_2, \ldots, y_n], |\Phi|) \in \lfloor A \rfloor$. Then $|\Phi| \mapsto \lfloor N \rfloor_{x:P, y:\mathtt{Vec}\ n\ P\Phi}(y_1, [y_2, \ldots, y_n], |\Phi|) \in \bigtimes_{x:P \in \Phi} \lfloor P \rfloor \to \lfloor A \rfloor$.

- If $\Phi \vdash \mathtt{range}\ : (n : \mathtt{Nat}) \to (m : \mathtt{Nat}) \to \mathtt{Vec}\ (m - n)\ \mathtt{Nat}$, then $\lfloor \mathtt{range}\ \rfloor_{\Phi} =$ $n, m, |\Phi| \mapsto \bigtimes_{i=n}^{m-1} i \in \bigtimes_{z:P \in x:\mathtt{Nat}, y:\mathtt{Nat}, \Phi} \lfloor P \rfloor \to \mathbb{N}^{m-n}$

□

Lemma 17. *([6](#)) Given an evaluable type A, a type judgement $\Phi \vdash M : A$, and $M \to N$, then $\lfloor M \rfloor_{\Phi} = \lfloor N \rfloor_{\Phi}$.*

Proof. By induction on the evaluation function $\lfloor M \rfloor_{\Phi}$:

- If $M = x$, $M = n$, $M = \mathtt{VNil}^{\mathtt{Nat}}$, $M = \lambda' x.M'$, $M = M_1\ ::\ M_2$ or $M = \mathtt{range}\ $ then M is in normal form and it does not reduce.
- If $M = M_1 \square M_2$ we have three cases:
  - If $M \to M_1 \square N$ with $M_2 \to N$, then:
  $$\begin{aligned} \lfloor M_1 \square M_2 \rfloor_{\Phi} &= |\Phi| \mapsto \lfloor M_1 \rfloor_{\Phi}(|\Phi|) \square \lfloor M_2 \rfloor_{\Phi}(|\Phi|) \\ &= |\Phi| \mapsto \lfloor M_1 \rfloor_{\Phi}(|\Phi|) \square \lfloor N \rfloor_{\Phi}(|\Phi|) \\ &= \lfloor M_1 \square N \rfloor_{\Phi} \end{aligned}$$
  - If $M \to N \square V$ with $M_1 \to N$, then:
  $$\begin{aligned} \lfloor M_1 \square V \rfloor_{\Phi} &= |\Phi| \mapsto \lfloor M_1 \rfloor_{\Phi}(|\Phi|) \square \lfloor V \rfloor_{\Phi}(|\Phi|) \\ &= |\Phi| \mapsto \lfloor N \rfloor_{\Phi}(|\Phi|) \square \lfloor V \rfloor_{\Phi}(|\Phi|) \\ &= \lfloor N \square V \rfloor_{\Phi} \end{aligned}$$
  - If $M \to n$ with $M_i = n_i \in \mathbb{N}$ and $n = n_1 \square n_2$, then:
  $$\begin{aligned} \lfloor n_1 \square n_2 \rfloor_{\Phi} &= |\Phi| \mapsto \lfloor n_1 \rfloor_{\Phi}(|\Phi|) \square \lfloor n_2 \rfloor_{\Phi}(|\Phi|) \\ &= |\Phi| \mapsto n_1 \square n_2 \\ &= |\Phi| \mapsto n \\ &= \lfloor n \rfloor_{\Phi} \end{aligned}$$
- If $M = M_1\ @\ M_2$ we have three cases:
  - If $M \to M_1\ @\ N$ with $M_2 \to N$, then:
  $$\begin{aligned} \lfloor M_1\ @\ M_2 \rfloor_{\Phi} &= |\Phi| \mapsto \lfloor M_1 \rfloor_{\Phi}(\lfloor M_2 \rfloor_{\Phi}(|\Phi|), \Phi) \\ &= |\Phi| \mapsto \lfloor M_1 \rfloor_{\Phi}(\lfloor N \rfloor_{\Phi}(|\Phi|), \Phi) \\ &= \lfloor M_1\ @\ N \rfloor_{\Phi} \end{aligned}$$

- If $M \to N \mathbin{@} V$ with $M_1 \to N$, then:

$$\begin{aligned}\lfloor M_1 \mathbin{@} V \rfloor_\Phi &= |\Phi| \mapsto \lfloor M_1 \rfloor_\Phi (\lfloor V \rfloor_\Phi (|\Phi|), \Phi)\\ &= |\Phi| \mapsto \lfloor M_1 \rfloor_\Phi (\lfloor V \rfloor_\Phi (|\Phi|), \Phi)\\ &= \lfloor N \mathbin{@} V \rfloor_\Phi\end{aligned}$$

- If $M \to M'[V/x]$ with $M_1 = \lambda' x.M'$ and $M_2 = V$, then:

$$\begin{aligned}\Big\lfloor (\lambda' x.M) @ V \Big\rfloor_\Phi &= |\Phi| \mapsto \Big\lfloor \lambda' x.M \Big\rfloor_\Phi (\lfloor V \rfloor_\Phi (|\Phi|), \Phi)\\ &= |\Phi| \mapsto (x, |\Phi| \mapsto \lfloor M \rfloor_{x,\Phi} (x, |\Phi|))(\lfloor V \rfloor_\Phi (|\Phi|), \Phi)\\ &= |\Phi| \mapsto \lfloor M[V/X] \rfloor_\Phi (|\Phi|)\\ &= \lfloor M[V/X] \rfloor_\Phi\end{aligned}$$

- If $M = \texttt{ifz}\ M'\ \texttt{then}\ L\ \texttt{else}\ R$ we have three cases:
  - If $M \to \texttt{ifz}\ N\ \texttt{then}\ L\ \texttt{else}\ R$ with $M' \to N$, then:

$$\begin{aligned}\Big\lfloor \texttt{ifz}\ M'\ \texttt{then}\ L\ \texttt{else}\ R \Big\rfloor_\Phi &= |\Phi| \mapsto \begin{cases}\lfloor M \rfloor_\Phi (|\Phi|) \text{ if } \lfloor M' \rfloor_\Phi (|\Phi|) = 0\\ \lfloor N \rfloor_\Phi (|\Phi|) \text{ otherwise}\end{cases}\\ &= |\Phi| \mapsto \begin{cases}\lfloor M \rfloor_\Phi (|\Phi|) \text{ if } \lfloor M' \rfloor_\Phi (|\Phi|) = 0\\ \lfloor N \rfloor_\Phi (|\Phi|) \text{ otherwise}\end{cases}\\ &= \lfloor \texttt{ifz}\ N\ \texttt{then}\ L\ \texttt{else}\ R \rfloor_\Phi\end{aligned}$$

  - If $M \to L$ with $M' = 0$, then:

$$\begin{aligned}\Big\lfloor \texttt{ifz}\ M'\ \texttt{then}\ L\ \texttt{else}\ R \Big\rfloor_\Phi &= |\Phi| \mapsto \begin{cases}\lfloor M \rfloor_\Phi (|\Phi|) \text{ if } \lfloor M' \rfloor_\Phi (|\Phi|) = 0\\ \lfloor N \rfloor_\Phi (|\Phi|) \text{ otherwise}\end{cases}\\ &= |\Phi| \mapsto \lfloor L \rfloor_\Phi (|\Phi|)\\ &= \lfloor L \rfloor_\Phi\end{aligned}$$

  - The symmetric case for the else branch is similar to the previous one.
- If $M = \texttt{for}\ k\ \texttt{in}\ M'\ \texttt{do}\ R$ we have three cases:
  - If $M \to \texttt{for}\ k\ \texttt{in}\ N\ \texttt{do}\ R$ with $M' \to N$, then:

$$\begin{aligned}\Big\lfloor \texttt{for}\ k\ \texttt{in}\ M'\ \texttt{do}\ R \Big\rfloor_\Phi &= |\Phi| \mapsto \bigtimes_{k \in \lfloor M' \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|)\\ &= |\Phi| \mapsto \bigtimes_{k \in \lfloor N \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|)\\ &= \lfloor \texttt{for}\ k\ \texttt{in}\ N\ \texttt{do}\ R \rfloor_\Phi\end{aligned}$$

  - If $M \to R[k/M_1]\ : \texttt{for}\ k\ \texttt{in}\ M_2\ \texttt{do}\ R$ with $M' = V :: L$, then:

$$\begin{aligned}\lfloor \texttt{for}\ k\ \texttt{in}\ M_1\ ::\ M_2\ \texttt{do}\ R \rfloor_\Phi &= |\Phi| \mapsto \bigtimes_{k \in \lfloor M_1\ ::\ M_2 \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|)\\ &= |\Phi| \mapsto \bigtimes_{k \in \lfloor M_1 \rfloor_\Phi (|\Phi|) \times \lfloor M_2 \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|)\end{aligned}$$

$$= |\Phi| \mapsto \lfloor R \rfloor_{k,\Phi} (\lfloor M_1 \rfloor_\Phi (|\Phi|), |\Phi|) \times \bigtimes_{k \in \lfloor M_2 \rfloor_\Phi (|\Phi|} \lfloor$$

$$= \lfloor R[M_1/k] \rfloor_\Phi \times \lfloor \texttt{for } k \texttt{ in } M_2 \texttt{ do } R \rfloor_\Phi$$

$$= \lfloor R[M_1/k] \ :: \ \texttt{for } k \texttt{ in } M_2 \texttt{ do } R \rfloor_\Phi$$

- If $M \to \texttt{VNil}$ with $M' = \texttt{VNil}^{\texttt{Nat}}$, then:

$$\begin{aligned}
\left\lfloor \texttt{for } k \texttt{ in } \texttt{VNil}^{\texttt{Nat}} \texttt{ do } R \right\rfloor_\Phi &= |\Phi| \mapsto \bigtimes_{k \in \lfloor \texttt{VNil}^{\texttt{Nat}} \rfloor_\Phi (|\Phi|)} \lfloor R \rfloor_\Phi (k, |\Phi|) \\
&= |\Phi| \mapsto \bigtimes_{k \in []} \lfloor R \rfloor_\Phi (k, |\Phi|) \\
&= |\Phi| \mapsto [] \\
&= \left\lfloor \texttt{VNil}^{\texttt{Nat}} \right\rfloor_\Phi
\end{aligned}$$

- If $M = \texttt{let } x :: y = M_1 \texttt{ in } M_2$ we have two cases:
  - If $M \to \texttt{let } x :: y = N \texttt{ in } M_2$ with $M_1 \to N$, then:

$$\begin{aligned}
\lfloor \texttt{let } x :: y = M_1 \texttt{ in } M_2 \rfloor_\Phi &= |\Phi| \mapsto \lfloor M_2 \rfloor_\Phi (y_1, [y_2, \ldots, y_n], |\Phi|) \text{ where } [y_1,\ldots,y_n]= \\
&= |\Phi| \mapsto \lfloor M_2 \rfloor_\Phi (y_1, [y_2, \ldots, y_n], |\Phi|) \text{ where } [y_1,\ldots,y_n]= \\
&= \lfloor \texttt{let } x :: y = N \texttt{ in } M_2 \rfloor_\Phi
\end{aligned}$$

  - If $M \to N[x/M_1][y/M_2]$ with $M' = M_1 :: M_2$, then:

$$\begin{aligned}
\lfloor \texttt{for } k \texttt{ in } M_1 \ :: \ M_2 \texttt{ do } N \rfloor_\Phi &= |\Phi| \mapsto \lfloor N \rfloor_{x,y,\Phi} (y_1, [y_2, \ldots, y_n], |\Phi|) \\
&= |\Phi| \mapsto \lfloor N \rfloor_{x,y,\Phi} (M_1, M_2, |\Phi|) \\
&= |\Phi| \mapsto \lfloor N[M_1/x][M_2/y] \rfloor_\Phi (|\Phi|) \\
&= \lfloor N[M_1/x][M_2/y] \rfloor_\Phi
\end{aligned}$$

    where $[y_1, \ldots, y_n] = \lfloor M_1 :: M_2 \rfloor_\Phi (|\Phi|)$.

- If $M = \texttt{range } \ @n \ @M_2$ then:

$$\begin{aligned}
\lfloor \texttt{range } @n @m \rfloor_\Phi &= |\Phi| \mapsto \bigtimes_{i=n}^{m-1} i \\
&= |\Phi| \mapsto \begin{cases} [] \text{ if } n - m = 0 \\ n \times \bigtimes_{i=n+1}^{m-1} i \text{ otherwise} \end{cases} \\
&= |\Phi| \mapsto \begin{cases} [] \text{ if } \lfloor n - m \rfloor_\Phi = 0 \\ \lfloor n \rfloor_\Phi \times \lfloor \texttt{range } @(n+1) @m) \rfloor \text{ otherwise} \end{cases} \\
&= \lfloor \texttt{ifz } m - n \texttt{ then VNil else } n \ :: \texttt{range } \ @(n+1) \ @m \rfloor_\Phi
\end{aligned}$$

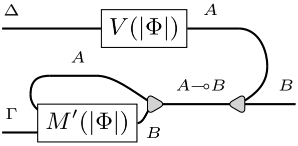
□

Lemma 18. *(7) The translation procedure is correct in respect to the operational semantics. If A is a translatable type, $\Phi, \Gamma \vdash M : A$, and $M \to N$, then $[\![M]\!]_{\Phi,\Gamma} = [\![N]\!]_{\Phi,\Gamma}$.*

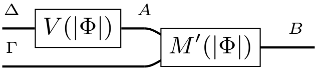
Proof. By case analysis on the reductions of translatable terms.

- If $M = (\lambda x^A.M')V$ and $N = M'[V/x]$,

$$\left[\!\!\left[(\lambda x^A.M')V\right]\!\!\right]_{\Phi,\Delta,\Gamma} = |\Phi| \mapsto$$



$$\overset{(\mathbf{gs})}{=} \;|\Phi| \mapsto$$



$$= [\![M'[V/x]]\!]_{\Phi,\Delta,\Gamma}$$

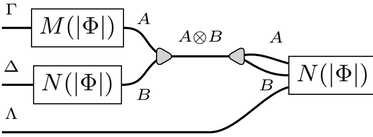
- If $M = (\lambda' x^A.M')@V$ and $N = M'[V/x]$,

$$\left[\!\!\left[(\lambda' x^A.M')@V\right]\!\!\right]_{\Phi,\Gamma} = |\Phi| \mapsto \xrightarrow{\Gamma} \boxed{\twoheadrightarrow (\lambda' x^A.M')_{\Phi}(\twoheadrightarrow V_{\Phi}(|\Phi|), |\Phi|)} \xrightarrow{B}$$

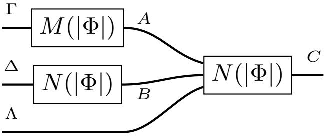
$$= |\Phi| \mapsto \xrightarrow{\Gamma} \boxed{\twoheadrightarrow (M'_{\Phi}(\twoheadrightarrow V_{\Phi}(|\Phi|), |\Phi|)} \xrightarrow{B} = [\![M'[V/x]]\!]_{\Phi,\Gamma}$$

- If $M = \texttt{let } x^A \otimes y^B = V_1 \otimes V_2 \texttt{ in } M'$ and $N = M'[V_1/x][V_2/y]$,

$$\left[\!\!\left[\texttt{let } x^A \otimes y^B = V_1 \otimes V_2 \texttt{ in } M'\right]\!\!\right]_{\Phi,\Gamma,\Delta,\Lambda} = |\Phi| \mapsto$$



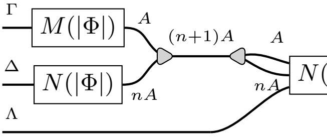
$$\overset{(\mathbf{gs})}{=} \;|\Phi| \mapsto$$



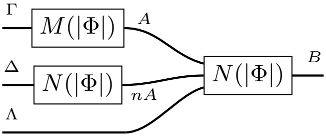
$$= [\![M'[V_1/x][V_2/y]]\!]_{\Phi,\Delta,\Gamma,\Lambda}$$

- If $M = \texttt{let } x^A :: y^{\texttt{Vec}\ n\ A} = V_1 :: V_2 \texttt{ in } M'$ and $N = M'[V_1/x][V_2/y]$,

$$\left[\!\!\left[\texttt{let } x^A :: y^{\texttt{Vec}\ n\ A} = V_1 :: V_2 \texttt{ in } M'\right]\!\!\right]_{\Phi,\Gamma,\Delta,\Lambda} = |\Phi| \mapsto$$



$$\overset{(\mathbf{gs})}{=} \;|\Phi| \mapsto$$

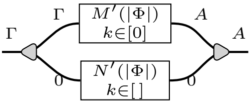


$$= [\![M'[V_1/x][V_2/y]]\!]_{\Phi,\Delta,\Gamma,\Lambda}$$

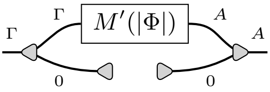
- If $M = \texttt{ifz } L \texttt{ then } M' \texttt{ else } N'$,

– if $L = 0$ and $N = M'$, $\lfloor L \rfloor_\Phi = 0$ and

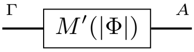
$[\![\texttt{ifz } L \texttt{ then } M' \texttt{ else } N']\!]_{\Phi,\Gamma} = |\Phi| \mapsto$



$\overset{Lemma\ 3}{=} |\Phi| \mapsto$



$\overset{(\emptyset 4)}{=} |\Phi| \mapsto$

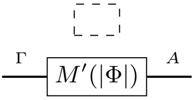


$= [\![$

– The case where $L > 0$ and $N = N'$ is symmetric to the case above.

- If $M = \texttt{VNil}; M'$ and $N = M'$,

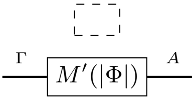
$[\![\texttt{VNil};_v M']\!]_{\Phi,\Gamma} = |\Phi| \mapsto$



$= [\![M']\!]_{\Phi,\Gamma}$

- If $M = \star; M'$ and $N = M'$,

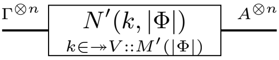
$[\![\star; M']\!]_{\Phi,\Gamma} = |\Phi| \mapsto$



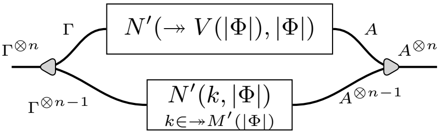
$= [\![M']\!]_{\Phi,\Gamma}$

- If $M = \texttt{for } k \texttt{ in } V :: M' \texttt{ do } N'$ and $N = N'[k/V] :: \texttt{for } k \texttt{ in } M' \texttt{ do } N'$,

$[\![\texttt{for } k \texttt{ in } V :: M' \texttt{ do } N']\!]_{\Phi,\Gamma^{\otimes n}} = |\Phi| \mapsto$



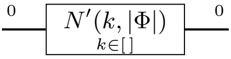
$\overset{\text{Lemma 2}}{=} |\Phi| \mapsto$



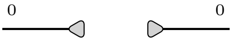
$= [\![N'[k/V] :: \texttt{for } k \texttt{ in } M$

- If $M = \texttt{for } k \texttt{ in VNil do } N'$ and $N = \texttt{VNil}$,

$[\![\texttt{for } k \texttt{ in VNil do } N']\!]_\Phi = |\Phi| \mapsto$



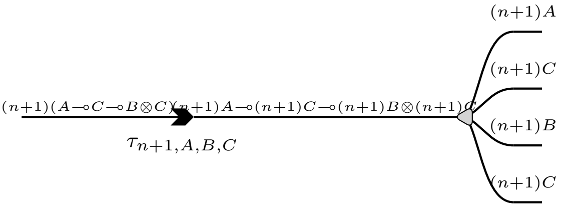
$\overset{Lemma\ 3}{=} |\Phi| \mapsto$



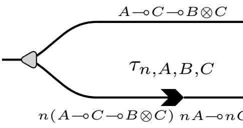
$= |\Phi| \mapsto$

$= [\![\texttt{VNil}]\!]_\Phi$

- If $M = \texttt{accuMap}_{A,B,C}\ N'\ M_{xs}\ M_{fs}\ M_z$ and $N = \texttt{ifz } N' \texttt{ then } xs\ ;_v\ fs\ ;_v\ \texttt{VNil} \otimes M_z \texttt{ else let } x :: xs' = M_{xs} \texttt{ in let } f :: fs' = M_{fs} \texttt{ in let } y \otimes z' = f\ x\ z \texttt{ in let } ys \otimes z'' = \texttt{accuMap}\ @(N'-1)\ xs'\ fs'\ z' \texttt{ in } (y :: ys) \otimes z''$. Let $n = \lfloor N_1 \rfloor_\Phi (|\Phi|)$.
Notice that, by definition of $\tau_{n,A,B,C}$,

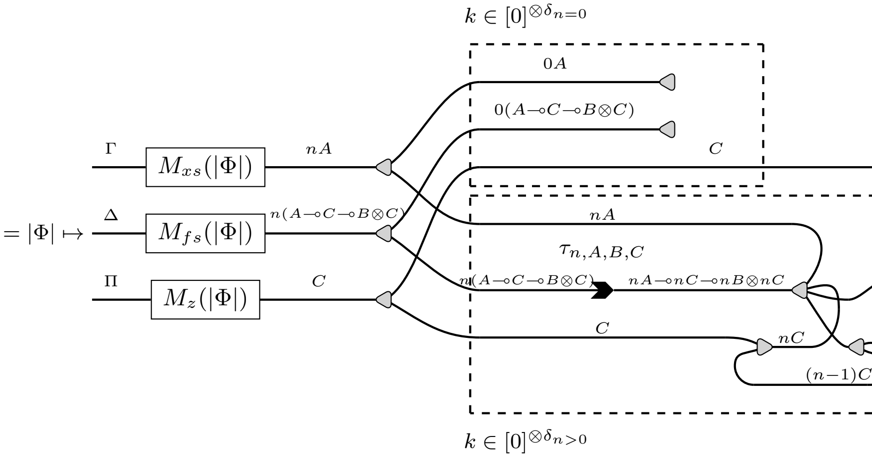

$(n{+}1)(A{\multimap}C{\multimap}B{\otimes}C)$
$(n{+}1)A{\multimap}(n{+}1)C{\multimap}(n{+}1)B{\otimes}(n{+}1)C$
$\tau_{n+1,A,B,C}$
$(n{+}1)A$
$(n{+}1)C$
$(n{+}1)B$
$(n{+}1)C$

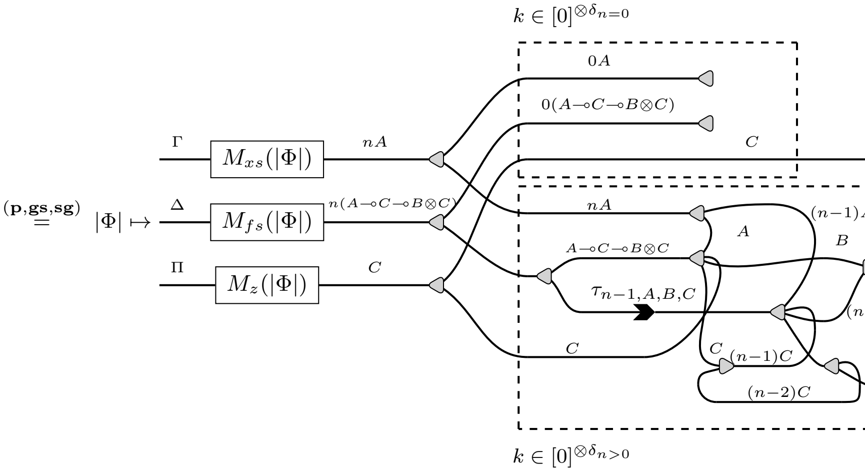
$\overset{(\mathbf{p})}{=}$


$A{\multimap}C{\multimap}B{\otimes}C$
$\tau_{n,A,B,C}$
$n(A{\multimap}C{\multimap}B{\otimes}C)$
$nA{\multimap}n$

Therefore,

$\left\llbracket \mathtt{accuMap}_{A,B,C}\ N'\ M_{xs}\ M_{fs}\ M_z \right\rrbracket_{\Phi,\Gamma,\Delta,\Pi}$

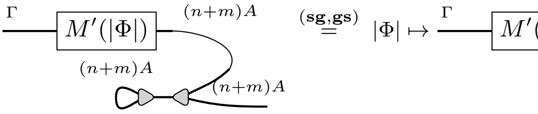

$k \in [0]^{\otimes \delta_{n=0}}$
$0A$
$0(A \multimap C \multimap B \otimes C)$
C
Γ
$M_{xs}(|\Phi|)$
nA
$= |\Phi| \mapsto$
Δ
$M_{fs}(|\Phi|)$
$n(A \multimap C \multimap B \otimes C)$
nA
$\tau_{n,A,B,C}$
Π
$M_z(|\Phi|)$
C
$n(A \multimap C \multimap B \otimes C)$
$nA \multimap nC \multimap nB \otimes nC$
C
nC
$(n-1)C$
$k \in [0]^{\otimes \delta_{n>0}}$


$k \in [0]^{\otimes \delta_{n=0}}$
$0A$
$0(A \multimap C \multimap B \otimes C)$
C
Γ
$M_{xs}(|\Phi|)$
nA
$\overset{(\mathbf{p},\mathbf{gs},\mathbf{sg})}{=}$
$|\Phi| \mapsto$
Δ
$M_{fs}(|\Phi|)$
$n(A \multimap C \multimap B \otimes C)$
nA
$(n-1)$
A
B
$A \multimap C \multimap B \otimes C$
$\tau_{n-1,A,B,C}$
Π
$M_z(|\Phi|)$
C
$(n$
C
C
$(n-1)C$
$(n-2)C$
$k \in [0]^{\otimes \delta_{n>0}}$

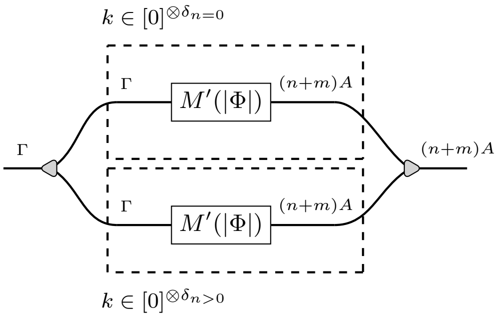
$= \llbracket$ `ifz` N' `then` $xs\ ;_v\ \ fs\ ;_v$ `VNil` $\otimes M_z$ `else let` $x :: xs' = M_{xs}$ `in let` $f :: fs' = M_{fs}$ `in let` $y \otimes z' = f\ x\ z$ `in let` $ys \otimes z'' =$ `accuMap` $@(N'-1)\ xs'\ fs'\ z'$ `in` $(y :: ys) \otimes z''\rrbracket_{\Phi,\Gamma,\Delta,\Pi}$

- If $M = \mathtt{split}_A\ @n\ @m\ xs$ and $N =$ `ifz` n `then VNil` $\otimes\, xs$ `else let` $y :: xs' = xs$ `in let` $ys_1 \otimes ys_2 = \mathtt{split}@(n-1)\ @m\ xs'$ `in` $(y :: ys_1) \otimes ys_2$. Let $n = \lfloor N_1 \rfloor_\Phi (|\Phi|)$ and $m = \lfloor N_2 \rfloor_\Phi (|\Phi|)$.

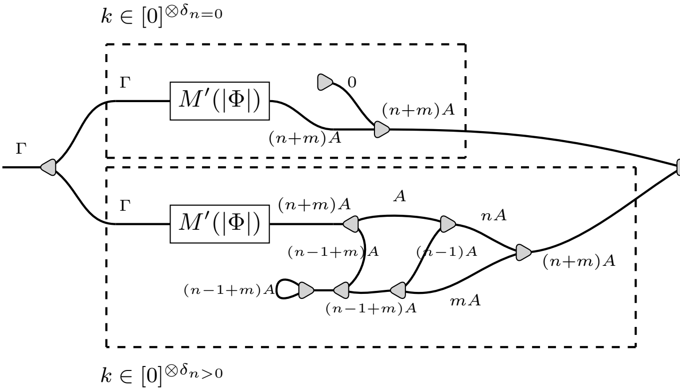
$\llbracket \mathtt{split}_A\ @n\ @m\ xs \rrbracket_{\Phi,\Gamma} = |\Phi| \mapsto$



$\overset{(\emptyset\mathbf{4}), Lemma\ 3}{=} \quad |\Phi| \mapsto$



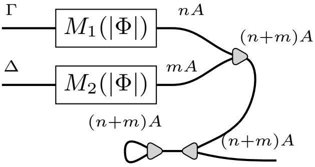
$\overset{(\mathbf{sg}, \mathbf{gs}, \emptyset\mathbf{4})}{=} \quad |\Phi| \mapsto$

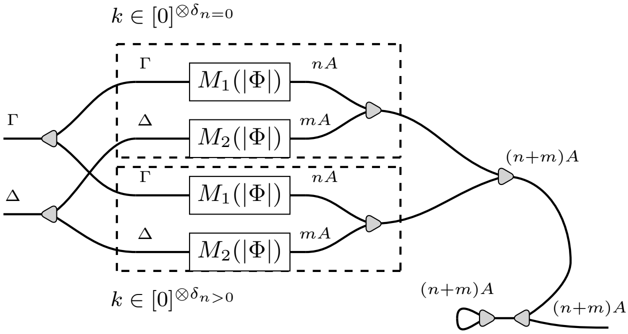


$= [\![$`ifz` n `then` `VNil` $\otimes\, xs$ `else let` $y :: xs' = xs$ `in let` $ys_1 \otimes ys_2 =$ `split`$@(n-1)\ @m\ xs'$ `in` $(y :: ys_1) \otimes ys_2]\!]_{\Phi,\Gamma}$

- If $M =$ `append`$_A\ @N_1\ @N_2\ M_1\ M_2$ and $N =$ `ifz` N_1 `then` $M_1 ;_v\ M_2$ `else let` $x :: xs' = M_1$ `in` $x ::$ (`append` $@(N_1-1)\ @N_2\ M_1\ M_2$). Let $n = \lfloor N_1 \rfloor_\Phi (|\Phi|)$ and $m = \lfloor N_2 \rfloor_\Phi (|\Phi|)$.

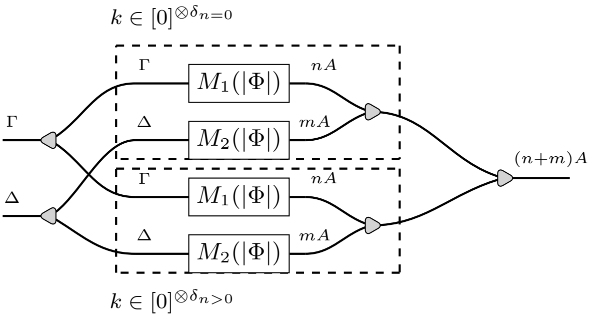
$[\![$`append`$_A\ @N_1\ @N_2\ M_1\ M_2]\!]_{\Phi,\Gamma,\Delta} = |\Phi| \mapsto$



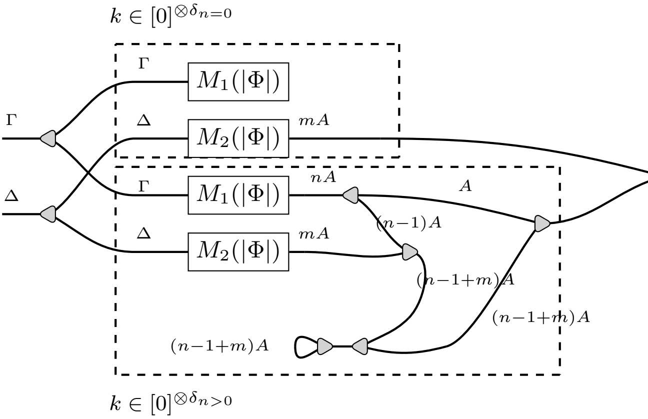


$\overset{(\emptyset\mathbf{4}), Lemma\ 3}{=} \quad |\Phi| \mapsto$

$\overset{(\mathbf{sg},\mathbf{gs})}{=} \quad |\Phi| \mapsto$


$k \in [0]^{\otimes \delta_{n=0}}$
Γ
Δ
$M_1(|\Phi|)$
$M_2(|\Phi|)$
nA
mA
$(n{+}m)A$
$k \in [0]^{\otimes \delta_{n>0}}$

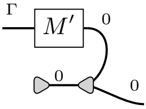
$\overset{(\mathbf{i1},\emptyset\mathbf{1},\emptyset\mathbf{4},\mathbf{sg},\mathbf{gs})}{=} \quad |\Phi| \mapsto$



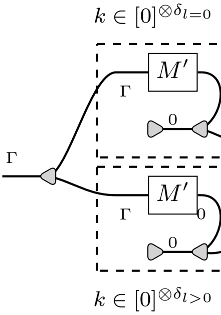
$= [\![\texttt{ifz}\ N_1\ \texttt{then}\ M_1\ ;_v\ \ M_2\ \texttt{else}\ \texttt{let}\ x :: xs' = M_1\ \texttt{in}\ x :: (\texttt{append}\ @(N_1 - 1)\ @N_2\ M_1\ M_2]\!]_{\Phi,\Gamma}$

- If $M = \texttt{drop}\ @N'\ M'$ and $N = \texttt{ifz}\ N'\ \texttt{then}\ M'\ ;\ \star\ \texttt{else}\ \texttt{let}\ x :: xs' = M'\ \texttt{in}$ $x\ ;\ \texttt{drop}\ @(N' - 1)\ xs'$. Let $l = \lfloor N' \rfloor_\Phi\ (|\Phi|)$,

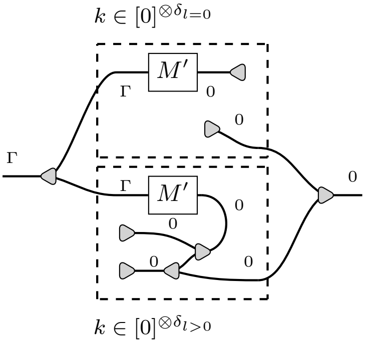
$[\![\texttt{drop}\ @N'\ M']\!]_{\Phi,\Gamma} = |\Phi| \mapsto$



$\overset{(\emptyset\mathbf{4}), Lemma\ 3}{=} \quad |\Phi| \mapsto$


$k \in [0]^{\otimes \delta_{l=0}}$
Γ
M'
Γ
0
M'
Γ
0
0
$k \in [0]^{\otimes \delta_{l>0}}$

$\overset{(\mathbf{z3},\emptyset\mathbf{4})}{=} \quad |\Phi| \mapsto$


$k \in [0]^{\otimes \delta_{l=0}}$
M'
Γ
0
0
Γ
0
Γ
M'
0
0
0
0
$k \in [0]^{\otimes \delta_{l>0}}$

$$= [\![\texttt{ifz}\ N'\ \texttt{then}\ M'\ ;\ \star\ \texttt{else let}\ x :: xs' = M'\ \texttt{in}\ x\ ;\ \texttt{drop}\ @(N'-1)\ xs']\!]_{\Phi,\Gamma}$$

- If $M \to N$ is an internal reduction of a translatable term, then the diagrams result equivalent via the inductive hypothesis.
- If $M \to N$ is an internal reduction of an evaluable term, then the diagrams result equivalent via the inductive hypothesis and Lemma 6.

□

.1 Proofs for validity of LetTens rule

Proposition 5. *For all value distributions $\vec{v_1}, \vec{v_2}, \vec{w_1}, \vec{w_2}$ we have:*

$$\langle (\vec{v_1}, \vec{w_1}) \mid (\vec{v_2}, \vec{w_2}) \rangle = \langle \vec{v_1} \mid \vec{v_2} \rangle \langle \vec{w_1} \mid \vec{w_2} \rangle$$

Proof. Let us write $\vec{v_1} = \sum_{i_1=1}^{n_1} \alpha_{i_1} v_{i_1}$, $\vec{v_2} = \sum_{i_2=1}^{n_2} \alpha'_{i_2} v_{i_2}$, $\vec{w_1} = \sum_{j_1=1}^{m_1} \beta_{j_1} w_{j_1}$ and $\vec{w_2} = \sum_{j_2=1}^{m_2} \beta'_{j_2} w_{j_2}$. Then we have:

$$\begin{aligned}
&\langle (\vec{v_1}, \vec{w_1}) \mid (\vec{v_2}, \vec{w_2}) \rangle \\
&= \langle \sum_{i_1=1}^{n_1} \sum_{j_1=1}^{m_1} \alpha_{i_1} \beta'_{j_1} (v_{i_1}, w_{j_1}) \mid \sum_{i_2=1}^{n_2} \sum_{j_2=1}^{m_2} \alpha_{i_2} \beta'_{j_2} (v_{i_2}, w_{j_2}) \rangle \\
&= \sum_{i_1}^{n_1} \sum_{j_1}^{m_1} \sum_{i_2}^{n_2} \sum_{j_2}^{m_2} \overline{\alpha_{i_1} \beta_{j_1}} \alpha'_{i_2} \beta'_{j_2} \langle (v_{i_1}, w_{j_1}) \mid (v_{i_2}, w_{j_2}) \rangle \\
&= \sum_{i_1}^{n_1} \sum_{j_1}^{m_1} \sum_{i_2}^{n_2} \sum_{j_2}^{m_2} \overline{\alpha_{i_1} \beta_{j_1}} \alpha'_{i_2} \beta'_{j_2} \delta_{(v_{i_1}, w_{j_1}), (v_{i_2}, w_{j_2})} \\
&= \sum_{i_1}^{n_1} \sum_{j_1}^{m_1} \sum_{i_2}^{n_2} \sum_{j_2}^{m_2} \overline{\alpha_{i_1} \beta_{j_1}} \alpha'_{i_2} \beta'_{j_2} \delta_{v_{i_1}, v_{i_2}} \delta_{w_{j_1}, w_{j_2}} \\
&= (\sum_{i_1}^{n_1} \sum_{j_1}^{m_1} \overline{\alpha_{i_1}} \alpha'_{i_2} \delta_{v_{i_1}, v_{i_2}}) (\sum_{i_2}^{n_2} \sum_{j_2}^{m_2} \overline{\beta_{j_1}} \beta'_{j_2} \delta_{w_{j_1}, w_{j_2}}) \\
&= (\sum_{i_1}^{n_1} \sum_{j_1}^{m_1} \overline{\alpha_{i_1}} \alpha'_{i_2} (v_{i_1}, v_{i_2})) (\sum_{i_2}^{n_2} \sum_{j_2}^{m_2} \overline{\beta_{j_1}} \beta'_{j_2} (w_{j_1}, w_{j_2}))
\end{aligned}$$

$$= \langle \vec{v_1} \mid \vec{v_2} \rangle \langle \vec{w_1} \mid \vec{w_2} \rangle$$

□

Lemma 19. *Given a type A, two vectors $\vec{u_1}, \vec{u_2} \in [\![\sharp A]\!]$ and a scalar $\alpha \in \mathbb{C}$, there exists a vector $\vec{u_0} \in [\![\sharp A]\!]$ and a scalar $\lambda \in \mathbb{C}$ such that:*

$$\vec{u_1} + \alpha \vec{u_2} = \lambda \vec{u_0}$$

Proof. Let $\lambda := \|\vec{u_1} + \alpha \vec{u_2}\|$. When $\lambda \neq 0$, we take $\vec{u_0} = \frac{1}{\lambda}(\vec{u_1} + \alpha \vec{u_2}) \in [\![\sharp A]\!]$, and we are done.

When $\lambda = 0$, we first observe that $\alpha \neq 0$ since it would mean that $\|\vec{u_1}\| = 0$ which is absurd since $\|\vec{u_1}\| = 1$. Moreover, since $\lambda = \|\vec{u_1} + \alpha \vec{u_2}\| = 0$, we observe that all the coefficients of the distribution $\vec{u_1} + \alpha \vec{u_2}$ are zeroes when written in canonical form which implies that:

$$\vec{u_1} + \alpha \vec{u_2} = 0(\vec{u_1} + \alpha \vec{u_2}) = 0\vec{u_1} + 0\vec{u_2}$$

Using the triangular inequality we observe that:

$$\begin{aligned} 0 &< 2|\alpha| \\ &= \|2\alpha \vec{u_2}\| \\ &\leq \|\vec{u_1} + \alpha \vec{u_2}\| + \|\vec{u_1} + (-\alpha)\vec{u_2}\| \\ &= \|\vec{u_1} + (-\alpha)\vec{u_2}\| \end{aligned}$$

Hence $\lambda' := \|\vec{u_1} + (-\alpha)\vec{u_2}\| > 0$. Taking $\vec{u_0} := \frac{1}{\lambda'}(\vec{u_1} + (-\alpha)\vec{u_2}) \in [\![\sharp A]\!]$, we easily see that:

$$\vec{u_1} + \alpha \vec{u_2} = 0\vec{u_1} + 0\vec{u_2} = 0(\frac{1}{\lambda'}(\vec{u_1} + (-\alpha)\vec{u_2})) = \lambda \vec{u_0}$$

□

Proposition 6 (Polarization identity)**.** *For all values $\vec{v}$ and $\vec{w}$ we have:*

$$\langle \vec{v} \mid \vec{w} \rangle = \\ \frac{1}{4}(\|\vec{v} + \vec{w}\|^2 - \|\vec{v} + (-1)\vec{w}\|^2 - i\|\vec{v} + i\vec{w}\|^2 + i\|\vec{v} + (-i)\vec{w}\|^2)$$

Lemma 20. *Given a valid typing judgement of the term $\Delta, x_B : \sharp A \vdash \vec{s} : C$, a substitution $\sigma \in [\![\Delta]\!]$ and value distributions $\vec{u_1}, \vec{u_2} \in [\![\sharp A]\!]$, there are value distributions $\vec{w_1}, \vec{w_2} \in [\![C]\!]$ such that:*

$$\begin{aligned} &\vec{s}\langle\sigma\rangle \langle \vec{u_1}/x \rangle_{B_1} \langle \vec{v_1}/y \rangle_{B_2} \twoheadrightarrow \vec{w_1} \\ &\vec{s}\langle\sigma\rangle \langle \vec{u_2}/x \rangle_{B_1} \langle \vec{v_2}/y \rangle_{B_2} \twoheadrightarrow \vec{w_2} \end{aligned}$$

And, $\langle \vec{w_1} \mid \vec{w_2} \rangle = \langle \vec{u_1} \mid \vec{u_2} \rangle$.

Proof. From the validity of the judgement of the form $\Delta, x_A : \sharp A \vdash \vec{s} : C$, a substitution $\sigma \in [\![\Delta]\!]$, and value distributions $\vec{w_1}, \vec{w_2} \in [\![C]\!]$ such that $\vec{s}\langle\sigma\rangle \langle \vec{u_1}/x \rangle_A \twoheadrightarrow \vec{w_1}$ and $\vec{s}\langle\sigma\rangle \langle \vec{u_2}/x \rangle_A \twoheadrightarrow \vec{w_2}$. In particular, we have that $\|\vec{w_1}\| = \|\vec{w_2}\| = 1$. Applying Lemma 19 four times, we know there are vectors $\vec{u_{01}}, \vec{u_{02}}, \vec{u_{03}}, \vec{u_{04}} \in [\![\sharp A]\!]$ and scalars $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ such that:

$$\begin{aligned} \vec{u_1} + \vec{u_2} &= \lambda_1 \vec{u_{01}} \vec{u_1} + i\vec{u_2} = \lambda_3 \vec{u_{03}} \\ \vec{u_1} + (-1)\vec{u_2} &= \lambda_2 \vec{u_{02}} \vec{u_1} + (-i)\vec{u_2} = \lambda_4 \vec{u_{04}} \end{aligned}$$

From the validity of the judgement $\Delta, x_A : \sharp A \vdash \vec{s} : C$, we also know that there are value distributions $\vec{w_{01}}, \vec{w_{02}}, \vec{w_{03}}, \vec{w_{04}} \in [\![C]\!]$ such that $\vec{s}\langle\sigma\rangle\,\langle\vec{u_{0j}}\rangle \twoheadrightarrow \vec{w_{oj}}$ for all $f \in \{1 \cdots 4\}$. Combining the linearity of evaluation on the basis A with the uniqueness of normal forms we deduce from what precedes that:

$$\begin{aligned}\vec{w_1} + \vec{w_2} = \lambda_1 \vec{w_{01}} \vec{w_1} + i\vec{w_2} = \lambda_3 \vec{w_{03}}\\ \vec{w_1} + (-1)\vec{w_2} = \lambda_2 \vec{w_{02}} \vec{w_1} + (-i)\vec{w_2} = \lambda_4 \vec{w_{04}}\end{aligned}$$

Using the polarization identity (Prop 6), we conclude that:

$$\begin{aligned}&\langle \vec{w_1} \mid \vec{w_2} \rangle\\ &= \frac{1}{4}(\|\vec{w_1} + \vec{w_2}\| - \|\vec{w_1} + (-1)\vec{w_2}\| - i\|\vec{v_1} + i\vec{v_2}\| + i\|\vec{v_1} + (-i)\vec{v_2}\|)\\ &= \frac{1}{4}((\lambda_1)^2\|\vec{w_{01}}\| - (\lambda_2)^2\|\vec{w_{02}}\| - i(\lambda^2_)\|\vec{w_{03}}\| + i(\lambda^2_)\|\vec{w_{04}}\|)\\ &= \frac{1}{4}((\lambda_1)^2\|\vec{u_{01}}\| - (\lambda_2)^2\|\vec{u_{02}}\| - i(\lambda^2_)\|\vec{u_{03}}\| + i(\lambda^2_)\|\vec{u_{04}}\|)\\ &= \frac{1}{4}(\|\vec{u_1} + \vec{u_2}\| - \|\vec{u_1} + (-1)\vec{u_2}\| - i\|\vec{u_1} + i\vec{u_2}\| + i\|\vec{u_1} + (-i)\vec{u_2}\|)\\ &= \langle u_1 \mid u_2 \rangle\end{aligned}$$

□

Lemma 21. *Given a valid typing judgement of the form $\Delta, x_{B_1} : \sharp A_1, y_{B_2} : \sharp A_2 \vdash \vec{s} : C$, a substitution $\sigma \in [\![\Delta]\!]$ and value distributions $\vec{u_1}, \vec{u_2} \in [\![\sharp A]\!]$, there are value distributions $\vec{w_1}, \vec{w_2} \in [\![C]\!]$ such that:*

$$\begin{aligned}\vec{s}\langle\sigma\rangle\,\langle\vec{u_1}/x\rangle_{B_1}\langle\vec{v_1}/y\rangle_{B_2} \twoheadrightarrow \vec{w_1}\\ \vec{s}\langle\sigma\rangle\,\langle\vec{u_2}/x\rangle_{B_1}\langle\vec{v_2}/y\rangle_{B_2} \twoheadrightarrow \vec{w_2}\end{aligned}$$

And, $\langle \vec{w_1} \mid \vec{w_2} \rangle = 0$.

Proof. From Lemma 19 we know that there are $\vec{u_0} \in [\![\sharp A]\!], \vec{v_0} \in [\![\sharp B]\!]$ and $\lambda, \mu \in \mathbb{C}$ such that:

$$\vec{u_2} + (-1)\vec{u_1} = \lambda\vec{u_0} \quad \text{and} \quad \vec{v_2} + (-1)\vec{v_1} = \mu\vec{v_0}$$

For all $j, k \in \{0, 1, 2\}$, we have $\vec{s}\langle\sigma\rangle\,\langle\vec{u_j}/x\rangle_{B_1}\langle\vec{v_k}/y\rangle_{B_2} \twoheadrightarrow \vec{w_{jk}}$. In particular, we can take $\vec{w_1} = \vec{w_{11}}$ and $\vec{w_2} = \vec{w_{22}}$. Now we observe that:

1. $\vec{u_1} + \lambda\vec{u_0} = \vec{u_1} + \vec{u_2} + (-1)\vec{u_1} = \vec{u_2} + 0\vec{u_1}$, so that from linearity of substitution, linearity of evaluation and uniqueness of normal forms, we get:
$$\begin{aligned}\vec{w_{1k}} + \lambda\vec{w_{0k}} = \vec{w_{2k}} + 0\vec{w_{1k}}\\ \vec{w_{2k}} + (-\lambda)\vec{w_{0k}} = \vec{w_{1k}} + 0\vec{w_{2k}}\end{aligned} \qquad \text{(for all } k \in \{0, 1, 2\}\text{)}$$
2. $\vec{v_1} + \mu\vec{v_0} = \vec{v_1} + \vec{v_2} + (-1)\vec{v_1} = \vec{v_2} + 0\vec{v_1}$, so that from linearity of substitution, linearity of evaluation and uniqueness of normal forms, we get:
$$\begin{aligned}\vec{w_{j1}} + \mu\vec{w_{j0}} = \vec{w_{j2}} + 0\vec{w_{j1}}\\ \vec{w_{j2}} + (-\mu)\vec{w_{j0}} = \vec{w_{j1}} + 0\vec{w_{j2}}\end{aligned} \qquad \text{(for all } j \in \{0, 1, 2\}\text{)}$$
3. $\langle \vec{u_1} \mid \vec{u_2} \rangle = 0$, so that from Lemma 20 we get $\langle \vec{w_{1k}} \mid \vec{w_{2k}} \rangle = 0$ (for all $k \in \{0, 1, 2\}$).
4. $\langle \vec{v_1} \mid \vec{v_2} \rangle = 0$, so that from Lemma 20 we get $\langle \vec{w_{j1}} \mid \vec{w_{j2}} \rangle = 0$ (for all $j \in \{0, 1, 2\}$).

From the above, we get:

$$\begin{aligned}
\langle \vec{w_1} \mid \vec{w_2} \rangle = \langle \vec{w_{11}} \mid \vec{w_{22}} \rangle &= \langle \vec{w_{11}} \mid \vec{w_{22}} + 0\vec{w_{12}} \rangle \\
&= \langle \vec{w_{11}} \mid \vec{w_{12}} + \lambda \vec{w_{02}} \rangle && \text{(from (1), } k = 2\text{)} \\
&= \langle \vec{w_{11}} \mid \vec{w_{12}} \rangle + \lambda \langle \vec{w_{11}} \mid \vec{w_{02}} \rangle \\
&= 0 + \lambda \langle \vec{w_{11}} \mid \vec{w_{02}} \rangle && \text{(from (4), } j = 1\text{)} \\
&= \lambda \langle \vec{w_{11}} + 0\vec{w_{21}} \mid \vec{w_{02}} \rangle \\
&= \lambda \langle \vec{w_{21}} + (-\lambda)\vec{w_{01}} \mid \vec{w_{02}} \rangle && \text{(from (1), } k = 1\text{)} \\
&= \lambda \langle \vec{w_{21}} \mid \vec{w_{02}} \rangle - |\lambda|^2 \langle \vec{w_{01}} \mid \vec{w_{02}} \rangle \\
&= \lambda \langle \vec{w_{21}} \mid \vec{w_{02}} \rangle - 0 && \text{(from (4), } j = 0\text{)} \\
&= \langle \vec{w_{21}} \mid \vec{w_{22}} - \vec{w_{12}} \rangle \\
&= \langle \vec{w_{21}} \mid \vec{22} \rangle - \langle \vec{w_{21}} \mid \vec{w_1}2 \rangle \\
&= 0 - \langle \vec{w_{21}} \mid \vec{w_{12}} \rangle && \text{(from (4), } j = 2\text{)}
\end{aligned}$$

Hence $\langle \vec{w_1} \mid \vec{w_2} \rangle = \langle \vec{w_{11}} \mid \vec{w_{22}} \rangle = -\langle \vec{w_{21}} \mid \vec{w_{12}} \rangle$. Exchanging the indices in the previous reasoning, we also get

$$\langle \vec{w_1} \mid \vec{w_2} \rangle = -\langle \vec{w_{21}} \mid \vec{w_{12}} \rangle = -\langle \vec{w_{12}} \mid \vec{w_{21}} \rangle$$

So that we have:

$$\langle \vec{w_1} \mid \vec{w_2} \rangle = -\langle \vec{w_{21}} \mid \vec{w_{12}} \rangle = -\overline{\langle \vec{w_{21}} \mid \vec{w_{12}} \rangle} \in \mathtt{R}$$

If we now replace $\vec{u_2} \in [\![\sharp A]\!]$ with $i\vec{u_2} \in [\![\sharp A]\!]$, the very same technique allows us to prove that $i\langle \vec{w_1} \mid \vec{w_2} \rangle = \langle \vec{w_1} \mid i\vec{w_2} \rangle \in \mathtt{R}$. Therefore, $\langle \vec{w_1} \mid \vec{w_2} \rangle = 0$. □

Lemma 22. *Given a valid typing judgement of the form $\Delta, x_{B_1} : \sharp A_1, y_{B_2} : \sharp A_2 \vdash \vec{s} : C$, a substitution $\sigma \in [\![\Delta]\!]$, and value distributions $\vec{u_1}, \vec{u_2} \in [\![\sharp A]\!]$ and $\vec{v_1}, \vec{v_2} \in [\![\sharp B]\!]$, there are value distributions $\vec{w_1}, \vec{w_2} \in [\![C]\!]$ such that:*

$$\begin{aligned}
&\vec{s}\langle \sigma \rangle \langle \vec{u_1}/x \rangle_{B_1} \langle \vec{v_1}/y \rangle_{B_2} \twoheadrightarrow \vec{w_1} \\
&\vec{s}\langle \sigma \rangle \langle \vec{u_2}/x \rangle_{B_1} \langle \vec{v_2}/y \rangle_{B_2} \twoheadrightarrow \vec{w_2}
\end{aligned}$$

And, $\langle \vec{w_1} \mid \vec{w_2} \rangle = \langle \vec{u_1} \mid \vec{u_2} \rangle \langle \vec{v_1} \mid \vec{v_2} \rangle$.

Proof. Let $\alpha = \langle \vec{u_1} \mid \vec{u_2} \rangle$ and $\beta = \langle \vec{v_1} \mid \vec{v_2} \rangle$. We observe that:

$$\langle \vec{u_1} \mid \vec{u_2} + (-\alpha)\vec{u_1} \rangle = \langle \vec{u_1} \mid \vec{u_2} \rangle - \alpha \langle \vec{u_1} \mid \vec{u_1} \rangle = \alpha - \alpha = 0$$

And similarly that, $\langle \vec{v_1} \mid \vec{v_2} + (-\beta)\vec{v_1} \rangle = 0$. From Lemma 19, we know that there are $\vec{u_0} \in [\![\sharp A]\!]$, $\vec{v_0} \in [\![\sharp B]\!]$ and $\lambda, \mu \in \mathbb{C}$ such that:

$$\vec{u_2} + (-\alpha)\vec{u_1} = \lambda \vec{u_0} \text{ and } \qquad \vec{v_2} + (-\beta)\vec{v_1} = \mu \vec{v_0}$$

For all $j, k \in \{0, 1, 2\}$, we have$\langle \sigma \rangle \langle \vec{u_j}/x \rangle_{B_1} \langle \vec{v_k}/y \rangle_{B_2} \in [\![\Delta, x_{B_1} : \sharp A_1, y_{B_2} : \sharp A_2]\!]$, hence there is $\vec{w_{jk}} \in [\![C]\!]$ such that:

$$\vec{s}\langle \sigma \rangle \langle u_j/x \rangle_{B_1} \langle \vec{v_k}/y \rangle_{B_2} \twoheadrightarrow \vec{w_{jk}}$$

In particular, we can take $\vec{w_1} = \vec{w_{11}}$ and $\vec{w_2} = \vec{w_{22}}$. Now we observe that:

1. $\lambda \vec{u_0} + \alpha \vec{u_1} = \vec{u_2} + (-\alpha)\vec{u_1} + \alpha \vec{u_1} = \vec{u_2} + 0\vec{u_1}$, so that from the linearity of the substitution, linearity of evaluation and uniqueness of normal forms, we get:

$$\lambda \vec{w_{0k}} + \alpha \vec{w_{1k}} = \vec{w_{2k}} + 0\vec{w_{1k}} \qquad \text{(for all } k \in \{0, 1, 2\}\text{)}$$

2. $\mu\vec{v_0} + \beta\vec{v_1} = \vec{v_2} + (-\beta)\vec{v_1} + \beta\vec{v_1} = \vec{v_2} + 0\vec{v_1}$, so that from the linearity of the substitution, linearity of evaluation and uniqueness of normal forms, we get:

$$\mu\vec{w_{j0}} + \beta\vec{w_{j1}} = \vec{w_{j2}} + 0\vec{w_{j1}} \qquad \text{(for all } j \in \{0,1,2\}\text{)}$$

3. $\langle\vec{u_1} \mid \lambda\vec{u_0}\rangle = \langle\vec{u_1} \mid \vec{u_2} + (-\alpha)\vec{u_1}\rangle = 0$, so that from Lemma 20 we get:

$$\langle\vec{w_{1k}} \mid \lambda\vec{w_{0k}}\rangle = 0 \qquad \text{(for all } k \in \{0,1,2\}\text{)}$$

4. $\langle\vec{v_1} \mid \mu\vec{v_0}\rangle = \langle\vec{v_1} \mid \vec{v_2} + (-\beta)\rangle\vec{v_1} = 0$, so that from Lemma 20 we get:

$$\langle\vec{w_{j1}} \mid \mu\vec{w_{j0}}\rangle = 0$$

5. $\langle\vec{u_1} \mid \lambda\vec{u_0}\rangle = \langle\vec{v_1} \mid \mu\vec{v_0}\rangle = 0$ so that from Lemma 21 we get:

$$\langle\vec{w_{11}} \mid \lambda\mu\vec{w_{00}}\rangle = 0$$

(Again the equality $\langle\vec{w_{11}} \mid \lambda\mu\vec{w_{00}}\rangle$ is trivial when $\lambda = 0$ or $\mu = 0$. When $\lambda, \mu \neq 0$ we deduce from the above that $\langle\vec{u_1} \mid \vec{u_0}\rangle = \langle\vec{v_1} \mid \vec{v_0}\rangle = 0$, from which we get $\langle\vec{w_{11}} \mid \vec{w_{00}}\rangle = 0$ by Lemma 21)

From above, we get:

$$\begin{aligned}
&\vec{w_{22}} + 0\vec{w_{12}} + 0\vec{w_{01}} + 0\vec{w_{11}} \\
&\qquad = \lambda\vec{w_{02}} + \alpha\vec{w_{12}} + 0\vec{w_{01}} + 0\vec{w_{11}} && \text{(from 1}, k = 1\text{)} \\
&\qquad = \lambda(\vec{w_{02}} + 0\vec{w_{01}}) + \alpha(\vec{w_{12}} + 0\vec{w_{11}}) \\
&\qquad = \lambda(\mu\vec{w_{00}} + \beta\vec{w_{01}}) + \alpha(\mu\vec{w_{01}} + \beta\vec{w_{11}}) && \text{(from 2}, j = 0, 1\text{)} \\
&\qquad = \lambda\mu\vec{w_{00}} + \lambda\beta\vec{w_{01}} + \alpha\mu w_{10} + \vec{\alpha}\beta\vec{w_{11}}
\end{aligned}$$

Therefore:

$$\begin{aligned}
&\langle\vec{w_1} \mid \vec{w_2}\rangle \\
&= \langle\vec{w_{11}} \mid \vec{w_{22}} + 0\,\vec{w_{12}} + 0\,\vec{w_{01}} + 0\,\vec{w_{11}}\rangle \\
&= \langle\vec{w_{11}} \mid \lambda\mu\vec{w_{00}} + \lambda\beta\vec{w_{01}} + \alpha\mu w_{10} + \vec{\alpha}\beta\vec{w_{11}}\rangle \\
&= \langle\vec{w_{11}} \mid \lambda\mu\vec{w_{00}}\rangle + \langle\vec{w_{11}} \mid \lambda\beta\vec{w_{01}}\rangle + \langle\vec{w_{11}} \mid \alpha\mu\vec{w_{10}}\rangle + \langle\vec{w_{11}} \mid \alpha\beta\vec{w_{11}}\rangle \\
&= \lambda\mu\langle\vec{w_{11}} \mid \vec{w_{00}}\rangle + \lambda\beta\langle\vec{w_{11}} \mid \vec{w_{01}}\rangle + \alpha\mu\langle\vec{w_{11}} \mid \vec{w_{10}}\rangle + \alpha\beta\langle\vec{w_{11}} \mid \vec{w_{11}}\rangle \\
&= 0 + 0 + 0 + \alpha\beta = \langle\vec{u_1} \mid \vec{u_2}\rangle\langle\vec{v_1} \mid \vec{v_2}\rangle
\end{aligned}$$

From 5, 3 and 4 with $j = 1$ and concluding with the definition of α and β. □

List of Figures

List of Tables

Bibliography

[1] Miriam Backens and Aleks Kissinger. ZH: A complete graphical calculus for quantum computations involving classical non-linearity. *Electronic Proceedings in Theoretical Computer Science*, 287:23–42, jan 2019. 2.4.2

[2] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski, and John van de Wetering. There and back again: A circuit extraction tale. *Quantum*, 5:421, mar 2021. 2.1

[3] Agustín Borgna, Simon Perdrix, and Benoît Valiron. Hybrid Quantum-Classical Circuit Simplification with the ZX-Calculus. In Hakjoo Oh, editor, *Programming Languages and Systems*, pages 121–139, Cham, 2021. Springer International Publishing. 2.1

[4] Titouan Carette, Yohann D'Anello, and Simon Perdrix. Quantum Algorithms and Oracles with the Scalable ZX-calculus. *Electronic Proceedings in Theoretical Computer Science*, 343:193–209, September 2021. 2.1, 2.2.1, 2.2.2, .1

[5] Titouan Carette, Dominic Horsman, and Simon Perdrix. SZX-calculus: Scalable Graphical Quantum Reasoning. *arXiv:1905.00041 [quant-ph]*, page 15 pages, 2019. arXiv: 1905.00041. 2.1, 2.2.1

[6] Titouan Carette, Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of graphical languages for mixed states quantum mechanics, 2019. .1

[7] Iliano Cervesato and Frank Pfenning. A linear logical framework. *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 264–275, 1996. 2.3

[8] Bob Coecke and Simon Perdrix. Environment and classical channels in categorical quantum mechanics. *Logical Methods in Computer Science*, Volume 8, Issue 4, November 2012. 2.2.1

[9] Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. *Quantum*, 4:218, January 2020. tex.ids: debeaudrapZXCalculus-Language2020a arXiv: 1704.08670. 2.1

[10] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439:553 – 558, 1992. 3.5.1

[11] Cirq Developers. Cirq, August 2021. See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors. 2.1

[12] Alejandro Díaz-Caro. Towards a computational quantum logic: An overview of an ongoing research program, 2025. Invited paper. 3.4.1

[13] Alejandro Díaz-Caro, Mauricio Guillermo, Alexandre Miquel, and Benoît Valiron. Realizability in the unitary sphere. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019)*, pages 1–13, 2019. 3.1, 3, 3.4.2, 3.4.2, 3.6

[14] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic simplification of quantum circuits with the zx-calculus, 2019. 2.1

[15] Alejandro Díaz-Caro and Octavio Malherbe. Quantum control in the unitary sphere: Lambda-s1 and its categorical model. *Logical Methods in Computer Science*, Volume 18, Issue 3, September 2022. 3.2.1

[16] Alejandro Díaz-Caro and Nicolas A. Monzon. A quantum-control lambda-calculus with multiple measurement bases, 2025. 3.1

[17] MD SAJID ANIS et al. Qiskit: An open-source framework for quantum computing, 2021. 2.1

[18] Peng Fu, Kohei Kishida, Neil J. Ross, and Peter Selinger. A tutorial introduction to quantum circuit programming in dependently typed Proto-Quipper. *arXiv:2005.08396 [quant-ph]*, December 2020. arXiv: 2005.08396. 2.1, 2.2

[19] Peng Fu, Kohei Kishida, and Peter Selinger. Linear Dependent Type Theory for Quantum Programming Languages. *arXiv:2004.13472 [quant-ph]*, December 2021. arXiv: 2004.13472. 2.1

[20] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. *ACM SIGPLAN Notices*, 48(6):333, Jun 2013. 2.1, 2.2

[21] Stephen C. Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):109–124, 1945. 3.1

[22] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011. 2.5

[23] Simon Perdrix. *Quantum Entanglement Analysis Based on Abstract Interpretation*, page 270–282. Springer Berlin Heidelberg, 2008. 3.1

[24] Damian S. Steiger, Thomas Häner, and Matthias Troyer. Projectq: an open source software framework for quantum computing. *Quantum*, 2:49, Jan 2018. 2.1

[25] John van de Wetering. Zx-calculus for the working quantum computer scientist, 2020. 2.1, 2.2.1